Technical University of Crete, Greece

School of Electrical and Computer Engineering

Preferences-Aware Social Ridesharing

Emmanouil Pagkalos

Chania, October 2021

Abstract

Ridesharing is a shared-economy transportation paradigm, in which people can catch rides in vehicles that are (possibly) privately owned and driven by others, in order to reduce transportation costs and possibly enjoy a more pleasant ride, when compared to those offered by alternative means of transportation. Despite its touted advantages, e.g. in terms of increased socialization and potentially huge positive environmental impact, Ridesharing has not gained much popularity yet. Arguably, for this to happen Ridesharing needs to offer a clearly more-pleasant-than-others transportation experience to people who use it.

In this thesis, we approach the Ridesharing problem via Artificial Intelligence solution concepts, originating primarily in the Multiagent Systems, graph-theoretic, and game-theoretic research literature; and, importantly, we take into consideration the riders' (agents) preferences about attributes of their co-riders. Taking into account such preferences, bears the potential to greatly impact positively the participants' satisfaction.

Our thesis offers an initial, but complete, framework for preferences-aware Ridesharing. We first employ the concept of a hypergraph to the set of agents, in order to create an initial clustering given the areas where they move. We then use a greedy algorithm and a branch and bound algorithm in order to distribute the agents into vehicles with the aim of (a) maintaining the drivers' detours to as low levels as possible; and, at the same time, (b) satisfying most of the expressed agents' preferences. Furthermore, we put forward a cost-sharing scheme to compensate drivers for their participation in the scheme and the extra costs this entails, and to create incentives for participation via overall drivers' costs reduction.

We studied systematically the performance of our Ridesharing framework via simulation scenarios run on maps of the four main cities of Crete: Heraklion, Chania, Rethymno, and Agios Nikolaos. Our results show that (a) our approach has high effectiveness in terms of covering the transportation needs of the (non-driving) commuters; (b) at the same time, the average extra distance that drivers need to cover, when offering their services to passengers, is kept to acceptable levels; (c) the costs of the drivers are substantially reduced when compared to their costs, when not participating in Ridesharing; and (d) the agents' preferences are satisfied to a large degree, a fact which arguably improves their whole Ridesharing experience. Notably, our simulation results are particularly encouraging for scenarios with much-lower-than-current vehicle ownership (drivers' percentage), thus underscoring the potential of Ridesharing in a future, greener, largely free of privately-owned vehicles, world.

Abstract in Greek

Το Ridesharing είναι μια συγκεκριμενοποίηση της λεγόμενης οικονομίας διαμοιρασμού στον τομέα των μετακινήσεων. Η χρήση του Ridesharing επιτρέπει την μεταφορά ατόμων με οχήματα που ανήκουν και οδηγούνται από άλλους (πιθανώς) ιδιώτες, προκειμένου οι εμπλεκόμενοι αφενός να μειώσουν το κόστος μετακίνησής τους, και αφετέρου να απολαύσουν ενδεχομένως μια πιο ευχάριστη διαδρομή σε σύγκριση με αυτές που προσφέρονται από εναλλακτικά μέσα μεταφοράς. Παρά τα προβαλλόμενα πλεονεκτήματά του, λ.χ. από την άποψη της αυξημένης κοινωνικοποίησης των εμπλεκόμενων, καθώς και των δυνητικά τεράστιων θετικών περιβαλλοντικών επιπτώσεών του (λόγω μείωσης της ανάγκης για ιδιόκτητα μέσα μεταφοράς), το Ridesharing δεν έχει αποκτήσει ακόμη μεγάλη δημοτικότητα. Κατά πάσα πιθανότητα, για να συμβεί αυτό το Ridesharing πρέπει να προσφέρει μια σαφώς πιο ευχάριστη, σε σχέση με άλλες, εμπειρία μετακίνησης σε όσους το χρησιμοποιούν.

Στην παρούσα διπλωματική εργασία, προσεγγίζουμε το πρόβλημα Ridesharing μέσω λύσεων Τεχνητής Νοημοσύνης, προερχόμενων κατά κύριο λόγο από τα πεδία των Πολυπρακτορικών Συστημάτων, της Θεωρίας Γράφων, και της Θεωρίας Παιγνίων. Ένα κομβικής σημασίας στοιχείο στην προσέγγισή μας είναι ότι, για πρώτη φορά στη βιβλιογραφία, λαμβάνουμε υπόψη τις προτιμήσεις των εμπλεκομένων σχετικά με τα χαρακτηριστικά των συνεπιβατών τους. Αυτό είναι σημαντικό, καθώς είναι αναμενόμενο η κάλυψη τέτοιων προτιμήσεων να αυξήσει σημαντικά το συνολικό επίπεδο ικανοποίησης των συμμετεχόντων από την υπηρεσία.

Η διπλωματική μας εργασία προσφέρει ένα αρχικό, αλλά πλήρες πλαίσιο, για κοινή χρήση προτιμήσεων στο Ridesharing. Αρχικά χρησιμοποιούμε την έννοια των υπεργράφων, προκειμένου να δημιουργήσουμε μια αρχική ομαδοποίηση των πρακτόρων, λαμβάνοντας υπόψη τις περιοχές όπου κινούνται. Στη συνέχεια, χρησιμοποιούμε έναν άπληστο αλγόριθμο και έναν αλγόριθμο διακλάδωσης και δέσμευσης για να διανείμουμε τους πράκτορες σε οχήματα, με στόχο (α) να διατηρήσουμε τις παρακάμψεις των οδηγών σε όσο το δυνατόν χαμηλότερα επίπεδα και, ταυτόχρονα, (β) να ικανοποιήσουμε τις εκφρασμένες προτιμήσεις των πρακτόρων στο μέγιστο δυνατό βαθμό. Επιπλέον, προτείνουμε ένα σύστημα επιμερισμού του κόστους, για να αποζημιώσουμε τους οδηγούς για τη συμμετοχή τους στο πρόγραμμα και το επιπλέον κόστος που αυτή συνεπάγεται, και για να μειώσουμε τα έξοδά τους σε σχέση με τη μη συμμετοχή τους και άρα να δημιουργήσουμε κίνητρα συμμετοχής.

Μελετήσαμε συστηματικά την απόδοση του προτεινόμενου πλαισίου Ridesharing μέσω σεναρίων προσομοίωσης που εκτελούνται σε χάρτες των τεσσάρων κύριων πόλεων της Κρήτης: το Ηράκλειο, τα Χανιά, το Ρέθυμνο και τον Άγιο Νικόλαο. Τα αποτελέσματά μας δείχνουν ότι (α) η προσέγγισή μας έχει υψηλή αποτελεσματικότητα, όσον αφορά την

κάλυψη των αναγκών μετακίνησης των εν δυνάμει επιβατών (που δεν διαθέτουν όχημα), (β) ταυτόχρονα, η μέση επιπλέον απόσταση που πρέπει να διανύσουν οι οδηγοί για να προσφέρουν τις υπηρεσίες τους στους επιβάτες διατηρείται σε αποδεκτά επίπεδα, (γ) τα κόστη που πληρώνουν οι οδηγοί είναι σημαντικά μειωμένα σε σχέση με τα κόστη τους όταν δεν συμμετέχουν στην υπηρεσία Ridesharing , και (δ) οι προτιμήσεις των πρακτόρων ικανοποιούνται σε μεγάλο βαθμό, γεγονός το οποίο αναμφισβήτητα βελτιώνει ολόκληρη την εμπειρία χρήσης του Ridesharing. Αξιοσημείωτα, τα αποτελέσματα προσομοίωσης είναι ιδιαίτερα ενθαρρυντικά για σενάρια με πολύ χαμηλότερη ιδιοκτησία οχήματος από την τρέχουσα (χαμηλό ποσοστό οδηγών), υπογραμμίζοντας έτσι τις δυνατότητες που ανοίγει η χρήση Ridesharing σε έναν μελλοντικό, πιο πράσινο, σε μεγάλο βαθμό χωρίς ιδιόκτητα οχήματα, κόσμο.

Acknowledgements

I would like to express my gratitude to those who helped make this thesis possible.

First and foremost, I want to deeply thank my supervisor, Associate Professor Georgios Chalkiadakis, for all his support, guidance and trust throughout my thesis. I would also like to thank all my friends for motivating me, and especially the members of the Intelligent Systems Laboratory, Dimitris and Errikos. I appreciate every single comment and advice from our discussions.

Second, I want to express my gratitude towards the members of the thesis committee, Associate Professor Michail G. Lagoudakis and Associate Professor Panagiotis Partsinevelos, for their helpful comments and time.

Last, but not least, I cannot thank my family enough, Giannis, Rita and Nikos, who have supported me all my life and they never stopped believing in me.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

In everyday life, each individual needs to ensure the way of transportation from a starting point to a destination, e.g., from home to work. There are multiple ways of transportation such as taxis, bus, subway etc. Deciding among them largely depends on a comparison between their convenience and their cost. For example, it might be preferable to use a taxi over the bus; however, it comes with a higher price. An innovative way of transportation is Ridesharing [1, 2]. It is based on a concept where people who own cars cooperate with people who do not, in order to share rides and slit their expenses. Several variants of Ridesharing have been studied. An example is the efficient management of vehicles in order to serve a group of people using the minimum possible resources so that the overall cost is reduced [3, 4]. Other works focus on ordinary people who are willing to share their cars during their daily transportation with the aim of decreasing their expenses [5].

The advantages of Ridesharing are significant and that could possibly distinguish it to one of the most popular ways of transportation in the future. Reputable companies as Maramoja[1] and Lyft[2] already offer Ridesharing services to users. Drivers can share their cars and passengers can find a seat. Through Ridesharing a notable reduction of vehicles on the road can be achieved, which leads to a relaxation of congestion [4]. In addition, it is an eco-friendly idea, since greenhouse emissions are limited and a big amount of energy from electric cars can be saved. The welfare of people is improved, since public transportation is avoided and the cost of moving from a place to another can be reduced.

[1]https://maramoja.co.ke/
[2]https://www.lyft.com/

1. INTRODUCTION

Most of the works on Ridesharing focus on minimizing the number of trips, and subsequently the cost, through effective route determination [2]. Some approaches are suitable for companies which possess vehicles such as Uber[1], in order to manage their resources efficiently and serve their clients with the minimum cost [4]. Other works aim at matching participants so to reduce the total number of trips without enlarging detours and without causing unacceptable delays [3, 6]. Generally, Ridesharing pursue the merge of trips for participants with similar routes in order to reduce the number of vehicles on road and at the same time, it offers a potentially quite pleasant way of transportation.

Now, the field of artificial intelligence which studies the interaction between two or more intelligent agents is called Multiagent Systems, also denoted as MAS [7, 8]. The purpose of MAS is the solution of complicated problems where multiple agents are involved. In some cases, either their interests are conflicting or they need to cooperate in order to achieve common goals. Single self-interested agents often cannot accomplish difficult tasks alone, so they need to coordinate their actions in order to complete a challenge and share the profit. Aspects of the Ridesharing problem can be largely studied via *Cooperative Game Theory* which nowadays constitutes a sub-field of MAS [8]. Cooperative Game Theory focuses on problems where agents form coalitions with the aim to accomplish some targets. In this thesis, agents form coalitions which correspond to cars so that passengers are matched to seats in order to reach their destinations without using public transportation. Drivers offer empty seats of their vehicles in order to share the cost with other people. Obviously, in large problems where the number of agents is huge, potentially in the hundreds or thousands, the space of possible coalitions is enormous, so we need an effective approach in order to examine only the most promising solutions.

In this thesis, we approach the Ridesharing problem while taking into consideration agents' preferences about attributes of their co-riders. Apart from travel plans compatibility, we consider that agents who constitute a coalition should also feel comfortable when sharing the same car. To the best of our knowledge, this is done for the first time in the literature. In the next subsection, we provide an overview of our contributions.

[1]https://www.uber.com/

1.1 Contributions

The common goal of every study on Ridesharing is the efficient distribution of agents into vehicles in order to fulfil some requirements. These requirements depend on each study; however, most works focus on minimizing the overall cost, which means that routes must be as few as possible. In addition, some works include extra time constraints [4, 6, 2]. Apart from a route, each agent sets his time of departure. Hence, agents who cooperate need to have spatial and temporal compatibility. In this thesis, for the first time in the literature we turn the focus on preferences of agents about attributes of people with whom they are more willing to cooperate. For example, a college student might prefer to socialise with other students, while a middle-aged person might prefer to cooperate with people of her age. Therefore, it is natural to assume that all the agents of the system move simultaneously. In our work, we did not include the factor of time in order to emphasise on preferences and evaluate how much they are satisfied. Our goal is to satisfy most of the agents, i.e., to match each agent to a coalition, without enlarging the detours for drivers excessively. Our approach is based on partitioning of the city map. Furthermore, we apply a branch and bound method along with a main greedy algorithm. We aim at minimizing the number of passengers who will not join any coalition due to spatial incompatibility. At the same time, we want to keep the extra distance for drivers at low levels. To this end, and in a nutshell, we combine ideas originating mainly in graph theory, game theory, and intelligent search, in order to determine a coalition structure (a partition of the agents' space) which approaches the optimal in terms of time travel and agents' preferences regarding co-riders. We provide a thorough evaluation of our approach, focusing on the degree of agents' preferences satisfaction achieved, but also on the reduction of the drivers' costs and methods' execution time. Our results show case the robustness of our approach, and the fact that it can be employed for scheduling short-notice rides in realistic (reasonably sized) multiagent settings.

1.2 Outline

Chapter 2 contains all the necessary theoretical background which constitutes the foundations of our approach. We provide an analysis of *Graph Theory* and we examine the *Dijkstra's algorithm*. We also present a generalized form of graphs, called the *Hypergraphs*. Furthermore, we discuss about *Coalition Formation* and *Characteristic Function Games*. Lastly, we present some related work on the concept of Ridesharing. In Chapter 3, we introduce our approach on the Ridesharing problem. First, we set our requirements and restrictions. We explain in detail the application of hypergraph in order to partition the problem before moving to the coalition formation. Subsequently, we present our algorithms for the final determination of the coalition structure. An algorithm for the route extraction of each coalition is provided in order to evaluate our method. Chapter 4 consists of the experimental setup and the experimental evaluation of our approach. First, we present the data infrastructure where we applied our approach, i.e., our case study. Afterwards, we provide a detailed representation of the observed results and we discuss the performance. Finally, Chapter 5.1 is an epilogue of our work along with conclusions and suggestions about future work on this subject.

Chapter 2

Theoretical Background & Related Work

This chapter covers the theoretical background of our approach. Furthermore, we discuss related works and studies on Ridesharing.

2.1 Theoretical Background

In this section, we present the necessary theoretical background of this thesis. First, we present a brief analysis of Graph Theory along with common types of Graphs and Trees. We continue with Pathfinding and specifically we focus on Dijkstra's algorithm which is used in our implementation. Furthermore, we analyze Hypergraphs and finally we discuss Coalition Formation, Characteristic Function Games and the Shapley value.

2.1.1 Graph Theory

In everyday life we face practical problems where graph theory is applied in order to compute fast and reliable solutions. Multiple science areas such as applied mathematics, electrical engineering and computer science constantly enhance their ability to solve complicated tasks by using graphs. Electrical circuits, data structures and computation of optimal solutions are some of these problems. A graph is a structure made by a set of discrete objects and a set of connections between them. Those objects can represent anything—e.g., people, cities or vehicles—and formally they are referred to as vertices or nodes. A connection between two object stands for an interaction and it is called an edge. Formally, the definition of a graph is the following:

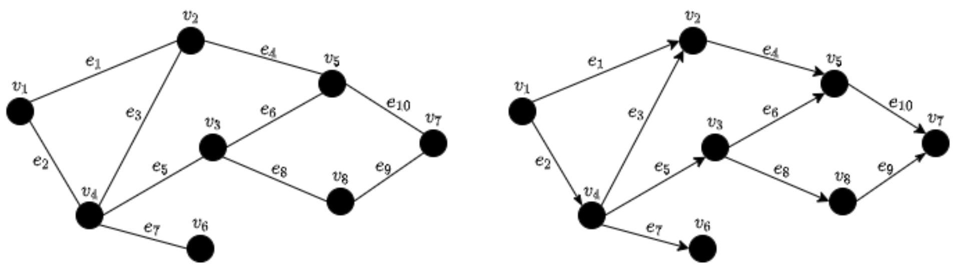
Definition 1 **([9])** *A graph G=(V,E) consists of a set of objects* $V = \{v_1, v_2, ..., v_n\}$ *called vertices, and another set* $E = \{e_1, e_2, ..., e_m\}$*, whose element are called edges, such that each edge* e_k *is identified with an unordered pair* (v_i, v_j) *of vertices. The vertices* v_i*,* v_j *associated with edge* e_k *are called the end vertices of* e_k*.*

The graph of Figure 2.1(a) contains eight vertices and ten edges among them. It is a simple graph since it does not contain any self-loop or parallel edges. Self-loop is an edge which connects a node to itself while parallel edges are two edges that share the same end vertices. This is a way of representation for objects that interact with each other. As an example, nodes could symbolize a set of adjacent cities and the edges correspond to roads which connect them.

This kind of graphs is called an *Undirected Graph*. That means, every edge between any two vertices a and b is mutual, a interacts with b the same way that b interacts with a. However, sometimes this is not the case and there is a need to clarify the direction for each edge. Hence, an edge e_k is considered to go from vertex a to vertex b or vice versa. A graph which contains edges with specified directions is defined as *Directed Graph*.

Definition 2 **([9])** *A directed graph (or a digraph for short) G consists of a set of vertices* $V = \{v_1, v_2, ..., v_n\}$*, a set of edges* $E = \{e_1, e_2, ..., e_m\}$*, and a mapping* Ψ *that maps every edge onto some ordered pair of vertices* (v_i, v_j)*.*

Directed Graphs are used in multiple cases where we need to define flow or non-symmetrical interaction between vertices. For example, in a social network where a person a might know person b but person b does not know a, this is visualized by a Directed Graph. The edge between a and b is written as (a, b) but not as (b, a). The difference in representation between Undirected and Directed Graphs is the presence of arrows which define the directions of the edges as shown in Figure 2.1(b).



(a) Undirected Graph (b) Directed Graph

Figure 2.1: Simple Graphs

Some applications use *Complete Graphs* in order to represent inner interactions among objects. This kind of graphs contain undirected edges between any possible pair of vertices. The formal definition is the following:

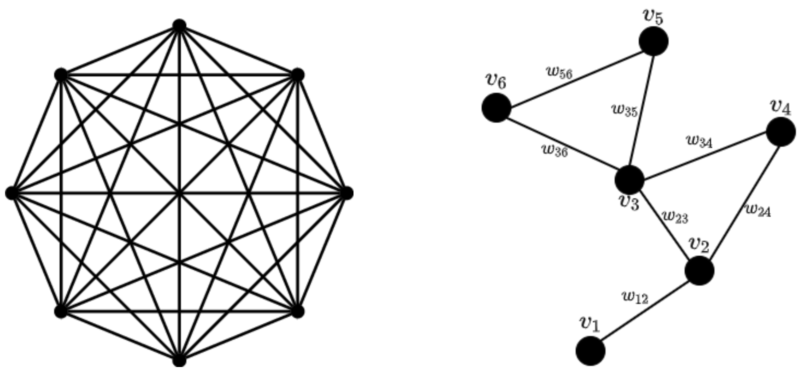
Definition 3 ([10]) *A simple graph G in which each pair of distinct vertices v_i and v_j are adjacent is a complete graph.*

A Complete Graph consisted of n vertices, K_n, contains exactly $\binom{n}{2} = \frac{n(n-1)}{2}$ edges. This kind of graphs can be used in applications where objects constitute a closed group and each object interacts with any other.

Another widely used kind of graphs is the *Weighted Graph*. This graph is very practical and it is used in order to solve daily problems such as finding the shortest route or calculate the minimum cost. It is a Directed or Undirected Graph which assigns to each edge a numerical value.

Definition 4 ([11]) *A weighted graph is a graph in which a number $w(e)$, called its weight is assigned to each edge e.*

This value is also referred to as *cost* because it is usually a measure of quantities such as distance, expense or time consumption. Weighted Graphs are used in multiple applications and problems in computer science. A well known example is the Travelling Salesman Problem [12].



(a) Complete Graph (b) Weighted Graph

Figure 2.2: Graph Structures

In some cases, our interest is concentrated on a limited part of the graph which contains a subset of vertices and edges. We ignore the rest of the graph and we extract information only from this *Subgraph*. Hence, a *Subgraph* is defined as follows:

Definition 5 ([9]) *A graph S is said to be a subgraph of graph G if all the vertices and all the edges of S are in G, and each edge of S has the same end vertices in S as in G.*

We denote a graph S as a subgraph to G by $S \subseteq G$. Clearly, a graph can be considered as a subgraph of its own. A subgraph H of graph S is also a subgraph of G if $S \subseteq G$. Finally, a single vertex of a graph G or a single edge, along with its end vertices, are also considered as subgraphs of G.

One of the most important and commonly used kind of graphs is definitely *Trees*. A Tree is a connected graph which does not contain cycles. A cycle is a path that begins and terminates at the same vertex. The formal definition for Trees is the following:

Definition 6 ([13]) *If G is a connected graph without any cycles then G is called a Tree. (If $|V| = 1$, then G is connected and hence is a Tree).*

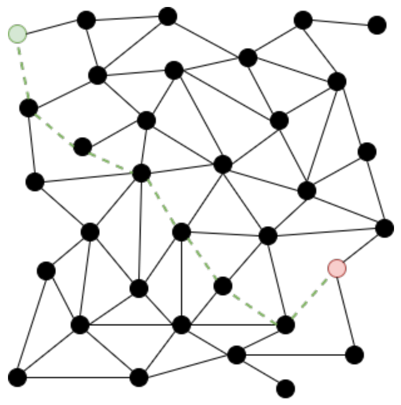
Every Tree satisfies some constraints which make Trees so useful and special. A Tree is a minimal connected graph where the number of vertices is always one more than the number of edges. Every pair of vertices (a, b) with $a \neq b$ is connected by exactly one path. Removing any edge from a Tree leads to a non connected graph. Trees are used in multiple areas of computer science, one of them is Game Theory where *Game Trees* are used in order to predict game's progress and pick the optimal action [14].

2.1.2 Pathfinding Algorithms

A big challenge in programming is searching for a path between a staring point A and a destination point B. Usually, we need to find the optimal path which connects some points of interest. Optimal solution depends on the purpose of the application. For example, we might search for the shortest path or the path which requires the least cost if we consider additional parameters such as inconvenience, congestion or time consumption.

Pathfinding algorithms are mostly used in applications including maps, mazes and networks. In some problems, the solution is even more complicated as there are restrictions which have to be satisfied in order to accept the outcome as feasible. Restrictions might be obstacles or spatial and priority constraints. This kind of algorithms are very useful because they calculate solutions for problems such as routing packets efficiently, navigate people to unknown regions and in our case calculate optimal route for a single person or for a group of people. Pathfinding is applied on graphs where a vertex A represents the origin and a vertex B represents the destination. The goal is to find the optimal path which connects these vertices. If the graph is weighted and each weight corresponds to a cost, then the optimal path is the one which has the least sum of weights. Otherwise, if the graph has no weights, we are interested in the path which contains the least number of edges. Hence, primarily we need to define what optimal stands for because it differs depending on the application.

As mentioned above, pathfinding is applied on graphs or on grids. A grid is a pattern or structure made from horizontal and vertical lines crossing each other to form squares. It is used in various applications, some popular examples are chess, snake and Tetris. Of course, we can represent a grid as a graph where every square, mostly called cell, is connected to its adjacent cells. In order to use a pathfinding algorithm in an application including a map, we need to convert the map to the form of a graph. On doing so, we can use any pathfinding algorithm we prefer. The most well known algorithms are *BFS*, *DFS*, *Dijkstra's algorithm* and *A* algorithm*. In our approach we use the Dijkstra's algorithm as it is considered to be the best algorithm for map-based pathfinding because of its simplicity, effectiveness and reliability.


(a) Graph

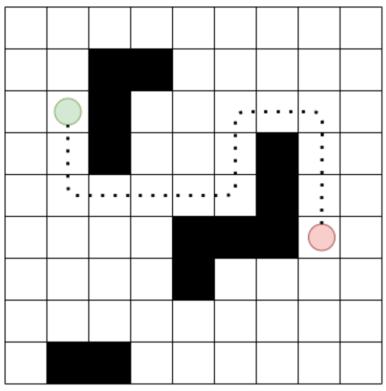

(b) Grid

Figure 2.3: Pathfinding

2.1.3 Dijkstra's Algorithm

Dijkstra's algorithm is definitely one of the most popular and widely used algorithms in computer science [15, 16]. It was invented by the Dutch computer scientist Edsger W. Dijkstra in 1956. It aims at finding the shortest path between a source node and any other node of the graph. Nevertheless, usually it is applied in order to find the shortest path between a starting point and a specific destination. This algorithm is very effective in pathfinding and specifically it is applied to weighted graphs. By shortest path we mean the route which causes the minimum cost [17], either it refers to time, distance or inconvenience. We search for the route among multiple potential routes which has the least sum of individual costs.

As mentioned in section 2.1.1, the weight of an edge usually represents the distance between the two end vertices when the graph is a representation of a city. However, in some cases the weight is a product of a combination taking into account multiple parameters, e.g., size of road, congestion, speed limit etc. Simplicity and effectiveness make Dijkstra's algorithm the protagonist in shortest path finding problems based on maps. Dijkstra's algorithm is used by GPS and Google maps due to its reliability. The implementation is based on exploring and evaluating every possible alternative route from source node to destination [18]. For every node, there is information about the minimum distance from source, i.e., the distance of the shortest route, and the node which precedes.

Hence, by backward induction we return from destination to source through the optimal route.

At this point, we examine the function of Dijkstra's algorithm. We refer to the starting node as source. This node is our reference point and every distance is calculated according to this. For any other node, its distance shows how far it is from the source. For instance, given a vertex V which has distance 6, this means that until the current point of the process we have found that the optimal path from source to V comes with a cost of 6. At first, the algorithm assigns nodes' distances to extremely large values and subsequently through searching, it reduces those values gradually. Obviously, the only zero distance is from source to itself, we assume that every weight of the graph is a positive value [18]. We keep a set which contains every visited node. The purpose is to avoid visiting the same nodes twice and continue exploring new vertices of the graph.

The procedure starts from the source node. We find all of its adjacent nodes and we assign their real distances. Next, we pick from the set of the non-visited nodes the one which has the lowest distance and we repeat with its adjacent nodes. If the node that we currently check, let's denote it as V, has distance d_1 and a neighbour node U is far from V by d_2, then the overall distance of U is $d_1 + d_2$. If this amount is lower than the current distance of U then we update it. Otherwise, we maintain the current value. We repeat this process with the remaining non-visited nodes until we reach the destination. Algorithm 1 presents Dijkstra's function in pseudocode. In case we need to find the distance for every node of the graph, we skip the lines 11 - 13, the procedure continues until every node has been visited.

In this thesis we suggest a solution for the Ridesharing problem [1, 2] applied to medium size cities. We use city graphs which are basically a simpler way of representation for the map. Every node of the graph depicts an intersection, edges define roads and weights represent the distance in kilometers between two points. This distance can be calculated by longitude and latitude of the nodes, this is an information which is available for every point. We apply Dijkstra's algorithm to determine the route that a driver needs to follow in order to move from her starting point to her destination. In next section 3.2, we explain how a driver's route is used in order to match the driver with potential passengers. In addition, after the coalition formation, we apply an algorithm to calculate the optimal route for the whole trip. A car might includes extra passengers apart from driver and some restriction must be satisfied in order to accept the trip as feasible. In this work, we use Dijkstra's algorithm to determine the sub-routes of the trip.

Algorithm 1 DIJKSTRA(*source*, *destination*)

```
1: set $unvisited$;
2: for $v$ in vertices do
3:     $v.distance = \infty$;
4:     $v.parent = NULL$;
5:     $unvisited$.insert($v$);
6: end for
7: $source.distance = 0$
8: while (!$unvisited$.empty) do
9:     $v = getVertexMinDistance(unvisited)$;
10:    $unvisited$.remove($v$);
11:    if ($v == destination$) then
12:        return $v$.path;
13:    end if
14:    for $u$ in neighbors of $v$ do
15:        $d = v.distance + weight(v, u)$;
16:        if ($d < u.distance$) then
17:            $u.distance = d$;
18:            $u.parent = v$;
19:        end if
20:    end for
21: end while
```

2.1.4 Hypergraphs

In section 2.1.1, we discussed the significance and the benefits of graphs in computer science. However, in some cases graphs are probably not the optimal option because they get exceedingly large and dense. We can approach complex problems more efficiently with a generalized form of graphs known as *Hypergraphs*. As presented in [19], a hypergraph is essentially a graph whose edges may connect multiple vertices, i.e., more that two. This is the main distinction between (simple) graphs and hypergraphs. A formal definition is given by Valdivia et al. in [20].

Definition 7 ([20]) *A hypergraph $G = (V, H) \mid H$ is a set of h hyperedges, i.e., edges that can connect any number of vertices. Formally, $h \in \mathcal{P}(V)$ where $\mathcal{P}(V)$ is the set all sets of V (a.k.a power-set of V)*

Hypergraphs are more flexible and clearer in illustration. An example of hypergraph is presented in Figure 2.4.

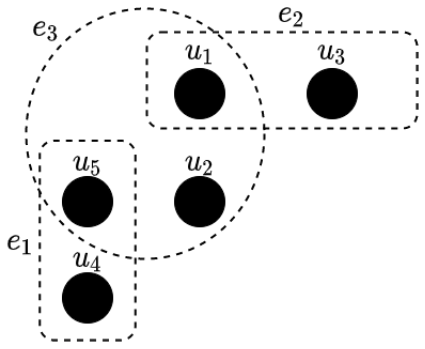
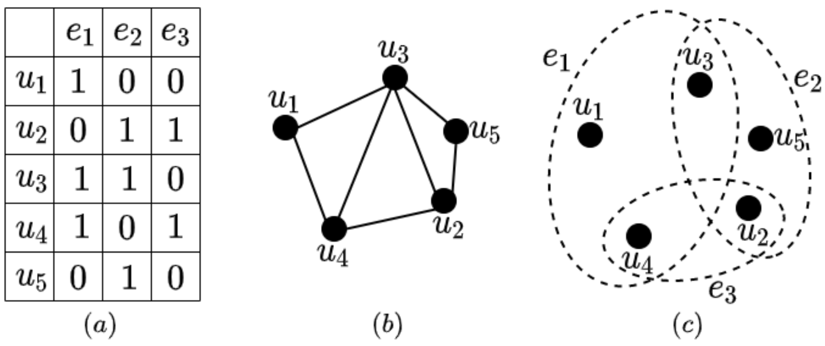


Figure 2.4: A Simple Hypergraph

In the example above, hypergraph G contains five vertices $V = \{u_1, u_2, u_3, u_4, u_5\}$ and three hyperedges $H = \{e_1, e_2, e_3\}$. As we notice, vertex u_5 belongs to hyperedges $\{e_1, e_3\}$, and hyperedge e_2 contains vertices $\{u_1, u_3\}$. In graphs, vertices normally correspond to objects while edges represent the interaction between them. In hypergraphs, a hyperedge usually refers to a common characteristic of the given set of vertices. For instance, consider that the vertices of Figure 2.4 correspond to cars, while the hyperedges refer to characteristics such as electric, petrol or automatic—e.g., if edge e_2 refers to automatic vehicles and e_3 refers to vehicles using petrol—then we know that car u_1 is an automatic petrol car.

A significant advantage of hypergraphs over (simple) graphs is the ability to include more information which contributes in object classification and clustering [21]. In some cases, the pairwise connection between objects is not sufficient to properly capture all the intrinsic properties of the problem at hand. Let's consider the example of Figure 2.5 where vertices correspond to movies and edges correspond to movie genres. We need a method that classifies movies in order to provide such information to a recommendation system. On the left side of this Figure, the table shows whether a movie belongs to a genre or not. According to the table, movie u_2 belongs to genres $\{e_2, e_3\}$, but not to e_1. A graph can connect movies which appear to belong to same genres. As such, a recommendation system could suggest movies which are somehow connected. However, the information about movies' character is not conspicuous. By applying a hypergraph in order to illustrate the case, as shown on the right side of Figure, we have the whole picture for the problem. For large-scale problems with huge amount of input data, the use of hypergraphs over (simple) graphs is preferable as data management is improved and the information is more compact.



	e_1	e_2	e_3
u_1	1	0	0
u_2	0	1	1
u_3	1	1	0
u_4	1	0	1
u_5	0	1	0

Figure 2.5: Hypergraph - Simple Graph Comparison

2.1.5 Coalition Formation

An interesting field of Multiagent Systems [22] is certainly *coalition formation*. This deals with agents' need of cooperation in order to achieve complicated tasks. Cooperation means that agents form groups with the aim of accomplishing missions that single self-interested agents cannot complete on their own or maximizing the overall welfare [23]. That leads to higher profit for each member. As described in [8], we distinguish two classes of coalition formation. The first class deals with agents who are not selfish and they are willing to cooperate in order to fulfil the requirements of a single system designer. By contrast, the second class covers cases where participants are selfish, rational agents whose motivation on joining a coalition is the maximization of their own profit or the fulfilment of a goal that they cannot achieve alone. The significance of coalition formation arises from everyday life challenges where independent individuals benefit from collaborating. Ridesharing is a characteristic example, agents who do not own a car are interested in joining a coalitions, i.e., a car, in order to reach their destination. On the other hand, drivers aim at minimizing their expenses by offering rides to co-participants. Coalition formation is a fundamental part of *cooperative games*, also known as *coalitional games* [24, 8].

As presented in [25], coalition formation can be separated into three states. Primarily, the *coalition structure generation* (CSG), this is basically the procedure of allocating agents to disjoint coalitions. Agents who consist a coalition interact with each other by coordinating their actions. Nevertheless, usually there is no interaction between distinct coalitions. The whole set of coalitions is formally referred to as *coalition structure*. Additionally, a coalition which includes a single agent is called a *singleton*, while a coalition which includes every agent of the set is called *grand coalition*.

The second state is known as *solving the optimization problem*. At this stage, we aim at maximizing coalition's profit by efficient use of abilities and resources. Alongside, we want to keep coalition's cost as low as possible. In other words, it is a determination of the optimal balance between profit and cost so that participants have no intention of leaving the coalition.

Finally, *dividing the value* is the process where total profit is distributed among members. Distribution has a crucial essence, it has to be fair and proportional to each member's contribution. As such, no player has a motive for deviating.

2.1.6 Characteristic Function Games

Characteristic Function Games, mainly denoted as CFG, provide the main theoretical framework for formation problems. Such games, which consist of N players (agents), a coalition C is a subset of the whole set of agents, $C \subseteq N$. Characteristic function games and coalition formation are directly connected because in every (CFG) there is a characteristic function which attaches a numerical value to every coalition. This function is essentially a utility function assigning utility to coalitions.

Definition 8 ([8]) *A characteristic function game G is given by a pair (N, v), where $N = \{1, ..., n\}$ is a finite, non-empty set of agents and $v : 2^N \to \mathbb{R}$ is a characteristic function, which maps each coalition $C \subseteq N$ to a real number $v(C)$. The number $v(C)$ is usually referred to as the value of the coalition C.*

Provided the utility function, in many cases we aim at maximizing the overall welfare of the coalition structure, i.e., the sum of utilities from every coalition to be the maximum possible.

$$v^*(CS) = \arg\max_{CS \in \Gamma(N)} \sum_{C \in CS} v(C)$$

Given the set N of agents, $\Gamma(N)$ is the set of all possible coalition structures.

2.1.7 Shapley Value and Induced Subgraph Games

The third state of coalition formation is *dividing the value* and distributing it among coalition's members. However, it is necessary to guarantee that the distribution is fair for every agent depending on their contribution. A solution concept was introduced by Lloyd Shapley in 1951 [26]. The Shapley value, as it is called, aims to determine fairly the share for each member so that no-one has a motive to deviate from coalition. By contribution, we refer to how much an agent improves the overall coalition's value when she joins it.

Before the definition of Shapley value, we present all the needed notation [8]. Given a characteristic function game $G = (N, v)$, we denote as Π_N the set of every possible *permutation* of N. A permutation is denoted as π and it belongs to Π_N, $\pi \in \Pi_N$. For instance, if $N = \{n, u, v\}$ then

$$\Pi_N = \{(n,u,v),(n,v,u),(u,n,v),(u,v,n),(v,n,u),(v,u,n)\} \tag{2.1}$$

Having a permutation π at hand, we denote as $S_\pi(i)$ the set which contains every predecessor of i in π. In continuation of the example above, given the permutation $\pi = (n, v, u)$ we notice that $S_\pi(n) = \emptyset$, $S_\pi(v) = \{n\}$, $S_\pi(u) = \{n, v\}$. Furthermore, we define the amount $\Delta_\pi^G(i)$ which represents how much agent i improves the value of the coalition which contains his predecessors from permutation π by joining them. The calculation of $\Delta_\pi^G(i)$ is given by the equation 2.2.

$$\Delta_\pi^G(i) = v\big(S_\pi(i) \cup \{i\}\big) - v\big(S_\pi(i)\big) \tag{2.2}$$

At this point, we can present the formal definition of Shapley value for an agent i.

Definition 9 ([8]) *Given a characteristic function game $G = (N, v)$ with $|N| = n$, the Shapley value of a player $i \in N$ is denoted by $\phi_i(G)$ and is given by*

$$\phi_i(G) = \frac{1}{n!} \sum_{\pi \in \Pi_N 1} \Delta_\pi^G(i)$$

In short, the Shapley value as defined in Definition 9, is the average marginal contribution among every possible permutation of N. It provides for every agent a measure of his contribution in coalition's value in order to distribute the gains in a realistic way.

Induced Subgraph Games (ISG) constitute a simple but important category of characteristic function games [27, 28] because they encompass many realistic settings where agents' collaborations are restricted by a graph. These games are presented by an undirected weighted graph $G(N, E)$ where N is the set of agents and E is the set of edges. The corresponding weight of an edge which connects two agents, A_i and A_j is denoted as $w_{i,j}$. In ISG, the Shapley value of an agent i can be easily computed by,

$$\phi_i(G) = \frac{1}{2} \sum_{\substack{(i,j) \in E \\ i \neq j}} w(i,j) \tag{2.3}$$

The equation 2.3 presents a much simpler way of calculation for the Shapley value. It is preferable to use this one in cases where the characteristic function game can be represented as an ISG. In section 3.4, we discuss the use of Shapley value in our implementation.

2.2 Related Work

Ridesharing is a widely researched topic in multiagent systems with variety of versions, approaches and purposes. Bicocchi et al. in [5] present a Recommend System for users according to their mobility habits. They aim to match people who could potentially share a ride. For each user there is a set of the most visited places such as home or work. Mobility analysis extracts a pattern for each user which is compared to other patterns in order to find similarities and finally to recommend ride partners. However, in this work the role for each user, driver or passenger, is not datum. It depends on their routes as the user with the containing route has to be the driver. Provided that users are willing to walk 1 km in order to be matched for a sharing ride, their experimental results show a reduction of 60% of the trips.

The purpose of Pelzer et al. in [3] is to allocate passengers to cars in such a way to maximize the utilization of space while minimizing the inconvenience due to detours. Hence, the main restriction is to keep inconvenience under a bound. This approach is based on partitioning of the road network in order to decompose the main problem in several sub-problems. Subsequently, a match making algorithm combines drivers and passengers once their routes coincide. The model used in this work is the single rider - single driver, meaning that apart from driver a car may contain one more passenger. Experimental results show a reduction of 42% of the daily trips.

Most works use the model of single driver-single rider. Alonso-Mora et al. in [4] present an anytime algorithm which aims at serving multiple passengers per vehicle. In this work, agents are not separated into drivers and passengers. The concept is to combine agents with similar routes and allocate them to vehicles in such a way to manage efficiently taxis and vans. The purpose is to serve the clients with as few as possible resources. Their method achieves a high reduction of vehicles during peak time. Specifically, 13000 taxis serving single client each, could be replaced by 3000 taxis of capacity four under the assumption that participants are willing to wait for 3.2 minutes with an extra delay of 1.5 minute.

Similarly, in [6] Riley et al. pursue minimization of waiting time as well as keeping time delays from detours at a low level. It is an approach that aims at optimal management of vehicles in order to serve reliably every client. Therefore, drivers are not subjects of the process as they serve clients' request and they do not have transportation request of their own. This work is based on a column-generation algorithm where time is divided into slots in order to serve gradually all requests. Their results guarantee that every client is served within a tolerable time frame and without significant deviation from their shortest route.

Bistaffa et al. [2] have a different approach of social Ridesharing as the participants constitute a social network. A coalition is considered feasible only if its members form a subgraph of the network. This restriction defines this problem as a Graph-Constrained Coalition Formation. This approach focuses on minimising the travel cost of the overall system while satisfying spatial and temporal preferences of participants. In addition, they present a payment allocation algorithm which distributes travel cost to coalition's members according to their contribution. Their method shows a reduction of overall cost up to 36.22% compare to a scenario without Ridesharing.

The main distinction between this thesis and all the works presented above is the preference aggregation. In our work, we handle preferences regarding potential co-riders which has not be done in any other related work to the best of our knowledge. We consider agents' preferences about co-passengers in order to combine agents who would feel more comfortable when collaborating.

Chapter 3

Our Approach on the Ridesharing Problem

In this chapter, we will analyse our approach for solving the Ridesharing problem [2]. As we mentioned in the Introduction, in this work we focus on forming feasible coalitions such that the inconvenience of participants, in terms of extra distance, is minimized while at the same time we aim to fulfil most of their preferences. Preferences reflect the kind of associates that an agent wishes to have. Our goal is to offer a pleasant trip to drivers and passengers in order to give them the motive to use the application again. Finally, we present a way of incentivisation for drivers, in terms of a monetary reward, so that they are willing to offer their cars again in the future. This is crucial as the whole concept of Ridesharing is based on agents who own cars and they feel encouraged to share it with other people.

In our implementation, we start by doing a partitioning of the city graph based on the initial routes of drivers. Afterwards, we form a Hypergraph where vertices correspond to passengers in order to decompose the problem into multiple simpler problems. We use a characteristic function to evaluate potential coalitions. The value of a coalition depends on the extra encumbrance for drivers and the interaction between members. Our purpose is to develop a coalition formation which provides a high overall welfare. Through a greedy algorithm we construct the coalition formation. Lastly, we calculate the whole trip for each coalition. In this way, we compare the extra distance for a driver when serving passengers and we adjust his expenses in order to reward his contribution.

3.1 Feasible Coalitions and Constraints

A fundamental factor when searching for a solution is usually the satisfaction of some constraints. A solution is said to be feasible if all the necessary conditions are met. Otherwise, it is rejected. Furthermore, a solution can be considered as superior when it satisfies more wants than another solutions. In this thesis, we handle two kinds of constraints, known as *hard* and *soft* constraints [29]. Hard constraints express some conditions which must be satisfied in order to accept a solution as feasible. Even one violated hard constraint leads to a non-accepted solution. On the other hand, soft constraints depict some desires that we want to satisfy to the maximum extent possible. Soft constraint do not reject solutions, but according to these constraints a solution can be regarded as for instance, "accepted", "good" or "excellent".

In this work, the solution that we are seeking is a coalition structure, i.e, a set of coalitions where each one represents a car along with its passengers. Therefore, to begin with, we need to define when a coalition, and by extension a coalition structure is feasible, meaning that it fulfils all the necessary functionalities. A coalition represents a car which contains its driver and possibly some passengers. Hence, concerning coalitions, we distinguish the following hard constraint.

1. If $|C| \geq 1 \Rightarrow \exists \quad r_i \in C: \quad r_i \in D \quad and \quad |C| \leq capacity.$
   Where C represents the Coalition and D the set of Drivers.

2. $|C \cap D| = 1.$
   The number of drivers per Coalition is exactly one.

3. If $r_i \in D: \quad P^1 = r_i^s \wedge P^n = r_i^d.$
   Coalition's route starts from driver's origin and finishes at driver's destination.

4. $\forall r_i \in C: P^x = r_i^s \wedge P^y = r_i^d \Rightarrow x < y.$
   For every agent in Coalition, their starting point precedes their destination.

All four conditions above must be satisfied for every coalition. That is, a coalition structure is feasible if it consists of coalitions that satisfy constraints 1-4.

Furthermore, it need to be clear that the coalitions of the final coalition structure are disjoint. In other words, there is no agent who belongs to more than one coalitions.

$$\nexists \quad C_i, C_j \in CS : \quad C_i \cap C_j \neq \emptyset \tag{3.1}$$

$$\sum_{i \in CS} |C_i| \leq N \tag{3.2}$$

A crucial parameter which we take into consideration in this work is the preferences of participants. Those preferences are expressed through soft constraints. Every participant has some attributes, sex, age and employment. In addition, participants convey their own soft constraints about attributes of people they prefer to collaborate. Our goal is to form coalitions in such a way so as to satisfy most agents' desires while we keeping drivers' detours as low as possible. Hence, participants' soft constraints constitute a measure of satisfaction. Below, we present some examples of soft constraints.

- Agent A prefers to cooperate with: Males, College Students
- Agent B prefers to cooperate with: Females, Employees, ages 30-40
- Agent C prefers to cooperate with: ages 18-25
- Agent D does not have any preference

Soft constraints are not obligatory, but the more of which are satisfied, the greater the coalition's value becomes. In our implementation, there is a trade-off between the satisfaction of soft constraints and the average extra distance that drivers have to travel in order to serve the passengers. If we emphasise the satisfaction of soft constraints, the average extra distance is increased. This happens because the members of a coalition are more compatible according to their preferences; however, it is most probably increasingly difficult to match their routes. By contrast, if small detours weigh more than soft constraints during evaluation, this will lead to a reduction of the satisfied soft constraints. We explain this trade-off in section 3.3.2.

3.2 Hypergraph Construction

In a model which contains a massive amount of agents, the number of possible combinations for feasible coalitions is overabundant. A practical solution is the grouping of agents who move in the same areas and thus could potentially cooperate. On the contrary, we reject collaborations between agents who travel in total different areas of the city. That would cause extremely long routes, with negative effect on inconvenience and agents' satisfaction. By grouping the agents, we divide the initial problem into multiple less complex sub-problems in order to avoid exhausting computations.

This grouping is implemented by applying a hypergraph. The vertices correspond to passengers, agents who do not own cars, and hyperedges represent the cars that are provided by drivers. Hence, the number of hyperedges depends on the number of drivers. In fact, for every driver there is a corresponding hyperedge which contains passengers who could be served by her.

3.2.1 Hyperedges

At this point, we examine precisely the way of hyperedge formation. A hyperedge which corresponds to a driver D, is a set of passengers who could collaborate with D due to spatial compatibility. First, we mark for every driver the area where he is available of giving a lift to potential passengers. Agents who intend to move outside this area, cannot collaborate with the corresponding driver, because that would force the driver to make long detours. Contrariwise, for agents who want to travel inside the area, the corresponding driver constitutes an option.

Each agent of the model comes with the information about his starting point and destination. In order to form the hyperedge which corresponds to each driver, we start by finding the route that she would follow in case that she was alone, without any extra passenger. We apply Dijkstra's algorithm (2.1.3) for the origin and destination of the driver in order to find her initial, optimal (for her) route. Subsequently, we indicate a circle around this route. This is the region where a driver can give a lift to a passenger, as explained in the next section 3.2.2. In Figure 3.1(a) the red line represents driver's initial path, while the red circle is the corresponding region. In sub-section 3.2.2, we define the limits of a hyperedge and the way that passengers are inserted into it.

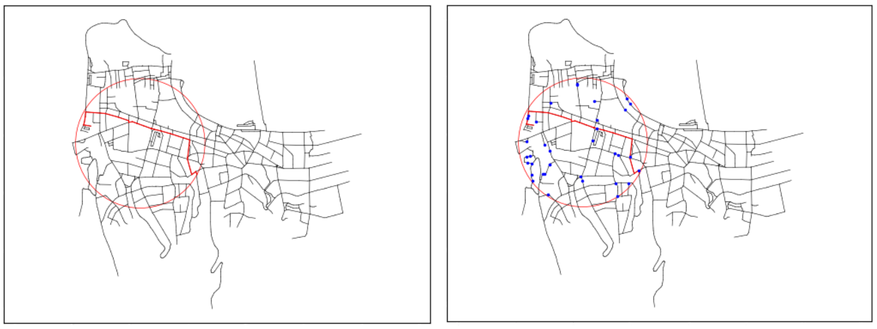

(a) Driver's Area (b) Points of potential Passengers

Figure 3.1: Hyperedge Formation

Once this procedure is completed, we know the region for every driver and we are ready to continue with the insertion of passengers into hyperedges. As we mentioned, a passenger belongs to a hyperedge if she moves inside the region of the corresponding driver. That means that both the starting point and destination of passenger, must be contained in region. This is the condition in order for a passenger to constitute a vertex in the driver's hyperedge. In other words, the attribute that all vertices (passengers) of a hyperedge have in common, is that they can be served by the same driver. Below, we list the conditions which must be satisfied.

3.2.2 Conditions for Hyperedge Formation

Each hyperedge has a corresponding circle equation[1],

$$(x-h)^2 + (y-k)^2 = r^2 \tag{3.3}$$

where (h, k) constitute the center coordinates of the circle and r represents the radius. A point (x, y) belongs to the circle iff it satisfies equation 3.3.

A passenger p_i belongs to a hyperedge H iff both her starting point, with coordinates (x_i^s, y_i^s), and destination, with coordinates, (x_i^f, y_i^f) are located inside hyperedge's circle.

[1]In future work, it would be interesting to replace the circle with an envelope (or "time-buffer") around the preferred driver's route, constructed taking into account the traffic burden of surrounding roads and other driver preferences.

- *condition 1:* $(x_i^s - h)^2 + (y_i^s - k)^2 \leq r^2$ (3.4)

- *condition 2:* $(x_i^f - h)^2 + (y_i^f - k)^2 \leq r^2$ (3.5)

If (condition 1 && condition 2) $\Rightarrow p_i \in H$

The condition above is calculated for every passenger and for every hyperedge of the hypergraph. At the end of the procedure, we have a picture of the passengers who claim a seat at every car. As we can perceive, some routes of drivers can be comparable. That means that those drivers move in similar regions. Therefore, a passenger may belongs to one or multiple hyperedges. However, in some cases a passenger is not compatible with any driver's route, then she becomes a singleton and she needs to use the public transportation. On the other hand, a driver may have a big set of potential passengers or in a less desirable scenario, an empty set. That depends on the distance of his initial route. Long routes lead to larger areas where it is more likely to find agents for matching. By contrast, narrow areas usually contain a small number of agents or no agents at all. Figure 3.1(b) depicts with blue dots the points of passengers, origin and destination, who plan to move inside a specific region. Those passengers are candidates for a seat in the corresponding car.

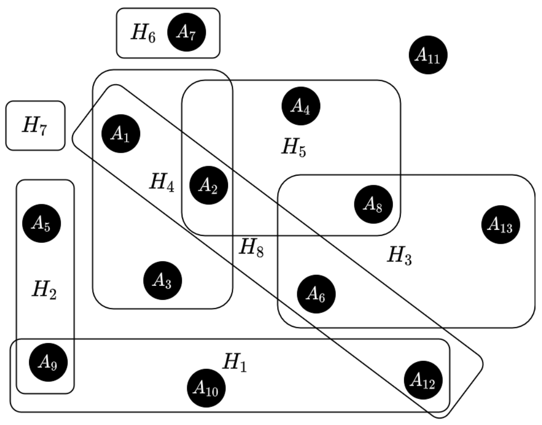


Figure 3.2: Hypergraph for Ridesharing

In Figure 3.2, a simple example of a hypergraph is presented. Vertices correspond to passengers and hyperedges constitute car options. Drivers are not directly part of the hypergraph. They are not presented by vertices as their contribution is expressed through hyperedges. The example above delivers the complete information about possible distribution of passengers to vehicles. For instance, agent A_2 has three different options. Given that she belongs to hyperedges H_4, H_5 and H_8, she could get a lift by drivers D_4, D_5 and D_8 respectively. For agent A_{10} there is only one option, driver D_1. Agent A_{11} was not matched to any driver so he perforce to become a singleton. In a similar way, we notice that driver D_2 can give a lift to agents A_5 and A_9, driver D_3 is available for A_6, A_8 and A_{13} while driver D_7 has not any compatible passenger.

Until this point, the hypergraph has been formed and all the needed information is available in order to start distributing passengers to vehicles efficiently. Agents who became singletons due to incompatibility, either drivers or passengers, will stay out of the process. Those agents are perceived to cause a high increase of the average extra travel distance. We want to avoid this, because drivers are not professionals, but are simple citizens willing to share their car whenever this is feasible. Subsequently, for the agents who constitute the hypergraph, we will then search for an effective coalition structure which covers an acceptable percentage of their preferences while at the same time it does not increase significantly the extra distance for drivers. As we know, for every passenger there is one or multiple options. Our purpose is to register each passenger to a suitable coalition (vehicle). This procedure is explained in detail in the next section.

3.3 Evaluation of the Coalitional Values

In this section, we present the first step for the computation of the final coalition formation structure. Each coalition corresponds to a vehicle, the owner of a vehicle (driver) plans to move from a specific point of the city (origin) to another (destination). Detours are almost inevitable when drivers undertake to give a lift to passengers. However, we want to keep those detours as short as possible in order to motivate drivers to participate. Hence, the most important parameter when forming the coalition structure is to match drivers with passengers so they will not diverge from their path significantly. We start by assigning to passengers a corresponding value for every hyperedge where they belong. This value is basically a measure of compatibility between the potential passenger and the driver of the vehicle. Therefore, a passenger who belongs to k hyperedges will have k different values for every corresponding driver.

The overall value of a coalition results from the individual values of its members plus the interactions between them. That means, the satisfaction of their soft constraints (preferences) when they co-exist in the same coalition (car). A high value for a coalition is translated as a set of agents who cooperate smoothly, without enlarging driver's route excessively. In addition, when the percentage of satisfied preferences is substantial, the coalition's welfare is improved with a positive effect on value. Our end purpose is to apply an algorithm which discovers a feasible coalition structure CS with a total value which approaches the optimal.

$$CS^* = \underset{CS \in \mathtt{CS}}{\arg\max} \sum_{C \in CS} v(C)$$

where $\mathtt{CS}$ denotes the set of all possible coalition structure that can occur given the set of agents.

3.3.1 Evaluation of Agents through Spatial and Preferences-Related Compatibility

A passenger's value with respect to a specific driver is the product of a combination between spatial compatibility and soft constraints satisfaction. Specifically, the most important factor when evaluating a passenger is whether her direction is similar to the driver's direction. Both agents move at the same region, we guaranteed that in the previous section 3.2; however, we have not determined yet if they travel in the same direction. Obviously, this is a crucial parameter for the collaboration between passenger and driver. Additionally, we consider the distance of passenger's starting point and destination from the initial route of driver. For example, a scenario is that passenger's route is a sub-route of driver's. In this case, the presence of passenger does not affect the driver and that increases her value. By contrast, when a driver has to diverge a lot in order to pick up and drop off a passenger that has a negative effect on passenger's value. Finally, we take into consideration whether passenger satisfies all or some of driver's soft constraints, and vice versa. High percentage of satisfied preferences leads to a higher welfare. To conclude, two major factors influence the passenger's value.

1. *Geographical Information.* We understand that a set of agents who consist a coalitions should have similar directions. Otherwise, routes would be tiresome and expensive. Hence, we try to combine agents with comparable routes.

2. *Preferences.* As mentioned in soft constraints 3.1, a coalition which satisfies most of its members' desires, is said to be a high quality coalition. Hence, we need to consider carefully the expressed preferences.

Concerning geographical information, we distinguish two parameters. Primarily, the deviation of directions and second the distance between driver's route and passenger's points.

3.3.1.1 Deviation from Driver's Direction

The first parameter that we calculate is the deviation between the passenger's direction and the driver's direction. Given the two points, origin (x_s, y_s) and destination (x_f, y_f), we calculated the direction in degrees according to Figure 3.3. It is defined to be the angle between the line defined by these two points and the horizontal axis. The equation which provides the slope of direction is 3.6 below.

$$\theta = \frac{arctan2\,(y_f - y_s, x_f - x_s)}{\pi} \cdot 180^\circ \tag{3.6}$$

This slope is calculated for every agent. We denote as θ_d and θ_p the angles of driver and passenger respectively. At the next step, we subtract to get the difference, $\Delta\theta = |\theta_d - \theta_p|$. We keep the smallest difference either clockwise or vice versa. The result lies within a range of 0° to 180°. For $\Delta\theta = 0^\circ$ there is a perfect match while for $\Delta\theta = 180^\circ$ their motion is totally opposite.

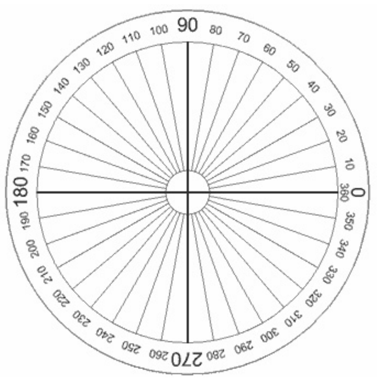


Figure 3.3: Direction in degrees

Given the calculated angle of deviation between directions, $\Delta\theta$, we are going to quantify this value and assign it to a parameter. This parameter expresses the similarity of directions for the pair driver-passenger and it is denoted as `Dir`.

First, we normalize $\Delta\theta$ within a range 0 to 10. Ten corresponds to the finest scenario of $\Delta\theta = 0°$ where both driver and passenger move in exactly the same direction. By contrast, zero corresponds to $\Delta\theta = 180°$. This is the worst case, it is observed when they more in exactly the opposite direction. The normalization equation 3.7 is presented below,

$$\texttt{Dir} = \frac{180 - \Delta\theta}{180} \cdot 10 \tag{3.7}$$

The parameter `Dir` is inversely proportional to $\Delta\theta$. We distinguish four regions depending on its value.

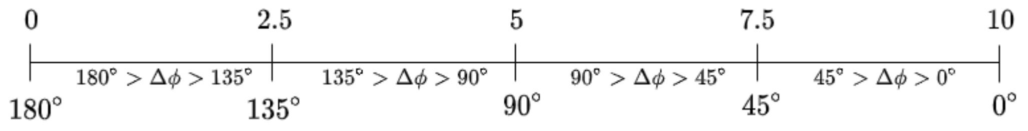


Figure 3.4: Range of value for `Dir`

As an example, let us consider a driver D who's direction is $\theta_d = 35°$. According to Figure 3.3, this driver moves in a northwest direction. Another passenger P moves in northeast direction with $\theta_p = 320°$. The deviation between their directions is $\Delta\theta = 75°$ counterclockwise. Hence, according to Figure 3.4, we notice that the parameter `Dir` for passenger P lies in the range of 5 to 7.5.

In this way, we calculate a gauge of direction similarity which is the most important factor in forming coalitions. This parameter is calculated for every potential passenger and the corresponding driver. For passengers who claim a seat in a vehicle, the ones with the higher `Dir` values are more likely to become members of the corresponding coalition. However, this is not the only factor that affects a passenger's value. We also take into consideration the diversion of the driver in order to serve a potential passenger.

3.3.1.2 Distance from Driver's Route

The second parameter concerning geographical information is the distance between passenger' points and the driver's initial route. In other words, we are interested on how much does a driver need to diverge in order to pick up and drop off a potential passenger. Small detours or no detours at all, affect passenger's value positively. On the other hand, if origin and destination of a passenger are far away from driver's route, that leads to enlarged trips with negative effect.

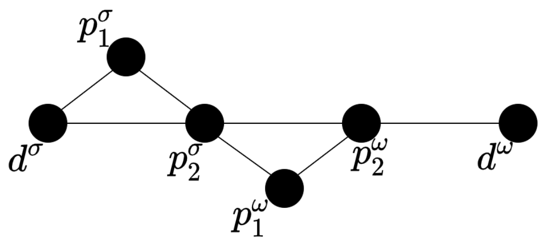


Figure 3.5: Example of Distance

The example of Figure 3.5 presents a driver D who starts from point d^σ and moves to point d^ω. There are two passengers who can get a lift from this driver. For the first passenger, P_1, her origin is the point p_1^σ and her destination is p_1^ω. The second passenger, P_2, wants to move from p_2^σ to p_2^ω. As we notice, the driver does not have to diverge at all in order to serve passenger P_2 because her points belong to the driver's route. By contrast, for passenger P_1, driver will need to make a detour and the final distance of the whole trip will be increased. Hence, although both passengers can be served by the same driver, P_2 appears to have an extra advantage because her starting point and destination are more compatible with driver's initial route.

The parameter which represents the distance as described above is denoted as `Dist`. In order to assign its value, first we calculate the two individual distances. The distance from the passenger's origin to the driver's route and the same for the passenger's destination. We keep the minimum observed euclidean distance, meaning the distance of a vertical line which connects a point of interest, starting point or destination of a passenger, with the driver's initial route.

$$S_p = \begin{cases} max & \text{if } Start\ point \in path \\ max - dist(Start\ point, path) & otherwise \end{cases} \tag{3.8}$$

$$D_p = \begin{cases} max & \text{if } Destination \in path \\ max - dist(Destination, path) & otherwise \end{cases} \tag{3.9}$$

Equations 3.8 and 3.9 present the calculation of the parameters for the starting point and the destination respectively for the passenger. If the point belongs to the driver's route, the parameter gets the maximum value which is 10 in order to be in the same range as parameter `Dir`. Otherwise, as the distance is increased, the values of the parameters are decreased. The final, overall factor of distance, `Dist`, is the average of S_p and D_p.

$$\texttt{Dist} = \frac{S_p + D_p}{2} \tag{3.10}$$

As we notice, the factor `Dist` is inversely proportional to the distance from the passenger's points to the driver's initial route.

In conclusion, as explained in sections 3.3.1.1 and 3.3.1.2, two factors influence the values of passengers according to their source and destination. First, the evaluation ensures that they intend to move in a similar direction as the driver, and secondly, passengers who cause shorter detours have priority over those who enlarge the trip.

3.3.1.3 Soft Constraints Satisfaction (Preferences)

The third factor that we take under consideration when calculating values for passengers is the satisfaction of soft constraints, i.e., the preferences of participants about other people. We seek to form coalitions which will contain members who are compatible in terms of personality. In previous section, we covered the requirements about spatial compatibility. We now attempt to evaluate the compatibility of agents' personality-related as we aim to offer a pleasant trip to participants. Hence, the value of a passenger is influenced by the preferences' accordance between her and the driver. Every agent can express zero to three preferences about other individuals. Those preferences refer to sex, age and status, such as college student, employee etc. This information is selected because at least in our understanding it does not refer to sensitive personal data and it does not violate agent's privacy. For a pair passenger-driver, passenger's value is increased for every satisfied soft constraint. Let's consider the Example 1 below,

Example 1
A driver D has the following features: female, college student, 23 y.o.
and the following soft constraints: female, college students
A potential passenger of D is agent P with features: male, college student, 22 y.o.
and soft constraints: college students, 18-24

As we notice, both driver and passenger have expressed two preferences. The driver's attributes meet both soft constraints of the passenger. On the other hand, passenger satisfies one of two from the driver's preferences. That is to say, three out of four soft constraints are satisfied in total.

Finally, the third factor corresponds to a rate of satisfied soft constraints when the passenger collaborates with a specific driver. We denote as k the total number of expressed preferences and as n the final number of satisfied preferences. Hence, n is a subset of k. In the Example 1 above, $k = 4$ and $n = 3$. The result of the third factor is given by the division n/k. We multiply the result by ten in order to normalise its value within the range from 0 to 10. The parameter SCS refers to Soft Constraint Satisfaction.

$$\texttt{SCS} = \frac{n}{k} \cdot 10 \tag{3.11}$$

So far, we have quantified all three factors that we need in order to assign the final value to the passenger. We explained each factor's significance and contribution. In the next section, we cover how we determine their weights for the final evaluation.

3.3.2 Final Value of a Passenger with respect to a Driver

At this point, we discuss the assignment of a passenger's value with respect to a specific driver. As mentioned in section 3.3, a passenger has multiple different values depending on each potential driver. In other words, a passenger who belongs to N hyperedges is evaluated for each one of them. This holds because the value is a product of comparison between passenger's and driver's attributes, i.e, spatial information and preferences. Hence, passenger's values differ for every driver. We have analysed the three factors that contribute to evaluation, `Dir`, `Dist` and `SCS`. Now, we will explain how we control the weight of contribution for each factor.

Let us consider an agent A who belongs to hyperedge H_i, corresponding to driver D_i, and to hyperedge H_j which corresponds to driver D_j. We denote as $v_{D_i}(A)$ the value of agent A w.r.t. driver D_i. Similarly, $v_{D_j}(A)$ represents the value of agent A w.r.t. driver D_j. As might be expected, the higher value corresponds to the better option. For example, if $v_{D_j}(A) > v_{D_i}(A)$ then it is preferable to match agent A with D_j rather that with D_i. Essentially, the value of a passenger represents the weight of her interaction (edge) with the corresponding driver.

Subsequently, we explain the equation which provides the final value. Every factor is accompanied by a parameter which reflects its importance and contribution to evaluation. We regard the most important factor to be the direction, `Dir`. Intuitively, we understand that agents who constitute a coalition should all move in a similar direction. Detours are probably inevitable; however, the opposite motion for a driver is not desirable, and thus we prefer to avoid it. The parameter which expresses the influence of direction is denoted as α.

During hypergraph construction 3.2, we grouped the agents according to the regions in which they move. At this point, we distinguish the ones who are closer to a driver's route and would thus cause the minimum increase for the whole trip. Clearly, those passengers have a priority in entering the coalition. The significance of the factor `Dist` is expressed via a parameter β.

Finally, we pursue the satisfaction of the most expressed preferences from both passenger and driver. Hence, via a parameter which is denoted as γ, we control the importance of the third factor, `SCS`. Equation 3.12 provides the final value for a passenger A w.r.t. a specific driver D. Of course, the factors `Dir`, `Dist` and `SCS` differ depending on the driver.

$$v_D(A) = \alpha \cdot \texttt{Dir} + \beta \cdot \texttt{Dist} + \gamma \cdot \texttt{SCS} \tag{3.12}$$

The values of the parameters α, β and γ are defined from the operator of the system in order to determine the weight of each factor. Generally, we assign a much higher value to parameter α to ensure that the members of a coalition move in similar directions. Parameters β and γ correspond to comparable values as they are considered of equivalent importance. Later, in our experiments we are going to examine the trade off between preferences and the extra distance for drivers. When α and β are greater in comparison to γ, we expect that the itineraries will be more compact and the average extra distance for drivers will remain low. However, in this case, the preferences become of secondary importance. By contrast, if we emphasise on soft constraints satisfaction by increasing the parameter γ, more preferences will be covered while the average extra distance for drivers will increase.

Now, let us consider Examples 2 and 3 below w.r.t. the hypergraph of Figure 3.2.

Example 2
Passenger A_2 belongs to three hyperedges, H_4, H_5 and H_8. There is a corresponding value for each one—e.g., $v_{D_4}(A_2) = 6.819$, $v_{D_5}(A_2) = 10.086$, $v_{D_8}(A_2) = 8.626$.
We conclude that for agent A_2 the best scenario is to join coalition 5. Her second best alternative is coalition 8 and finally coalition 4.

Example 3
Driver D_3 has three potential passengers in his hyperedge, A_6, A_8 and A_{13}.
Assume that their values w.r.t. driver D_3 are the following,
$v_{D_3}(A_6) = 8.626$, $v_{D_3}(A_8) = 4.042$, $v_{D_3}(A_{13}) = 8.880$. For the driver of coalition 3 the most compatible passenger is A_{13}, followed by A_6 and A_8.

As we mentioned earlier, the value is actually the weight of the interaction between a driver D and a passenger A, w_{DA}. In the next sections, we denote it as $v_D(A)$ for conciseness. We demonstrate this in the example of Figure 3.6.

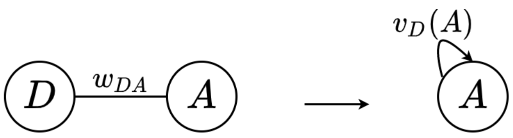


Figure 3.6: Value of an Agent

3.3.3 Value of a Coalition

Having the value of each member from a coalition at hand, we can now assign the coalition's value. In order to calculate the overall value, we sum all the values of the coalition's members, and in addition we include the weights of the interactions between them. A coalition represents a vehicle with members a driver and some passengers who got a seat in this car. The driver is necessary as she owns the car. Hard constraints 1 and 2 must be met *(page 22)*.

In other words, a coalition is a set of agents who co-exist in the same car and they interact with each other. This set can be presented as a complete graph (Definition 3) where every vertex corresponds to an agent and every edge has a weight which reflects the quality of the interaction. If the pair is driver-passenger, the weight equals the value of this passenger as we explained in section 3.3.2. Otherwise, for a pair passenger-passenger the weight represents a gauge of compatibility calculated by soft constraint satisfaction when these two agents collaborate, i.e., the weight is not influenced by geographical information, just by their satisfied preferences. Figure 3.7 shows a coalition C where its driver is denoted as D. Provided that the graph is complete, the driver is connected to any passenger. However, we do not include the driver in the illustration as her contribution is depicted through the passengers' individual driver-related values as shown in Figure 3.6. In addition, for each pair of passengers A_i and A_j there is a corresponding weight w_{ij}.

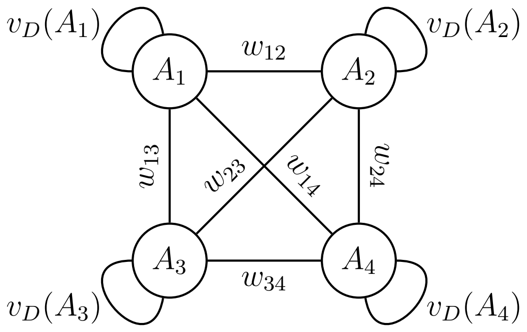


Figure 3.7: A Coalition depicted as a Complete Graph

The weight between two passengers is extracted in the same way as we covered in section 3.3.1.3, where we calculated the factor of preferences between a driver and a passenger. The only difference now is that both agents are passengers. A coalition's overall value is the sum of all individual values, plus the sum of weights between passengers. The respective equation is shown below,

$$v(C) = \sum_{A \in C} v_D(A) + \sum_{\substack{i,j \in C \\ i \neq j}} w(i,j) \tag{3.13}$$

where D is the driver of coalition C

We need to mention that Equation 3.13 corresponds to a superadditive function [30] as it satisfies the inequality,

$$v(A \cap B) \geq v(A) + v(B) \tag{3.14}$$

In our case, inequality 3.14 shows that when we add a new passenger to a coalition, a coalition's value can only be increased. That holds because an extra passenger corresponds to multiple new edges, which leads to possible fulfilment for more preferences. Hence, we aim to cover most of the seats for every coalition.

3.4 Determination of the Optimal Coalition

Each coalition corresponds to a vehicle which belongs to a specific driver. We assume that drivers who participate in Ridesharing, have simple and ordinary cars where the capacity is standard; apart from the driver, a car may contain four more passengers. In Section 3.2, we discussed how passengers are inserted into hyperedges. A hyperedge indicates a set of potential passengers for a specific car. However, this set may contain an amount of agents which exceeds the capacity of the car. Hence, some of those agents will stay out of coalition. In that case, among the whole set of passengers (N), we search for a subset that contains four passengers and has the optimal value.

$$C_{opt} \subseteq N \tag{3.15}$$

$$\text{where } |C_{opt}| = 4 \text{ and } v(C_{opt}) \geq v(C),\ \forall\ C \subseteq N$$

As we discussed in section 3.3.3, finding the optimal combination of four passenger that leads to the highest value is a complicated task. It is not enough to keep the four passengers who possess the top values. We need to take into consideration the weights of their interaction as well, thus we have to be aware of the value for every possible coalition and finally keep the one which is computed to be the optimal.

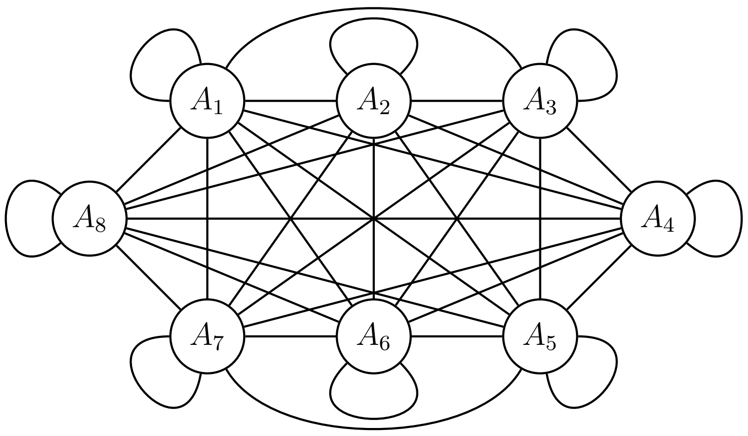


Figure 3.8: Set of Agents - Hyperedge

The Figure 3.8 represents a set of agents who constitute a hyperedge and they are connected through a complete weighted graph (weights are not depicted for a clearer illustration). They are all candidates for a seat at the same car. There are eight passengers who claim a position but only four of them can be accepted due to capacity constraint. As we defined coalition's value in Equation 3.13, we notice that the value of a sub-graph (Definition 5) is given by the sum of the weights from the included edges. For example, the subset $C = \{A_1, A_2, A_3\}$ has the value $v(C)$ where,

$$v(C) = v_D(A_1) + v_D(A_2) + v_D(A_3) + w_{12} + w_{13} + w_{23}$$

The challenge is to find the four passengers who obtain the optimal value when co-operating. A simple solution would be to apply a *brute-force*, also known as *exhaustive*, search in order to enumerate every possible coalition, then calculate its value and finally keep the most profitable one. This solution guarantees optimality but it comes with exponential complexity, thus it would cause unacceptable delay. An efficient solution to this problem is presented by Maxim Binshtok et al. in [31]. Specifically, they propose a Branch and Bound method in order to avoid unnecessary computations and they guarantee to determine the optimal subset fast and reliably.

We are now going to explain the algorithm's function in detail. The implementation is based on Trees (Definition 6). More specifically, the search starts from an empty set, by expansion we gradually evaluate the possible combinations when we include or reject one agent at a time. This process continues until we reach the optimal solution. Each node of the Tree corresponds to a state where some agents are included in optimal coalition while some others have be rejected. Given these two sets of agents, *accepted* and *rejected*, we calculate two parameters. First, the value of the coalition which contains the accepted agents, and secondly an *Upper bound* for this state. The Upper bound is an estimation about the maximum value that could be found from this point onwards, with respect to the current sets of accepted and rejected agents.

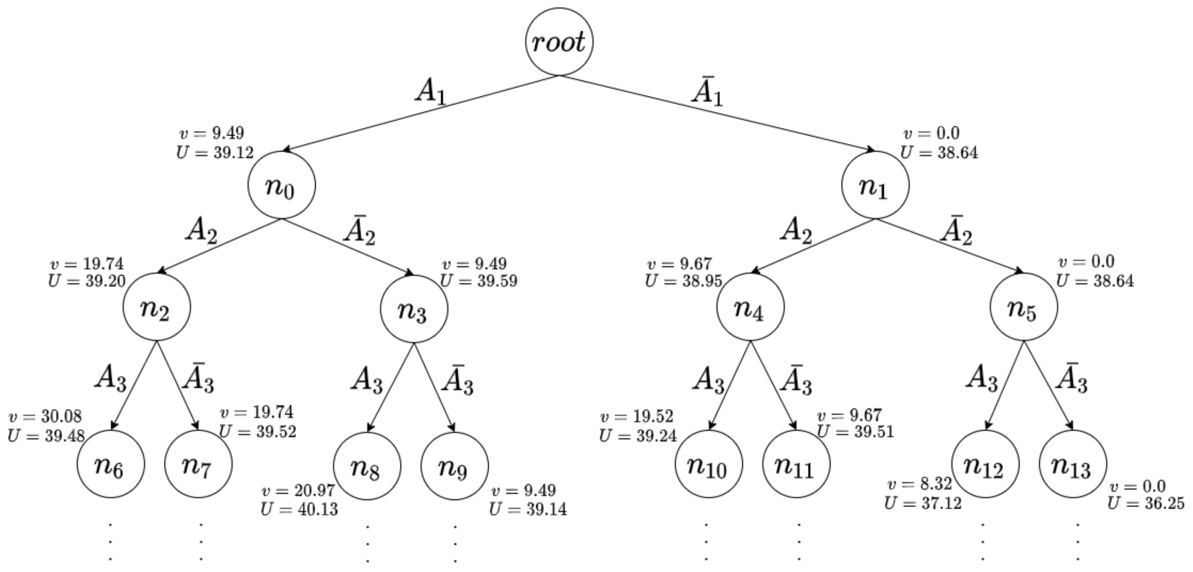


Figure 3.9: Example of Tree

The example of Figure 3.9 depicts a Tree until the third level of expansion. Node n_0 corresponds to a state where the first agent, denoted as A_1, is considered to be member of the optimal solution. By contrast, node n_1 expresses the opposite, agent A_1 is definitely out of the optimal coalition. Similarly, we deduce for every node the agents who have already been selected and the ones who were left out. For example, node n_8 indicates a scenario where agents A_1 and A_3 are members of the solution but A_2 is not. In addition, each node contains its value and the Upper bound.

The greatest advantage of this method is the ability of applying pruning. That means, we expand as few nodes as possible in order to reduce the search space significantly and thus improve the execution time. To do so, we keep the highest value found in a parameter called *Best*. When a new node is produced, if the number of included agents equals the capacity of the car then this node is a leaf and it can not be expanded further. If the capacity is not reached yet but the Upper bound of a node is lower that the Best value, then this node is no worthy for expansion because it will not lead to a better solution than the one which has already been found. Hence, this node gets blocked so there will be no waste of time on it. By doing so, we prune the Tree and we avoid unnecessary computations.

A very important factor for the performance of the algorithm is sorting the agents in such a way so that the optimal subset gets found as soon as possible. That would lead to early deep pruning and a big amount of non-useful computations could be avoided. Therefore, before starting the construction of the Tree, we do an evaluation of the agents so to determine the priority queue. We have the complete weighted graph of those agents at hand; hence, we can calculate the Shapley value (2.1.7) for every agent. Those who have higher Shapley values are considered to be significant candidates, and they are more likely to be part of the final subset. However, this is just an assumption for the initial sorting which may help the algorithm to find the solution faster.

To conclude this section, we adapted the *Branch and Bound* algorithm of [31] in our implementation. This algorithm takes as input a set S of candidates for a coalition, where $|S| > capacity$, and returns the optimal subset of agents who can achieve the maximum value. It uses pruning in order to restrict computations and reduce the execution time as much as possible. Algorithm 2 presents a pseudocode of the Branch and Bound algorithm. In next section, we will explain how the main algorithm uses the Branch and Bound in order to compute the final coalition structure.

Algorithm 2 BRANCH_N_BOUND(*candidates*)

```
1: sort candidates by their Shapley value;
2: Node root $\{\emptyset, \emptyset, 0, \infty\}$;
3: list Q;
4: Q.push(root);
5: Best = $\emptyset$;
6: while ( !Q.empty() ) do
7:    $Nd$ = Pop(Q);
8:    NODE_PARAMETERS($Nd$);
9:    if ( (!$Nd$.blocked) && ($Nd$.Upper_Bound $\geq$ Best.value) ) then
10:       if ($Nd$.value $>$ Best.value) then
11:          Best = $Nd$;
12:       end if
13:       $I$ = next agent of candidates;
14:       candidates.remove($I$);
15:       $Nd_1$ = extension of $Nd$ with agent $I$;
16:       $Nd_2$ = extension of $Nd$ without agent $I$;
17:       Q.insert($N_1$, $N_2$);
18:    end if
19: end while
20: return Best;
```

Algorithm 3 NODE_PARAMETERS(Nd)

```
1: list members = $Nd$.accepted;
2: list rejected = $Nd$.rejected;
3: $Nd$.value = calculate_value(members);
4: $Nd$.Upper_Bound = upperEstimation(members, rejected);
5: if (members.size() == capacity) then
6:    $Nd$.blocked = true;
7: end if
8: return $Nd$;
```

3.5 *PASR-CF* Algorithm

In this section, the *PASR-CF (Preferences Aware Social Ridesharing - Coalition Formation)* algorithm, the main algorithm of our approach is presented. So far, we have created the hypergraph which depicts for every agent the set of cars which can give her a lift. We have done the evaluation and we have quantified for each agent how much compatible she is for every possible vehicle. Furthermore, we implemented the algorithm presented in [31], which gets a set of candidates for a vehicle and guarantees to return the optimal subset of agents to cooperate. At this point, we will combine the above in order to form the final coalition structure. The main algorithm of our approach operates in a greedy manner since at each iteration, it picks the current optimal option.

Viet Dung Dang et al. in [32] propose a greedy algorithm for coalition formation in Multi-Sensor networks. In this work, there is overlapping among coalitions. What that means is that each agent may belong to multiple coalitions. However, in our work this cannot be the case because agents correspond to humans and coalitions represent vehicles. Hence, an agent belongs to one coalition at most. Nevertheless, before coalition formation, there exist overlaps in the Hypergraph. Every agent is probably contained in multiple hyperedges. This correlation between our work and [32] inspired the creation of the *PASR-CF* algorithm which was influenced by the greedy algorithm presented in [32].

After the creation of the Hypergraph, every agent claims a position among a set of vehicles. For each car, there is a corresponding value which expresses the compatibility between the potential passenger and the car's driver. Initially, we match each passenger to her optimal option, meaning the vehicle where she appears to have her highest value. Everyone becomes a candidate for her best option. Hence, each car has k potential passengers who claim a seat. If $k \leq capacity$ then everyone joins the coalition. Otherwise, among those k passengers only the four more capable will be accepted. That means, the four passengers plus the driver who lead to the highest value. For the rest of them, we are going to search for their alternatives in next iteration. The coalitions which are completely formed will no longer be part of the procedure and they do not constitute an option for the remaining passengers. In the next step, the same process is repeated with the remaining agents and coalitions. The algorithm terminates when there is no more available cars or there is none agent who still has any option. A pseudocode of our *PASR-CF* algorithm is presented below in Algorithm 4.

Algorithm 4 PASR-CF(*Hypergraph, agents*)

```
1: condition = true;
2: while (condition) do
3:     for a in agents do
4:         Opt = the most suitable coalition for a;
5:         insert a into Opt temporarily;
6:     end for
7:     for C in coalitions do
8:         if ($|C| \geq capacity$) then
9:             if ($|C| > capacity$) then
10:                branch_n_bound(C);
11:            end if
12:            C.closed = true;
13:            agents.remove(C.members);
14:        end if
15:    end for
16:    if (at least one agent with options left) then
17:        conditions = true;
18:    else
19:        conditions = false;
20:    end if
21: end while
22: return coalition structure;
```

When the *PASR-CF* algorithm is finished, the final coalition structure has been formed. The distribution of agents to cars is completed and now we have the information about each car. Each coalition corresponds to either a vehicle or to a singleton passenger. A vehicle contains its driver and probably some more passengers. A singleton passenger is someone who was not matched to any car due to spatial incompatibility or because other passengers were more convenient that him. For each agent, we have the information about her origin and destination. Hence, we know every point that the final itinerary has to go through in order to pick up and drop off everyone. Subsequently, we are going to extract the final optimal route for the each coalition.

3.6 Route Extraction

Given the distribution of agents into cars, i.e., the final coalition structure, we are now going to determine for each car the final route which has to be followed in order for everyone to go to their destination. Subsequently, we can compare the final trip to each driver's initial route in order to extract the results about the average extra distance for drivers when sharing their vehicles. In addition, based on the extra distance for a driver, we adjust her expenses with the aim of rewarding her contribution.

The route has to be feasible, meaning that constraints 3 and 4 *(page 22)* must be satisfied. The whole trip starts from driver's starting point and it ends up to his destination. It is obvious that for each member, the itinerary needs to go through her picking up point before moving to her destination. In other words, the origin must precede the destination in order to consider the route as feasible. Taking these restrictions into consideration, we search for the optimal route. This problem is similar to the Travelling Salesman Problem [12], with some extra precedence constraints, and the distinction that the final node is not the same with the starting node.

For the determination of the final trip, we developed the algorithm *CFRE* (Coalition's Final Route Extraction) which is similar to the algorithm of section 3.4, since they both use Branch and Bound technique. In this section, we search for an efficient and feasible permutation of stops, such that the final trip is minimized. Of course, there are multiple different routes that can be followed even when restrictions are satisfied. As we know, each coalition may contain zero to four passengers apart from driver. When a driver forms a singleton or when there is only one more passenger, the solution to these scenarios is clear.

```
if (no passengers):
  (driver's Start) → (driver's Dest)
if (one passenger):
  (driver's Start) → (passenger's Start) → (passenger's Dest) → (driver's Dest)
```

By contrast, when more passengers are contained, i.e., two or more, there are various alternative routes. We search for the one with the lowest cost in terms of kilometers travelled. Let us consider the Example 4 below,

Example 4

A coalition C contains its driver and three more passengers. We define the problem as,

1. $P(0) = (driver's)_{start}$, $P(last) = (driver's)_{destination}$
2. $s_1 < d_1$
3. $s_2 < d_2$
4. $s_3 < d_3$

That means that path's origin and destination are given. In addition, there are three restrictions. Point s_1 has to be visited before d_1, s_2 before d_2 and finally s_3 before d_3. Hence, the permutation (start, s_2, s_1, d_2, d_1, s_3, d_3, destination) is feasible, but the permutation (start, s_2, s_1, d_3, d_1, s_3, d_2, destination) is not, because d_3 cannot precedes s_3.

The implementation of the *CFRE* algorithm is based on Search Trees. It starts from the root node which includes the starting point of the trip. By Depth-first search it investigates possible routes and their corresponding costs. Every node of the Tree contains a point and the cost up to this. In addition, children nodes correspond to the next available options from the current node. For example, if the current point is *start* from the Example 4, the next available points are s_1, s_2 and s_3. If we move to s_2, the next feasible options are s_1, d_2, s_3 etc. When a leaf node is reached, i.e., a new path has been found, if the total cost is lower than the current minimum (*minCost*), this means that the new path is shorter than all the already known paths; therefore the *minCost* is updated. Subsequently, during the further expansion of the Tree, if a middle node's cost exceeds the *minCost*, then it is blocked and the search of this path stops there. In other words, the Tree is pruned at this point because this path will definitely lead to a route which has greater cost than the one that we have already found.

The Table 3.1 presents the costs of feasible transitions for a coalition which contains three passengers apart from driver. The cost is calculated via Euclidean distance in terms of kilometers. Dijkstra's algorithm could have been used instead, but that would cause an extra delay in execution time. We consider this approach to be a necessary balance between optimality and practical applicability. For transitions which violate constraint 4, we consider their costs to be infinite. Subsequently, we search for the best possible route given our costs as calculated

$\downarrow i,\ j \rightarrow$	$start$	s_1	s_2	s_3	d_1	d_2	d_3	$destination$
$start$	0	c_{s_1}	c_{s_2}	c_{s_3}	∞	∞	∞	∞
s_1	∞	0	$c_{s_1s_2}$	$c_{s_1s_3}$	$c_{s_1d_1}$	$c_{s_1d_2}$	$c_{s_1d_3}$	∞
s_2	∞	$c_{s_1s_2}$	0	$c_{s_2s_3}$	$c_{s_2d_1}$	$c_{s_2d_2}$	$c_{s_2d_3}$	∞
s_3	∞	$c_{s_1s_3}$	$c_{s_2s_3}$	0	$c_{s_3d_1}$	$c_{s_3d_2}$	$c_{s_3d_3}$	∞
d_1	∞	∞	$c_{s_2d_1}$	$c_{s_3d_1}$	0	$c_{d_1d_2}$	$c_{d_1d_3}$	c_{d_1dest}
d_2	∞	$c_{s_1d_2}$	∞	$c_{s_3d_2}$	$c_{d_1d_2}$	0	$c_{d_2d_3}$	c_{d_2dest}
d_3	∞	$c_{s_1d_3}$	$c_{s_2d_3}$	∞	$c_{d_1d_3}$	$c_{d_2d_3}$	0	c_{d_3dest}
$destination$	∞	∞	∞	∞	∞	∞	∞	0

Table 3.1: Cost Table

The starting point is always known, since it depends on the driver. From the current state, we examine the next options. As mentioned, it is useless moving to a destination point if the corresponding starting point has not been visited yet. Hence, the restriction below holds,

$$d_i \in options,\ iff\ \ s_i \in Path$$

This parameter is used in order to prune the Search Tree during expansion with the aim to avoid exploring alternatives which will not lead to a better solution. At every step, we calculate the cost by adding the individual costs of the intermediate routes. When every point has been visited in a feasible order, the route ends up to the final destination. The overall cost is given by the equation 3.16 below.

$$Total\ Cost = \sum_{\substack{i,j \in Route \\ (i,j) connected}} c(i,j) \tag{3.16}$$

In Figure 3.10, an example of a Search Tree is presented. Only the optimal path is depicted; however, until that point every node has been expanded. This path corresponds to the order according to which the points have to be visited so that the distance and consequently the cost are minimized.

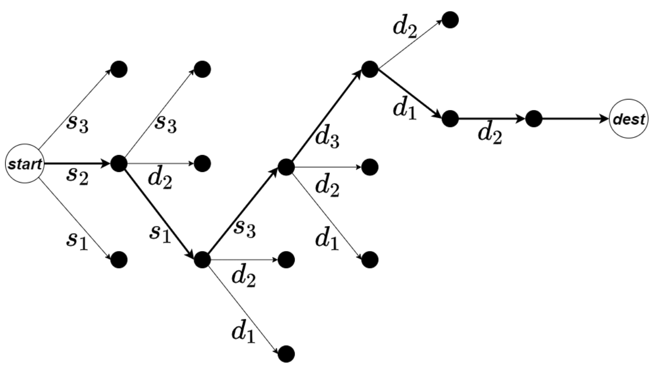


Figure 3.10: Optimal Path

In this example, the points are going to be visited according to the following sequence,

$$start \rightarrow s_2 \rightarrow s_1 \rightarrow s_3 \rightarrow d_3 \rightarrow d_1 \rightarrow d_2 \rightarrow dest$$

Route's Cost is calculated gradually during expansion and it is,

$$Cost = c(start, s_2) + c(s_2, s_1) + c(s_1, s_3) + c(s_3, d_3) + c(d_3, d_1) + c(d_1, d_2) + c(d_2, dest)$$

Algorithm 6 presents a pseudocode of *CFRE*. We apply this algorithm to each coalition in order to determine the final route for the whole trip. Afterwards, we can measure the average extra distance for drivers and reduce their expenses according to their detours.

Algorithm 5 NEW_STATE(p, parentState)

```
1: New State $s$;
2: $s.parent$ = parentState;
3: $s.options$ = options of $p$ according to $costTable$;
4: $s.cost$ = parentState.$cost$ + transition cost to $p$ (euclidean distance);
5: if (destination is reached) then
6:     $s.blocked$ = true;
7:     if ($s.cost \leq minCost$) then
8:         $minCost = s.cost$;
9:         $optimalSolution = N$;
10:    end if
11: end if
12: return $s$;
```

Algorithm 6 CFRE(Coalition Structure)

```
1: for $C$ in Coalition Structure do
2:   if $|C| == 1$ then
3:     $stops$\{driver's origin, driver's destination\};
4:   else if $|C| == 2$ then
5:     $stops$\{driver's origin, $s_1$, $d_1$, driver's destination\};
6:   else
7:     State $root$ = NEW_STATE(driver's origin, null);
8:     $costTable$ = costs of feasible transitions;
9:     Q.insert(root);
10:    $minCost = \infty$;
11:    while (!Q.empty()) do
12:      $N$ = Q.$pop()$;
13:      if ($!N.blocked$ && $N.cost \leq minCost$) then
14:        for $p$ in $N.options$ do
15:          State n = NEW_STATE(p, N);
16:          Q.insert(n);
17:        end for
18:      end if
19:    end while
20:    $stops = optimalSolution.stops$;
21:  end if
22:  return $stops$;
23: end for
```

At this point, we need to mention that our purpose in this part of our work is similar to the Route Computation which is presented by Bistaffa et al. in [33]. However, they use some (anti)monotonic cost functions which do not take into account preferences, they consist of terms which are beyond our work here. Moreover, it is not clear that the (anti)monotonic property holds always in practice. In addition, they used the A* algorithm. This algorithm was tested in our work but without good performance. We attribute this to the non-homogeneity of the various city graphs which make the choice of a heuristic function not obvious. Indeed, we believe that much engineering work is needed in order for the A* algorithm to work properly. Hence, since our experiments verify that our algorithm works well in practice, plus the Route extraction is not our primary concern, we used the method presented above in order to have reliable and quick results.

3.7 Driver's Reward

After the determination of the final itinerary for the whole coalition, it is expected that driver has to go through some detours in order to satisfy every passenger. That means that drivers cover some extra distance which causes an increase in charge. However, drivers' expenses must be fewer or equal to their initial cost, i.e., the cost of their routes when no passengers are included. The reduction is proportional to their contribution. In other words, a larger extra distance for a driver leads to greater deduction of her expenses. This is a significant aspect of the overall problem, since drivers need a motive to participate.

Given the final route of coalition and the initial route of driver, at this point we adjust his payment. The distance is calculated in kilometers. We consider the fuel cost per kilometer to be 0.14€. The overall cost of the route, R_C, is given by,

$$R_C = distance \cdot costPerKilometer \tag{3.17}$$

We mentioned that the reduction of drivers' expenses occurs by their detours. Hence, final cost is inversely proportional to the extra distance. Equation 3.18 gives the final amount that a driver has to pay, D_C. It depends on her initial cost and the deviation between initial and final distance.

$$D_C = \begin{cases} \dfrac{initCost - \left(1 - \dfrac{init_{KM}}{final_{KM}}\right) \cdot initCost}{|C|} & \text{, if } init_{KM} \neq final_{KM} \\ \dfrac{initCost}{|C|} & , otherwise \end{cases} \tag{3.18}$$

The equation presents that when a driver covers a distance which is n times larger than her initial route, her expenses become n times lower. Otherwise, if driver does not need to deviate at all in order to pick up and drop of the rest passengers, the cost is divided equally. Hence, we guarantee that a driver never has to pay a higher price than the one when he acts alone. By contrast, drivers can achieve a significant reduction in their expenses when participating in Ridesharing.

3.8 Cost Sharing

We are now going to distribute the remaining expenses among the passengers of the vehicle. In section 3.7, we mentioned that drivers cover a share of the total cost which is proportional to their contribution. The passengers need to share the rest. However, the cost for every passenger should be fair and proportional to her participation in the trip. Hence, the share for each passenger arises from the distance which is covered by the car when she is inside.

Let us denote as R_C the *Route's cost*, meaning the total cost for the coalition. Driver's share is denoted as D_C, thus the remaining cost for the passengers is the difference,

$$Cost = R_C - D_C \tag{3.19}$$

For every passenger p_i, we first calculate the distance that she travels by the car. We denote that distance as d_i. Subsequently, we compute a payment coefficient, c_i, for each passenger as show in Equation 3.20.

$$c_i = \frac{d_i}{\sum\limits_{j \in P} d_j} \tag{3.20}$$

where P corresponds to the set of passengers.

The share of each passenger is calculated when multiplying her payment coefficient with the overall *Cost* for passengers.

Example 5
Let us suppose that passengers need to share a cost of X€. The distance d_i for each passenger is shown in Table 3.2 below,

Passenger	*Distance*
p_1	$7km$
p_2	$3km$
p_3	$4km$
p_4	$6km$

Table 3.2: Traveled Distances

The sum of all four distances is 20km. Hence, the payment coefficient for each passenger is calculated as show below,

1. $c_1 = \frac{7}{20} = 0.35$
2. $c_2 = \frac{3}{20} = 0.15$
3. $c_3 = \frac{4}{20} = 0.2$
4. $c_4 = \frac{6}{20} = 0.3$

The final prices for those passengers are,

1. p_1 *has to pay* $0.35 \cdot X€$
2. p_2 *has to pay* $0.15 \cdot X€$
3. p_3 *has to pay* $0.2 \cdot X€$
4. p_4 *has to pay* $0.3 \cdot X€$

We make the assumption that participants use digital payments in their transactions, thus the expenses can be automatically split between the members of the coalition.

Chapter 4

Experimental Evaluation

4.1 Creation of City Graphs

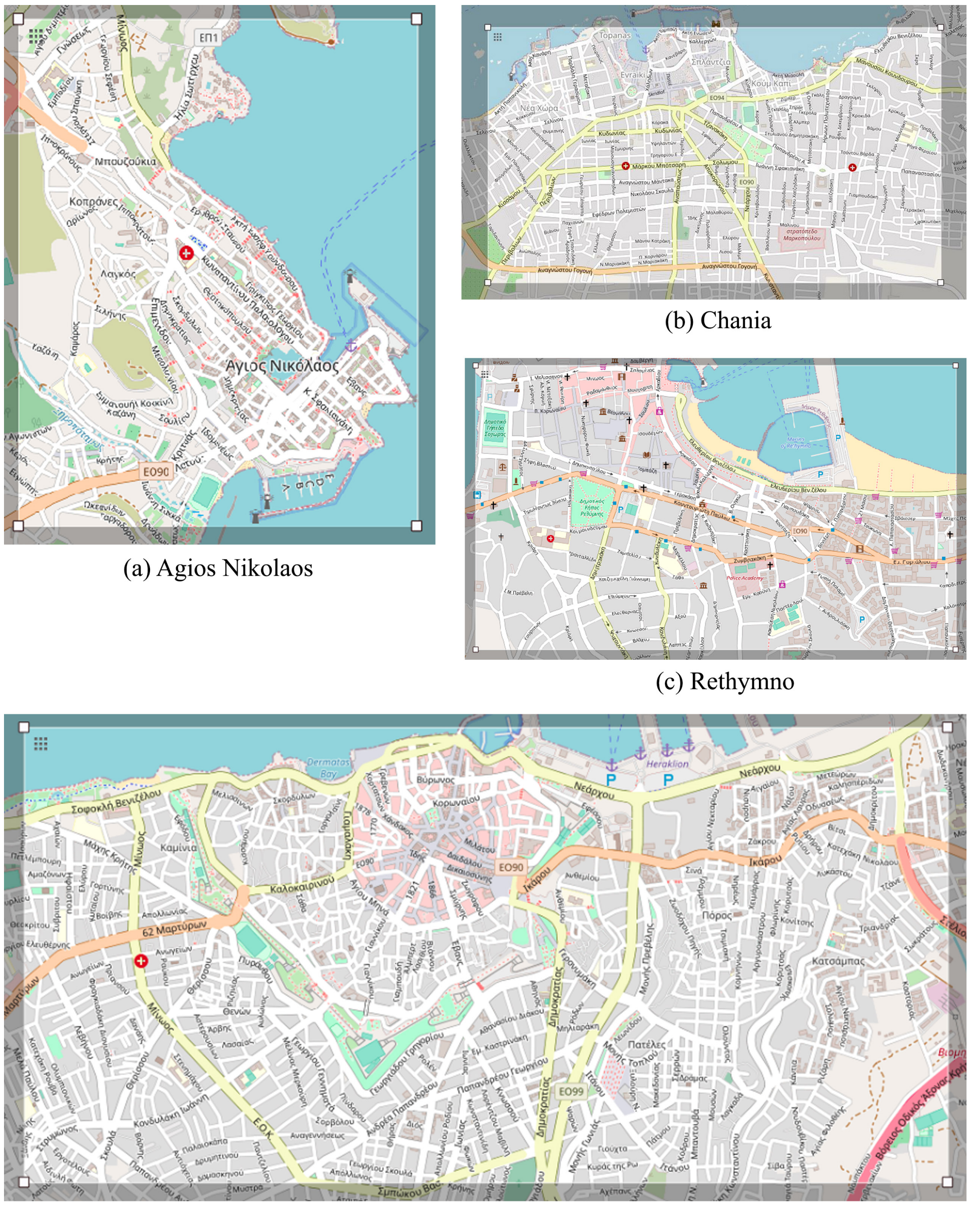
For the implementation and evaluation of our approach, we need the graph of a city where the Ridesharing takes place. This must be a connected graph, i.e., there is at least one path between any two nodes. Graph's vertices correspond to spots of the city, while edges represent the roads which connect them. OpenStreetMap[1] is a free and available map which is created by volunteer users. It includes all the necessary information about any city concerning spots, roads etc. Through OpenStreetMap we can export an *xml* file that contains all the included information of this region.

For the needs of this thesis, we created a program which processes an *xml* file that contains all the information of a city and it produces the corresponding city graph. Hence, we have the ability to extract any graph with various number of vertices and use it for our implementation. Of course, this could be useful for plenty different works where city graphs are used. For example, a research about the optimal placement of chargers for electric vehicles.

In this work, our case studies focus on the four capitals of Crete. Heraklion, Chania, Agios Nikolaos and Rethymno. These are cities that differ considerably in size. Subsequently, the corresponding city graphs contain different number of vertices each. In Figure 4.1, we present the regions where we applied our approach for the Ridesharing

[1]https://www.openstreetmap.org/

problem. The *xml* files from those areas are used in order to construct the corresponding city graphs.



(a) Agios Nikolaos

(b) Chania

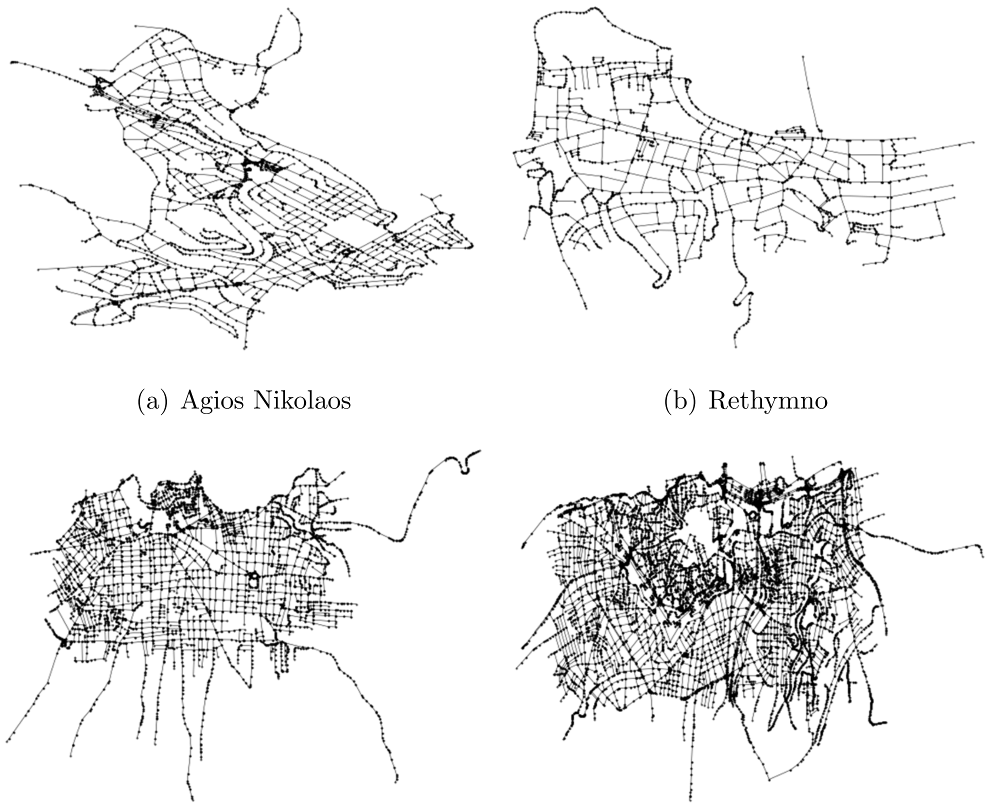
(c) Rethymno

(d) Heraklion

Figure 4.1: Case Study Cities

As we mentioned previously, our program takes an *xml* file as an input which contains information about a city. The generated graph consists of nodes which represent points-intersections of this city. Each of them corresponds to specific coordinates of longitude and latitude. In addition, streets depict sequences of points connected through edges. Figure 4.2 presents the city graphs of Figure 4.1 as they were constructed from our program.

Having those city graphs at hand, we will apply our approach on them. The number of agents will be proportional to the size of each city. Heraklion is the largest city and its corresponding graph includes 9084 vertices. Chania has the second larger extent and its graph contains 3278 vertices. Agios Nikolaos and Rethymno have similar size and their graphs are presented by 2185 and 1321 nodes respectively.



(a) Agios Nikolaos

(b) Rethymno

(c) Chania

(d) Heraklion

Figure 4.2: City Graphs

4.2 Experiments and Results

In this section, we evaluate the performance of our implementation. Our experiments ran on a desktop PC with an AMD Ryzen 7 2700X 3.7 GHz processor and with 16GB of RAM. The first factor to consider is the trade off that was mentioned in section 3.3.2 between preferences satisfaction and extra distance for drivers. We expect that giving emphasis on preferences, would affect the distribution of agents into cars and the average extra distance for drivers would be increased. This is because in order to connect agents who are more compatible in terms of preferences, we need to be more flexible about spatial requirements.

The second factor to evaluate is the percentage of passengers who did not join any coalition and they form singletons. These are the agents who are forced to use public transportation because they were not matched to any seat at any available vehicle. Of course, we wish to keep this percentage as low as possible. Furthermore, we examine what is the average extra distance that a driver has to cover when participating, and the extent to which his expenses can be reduced. Finally, we provide the execution time of our program for each city and for different scenarios with various numbers of agents.

4.2.1 Factors and Weights

In Section 3.3.2, we explained that three factors affect the values of passengers. The first factor is passenger's direction and how much it coincides with driver's direction. A parameter α depicts the weight of this factor. The second one is the distance between a passenger's points and the driver's initial route. Its importance is expressed through the parameter β. Finally, preferences constitute the third factor and the corresponding parameter is γ. Later, we will examine how the final results are affected when we change the parameters α, β and γ. We expect that while γ is increased, the percentage of total satisfaction for passengers will be improved. However, this will lead to an increase in total travelled distance. First, we need to define the way of measurement for preferences satisfaction among a coalition.

1. In the first step, we calculate the percentage of preferences satisfaction for every individual member A of the coalition. We know that, $|C| \leq 5$.

- Each member can express at most three preferences. About co-passengers' sex, age, and employment. Let us denote as $prNum$ the number of expressed preferences, so $prNum = 1, 2$ or 3.
- For each preference p, there is an indicator which expresses the extent to which the co-passengers of A satisfy p. For instance, a preference about age could be satisfied by n co-passengers among k. Hence, for this preference we have,

$$pref_1 = \frac{n}{k} \tag{4.1}$$

- The final percentage of preference satisfaction for agent i is calculated as the average of her individual indicators

$$pR(i) = \frac{\sum_{j=1}^{prNum} pref_j}{prNum} \tag{4.2}$$

  where $pR(i)$ is the preference satisfaction rate for agent i.

2. After the calculation of individual preference rates from the members, we can now extract the overall preference satisfaction rate of the coalition, denoted as CPR, which is the average.

$$\text{CPR} = \frac{\sum_{i=1}^{|C|} pR(i)}{|C|} \tag{4.3}$$

   In other words, let us suppose that a coalition contains k members. If for every preference of anyone, there is nobody who satisfies it, then the CPR is 0%. By contrast, if for any preference of anyone, everybody else satisfies it, then the preference rate is 100%. Both these scenarios are not likely to happen. However, the more successful a formation becomes, the closest to the maximum (100%) CPR would get.

3. The experimental results about preference satisfaction have been produced based on coalitions which contain two or more agents, its driver and one or more passengers. For singletons, there is no interaction between agents, so these are not included in the computation of the total percentage. In addition, we provide results about

the average extra distance that drivers need to cover when they have passengers on board.

At this point, we need to remind the reader of Equation 3.12,

$$v_D(A) = \alpha \cdot \texttt{Dir} + \beta \cdot \texttt{Dist} + \gamma \cdot \texttt{SCS}$$

which produces the final values for passengers. Given that equation, we aim to form a coalition structure which approaches the optimal overall welfare. As we observe, the value depends on parameters α, β and γ. We expect that when those parameters are changing, the final outcome is affected. Hence, we tried four different scenarios concerning these parameters in order to determine which one appears to be more profitable. The values for each scenario are shown below,

For the first scenario, we set $\gamma = 0$, in order to observe the percentage of preferences satisfaction (CPR) when preferences are not considered at all. Subsequently, in the next scenarios we increase the parameter γ, and expect the preferences satisfaction to improve. In the fourth scenario, γ gets a big value in order to examine how much we can increase preferences satisfaction without further enlarging the extra distance for drivers. The average extra distance (AED) in our experiments is the percentage of the increase in driver's distance when she has passengers on board, compared to when she acts alone.

We examine three different cases concerning the percentage of drivers among agents. More specifically, we consider 35%, 50% and 80%. According to ACEA (European Automobile Manufacturers' Association)[1], the number of passenger cars in Greece for 2018 was 5,164,183[2]. Provided that the population of Greece in 2018 was 10,522,246[3], we come to the conclusion that the percentage of vehicles per population was 49%. However, in our implementation the agents from the input data correspond to ages from eighteen and above. Hence, we expect that the real percentage in this case is higher than 49% but lower that 80%, which is an estimation that corresponds to special cases. Cities where the percentage of car owners is higher than the usual. We consider the first scenario of 35%, in order to cover potential cases where the set of drivers among participants constitutes

[1]https://www.acea.auto/

[2]https://www.acea.auto/files/ACEA_Report_Vehicles_in_use-Europe_2019-1.pdf?fbclid=IwAR2Q5N3by47tgImBCwgBV-vXjYN31hhJx1vk-V2rYy1DxQt5Th6KMx_jCsw

[3]https://www.worldometers.info/world-population/greece-population/

the minority. For example, when the Ridesharing is used by a community where most participants are college students. However, the concept of Ridesharing is to reduce the number of vehicles on road. This means that the percentage of car owner in a society using Ridesharing will be decreased. Hence, hopefully, in the future the scenario of 35% drivers might be the most realistic. For the time being, the scenarios of drivers reaching 50% or 80% are most realistic for the reason we described earlier.

4.2.2 Results

Now, we provide the experimental results according to which we evaluate our method. In the first part of the evaluation, we examine four factors. The first factor is the *average extra distance* (AED), which refers to the increase percentage of the distance that drivers cover when collaborating with passengers. This percentage is calculated according to Equation 4.4.

$$AED = 100 \cdot \frac{final - initial}{initial} \quad (\%) \tag{4.4}$$

The variable *final* refers to the distance, in kilometers, that a driver covers in order to pick up and drop off her passengers. On the other hand, *initial* refers to the distance that the driver would have driven if she was alone. If the final distance is double the initial distance, there is a 100% increase. In addition, we provide the average number of passengers per car (PPC). Second, we provide the *cost reduction* (CR) for drivers in every case. In other words, the percentage of the economic relief for a driver when she participates in Ridesharing as given from the Equation 3.18 (*page 50*). Finally, we evaluate the *average coalition's preference satisfaction rate* (CPR) which is calculated as we explained in section 4.2.1 (Equation 4.3).

4.2.2.1 Rethymno

First, we provide results for the city of Rethymno which is the smallest one in our case study. Its corresponding city graph consists of 1321 vertices. We consider three cases about the percentage of drivers among agents. In the first case, only 35% of participants are drivers. In the second, 50% of them are drivers, and finally in the third case, 80% of agents are drivers. The experimental results for Rethymno were computed for 500, 1000 and 1500 agents. The average distance for drivers without Ridesharing in the city of Rethymno was computed to be 1.1 km. This was calculated by a sample of thousand of drivers with standard deviation 0.03. Hence, we consider this value to be accurate and we take it under consideration in our results for Rethymno.

# of Agents		35% *drivers*	50% *drivers*	80% *drivers*
500	AED	+117.6%	+91.5%	+52.8%
	PPC	2.84	2.4	1.54
	CR	−81.9%	−77.2%	−67.6%
	CPR	46.09%	46.18%	44.63%
1000	AED	+109.5%	+87.9%	+48.4%
	PPC	2.9	2.46	1.53
	CR	−81.66%	−77.12%	−67%
	CPR	46.44%	46.13%	44.88%
1500	AED	+108.7%	+82.5%	+45.6%
	PPC	2.95	2.45	1.51
	CR	−81.84%	−76.8%	−66.9%
	CPR	46.74%	45.92%	45.25%

Table 4.1: Rethymno 1^{st}*scenario*

# of Agents		35% *drivers*	50% *drivers*	80% *drivers*
500	AED	+117.6%	+94.5%	+51.5%
	PPC	2.88	2.38	1.51%
	CR	−82.95%	−76.96%	−65.37%
	CPR	56.28%	60.31%	68.46%
1000	AED	+112.8%	+91.38%	+46%
	PPC	2.9	2.45	1.51
	CR	−81.91%	−77.3%	−67%
	CPR	61.68%	67%	72.15%
1500	AED	+110.1%	+86.21%	+48.9%
	PPC	2.93	2.24	1.54
	CR	−81.9%	−76.63%	−66.97%
	CPR	64.52%	69.33%	77%

Table 4.2: Rethymno $2^{nd} scenario$

# of Agents		35% *drivers*	50% *drivers*	80% *drivers*
500	AED	+119.5%	+92.7%	+52.32%
	PPC	2.78	2.37	1.5
	CR	−82.2%	−76.86%	−67.23%
	CPR	63.45%	66.81%	75.62%
1000	AED	+112.5%	+90.4%	+48.7%
	PPC	2.88	2.4	1.55
	CR	−81.93%	−76.82%	−67.3%
	CPR	69.32%	73.14%	81.59%
1500	AED	+112.77%	+84.2%	+49%
	PPC	2.94	2.47	1.54
	CR	−82.26%	−77.43%	−67.2%
	CPR	72%	76.44%	84.26%

Table 4.3: Rethymno $3^{rd} scenario$

# of Agents		35% *drivers*	50% *drivers*	80% *drivers*
500	AED	+121.2%	+107.9%	+57.8%
	PPC	2.79	2.38	1.51
	CR	−82%	−78.6%	−68.41%
	CPR	72.47%	77.58%	84.69%
1000	AED	+121.1%	+99.2%	+57.41%
	PPC	2.96	2.43	1.6
	CR	−82.87%	−78.12%	−68.65%
	CPR	77.39%	82.12%	87.95%
1500	AED	+119.6%	+98%	+57%
	PPC	2.95	2.46	1.6
	CR	−82.78%	−72.86%	−68.8%
	CPR	78.85%	83.46%	90.30%

Table 4.4: Rethymno 4^{th}*scenario*

Generally, we observe that while the number of drivers among agents is increased, the average extra distance (AED) is decreased. This is expected, since there are more alternatives in order to keep detours at low level. The reduction of expenses for drivers is proportional to the extra distance which they need to cover in order to pick up and drop off their passengers. In addition, we observe that in general, the number of agents is inversely proportional to the average extra distance.

Concerning the four scenarios about the parameters α, β and γ, we notice that when $\gamma = 0$, first scenario, the percentage of randomly covered preferences is about 44% to 46%, regardless of the number of agents and the percentage of drivers. While γ is increased, we observe an improvement in preference rate (CPR) which is also proportional to both the number of agents and the percentage of drivers. Those two quantities lead to bigger variety of possible combinations for coalitions when they are increased. Thus, the final coalitions are the ones which achieve a higher welfare.

About the number of passengers per car (PPC), we observe that it is affected by the number of agents. More specifically, while the number of agents is increased, PPC is also improved. This is expected since more passengers seek for a seat in a vehicle. In addition, PPC depends on the percentage of drivers among agents. While drivers' percentage is increased, PPC is reduced. However, we observe that PPC is not affected by each scenario, which means that the parameters α, β and γ have no effect on the number of passenger for the vehicles.

We notice that the improvement of the preference rate comes with a cost. More specifically, the average extra distance is also increased. That means that drivers need to cover more kilometers in order to accomplish the route for every passenger. More specifically, we observe that the first three scenarios have similar values in AED. However, CPR is changing because the parameter γ is gradually increased, thus more preferences are satisfied. In the fourth scenario where γ gets its greater value, $\gamma = 0.4$, CPR is higher than in previous scenarios, but AED has also been increased. Hence, we conclude that we can improve the participants' satisfaction without enlarging the routes, up to a point. If we exceed that point, we can achieve even better satisfaction but with larger routes.

The third scenario seems to be more efficient than the others, considering the combination between preference satisfaction (CPR) and the average extra distance (AED) for drivers. For example, when the number of agents is 1000 and the percentage of drivers is 80%, the AED for a driver who was assigned to pick up some passengers is 48.7%. In addition, in this case, drivers achieve an average cost reduction of 67.3%, while at the same time they offer transportation to 1.55 passengers on average. The CPR is 81.59%, which means that most of the passengers belong to coalitions where their co-passengers satisfy their preferences. For the second scenario, we observe that AED is 46%, negligible deviation from the AED of the third scenario. The cost reduction is 67% and the PPC is 1.51. The preference rate (CPR) is 72.15%, which is 9.44% lower than the CPR of the third scenario, a significant deviation.

Finally, for the fourth scenario, we notice that CPR reaches 87.95%, which is the higher observed value in all four scenarios for the case of 1000 agents with 80% drivers. However, the AED in this case is 57.41%, much higher than in previous scenarios. Hence, we conclude that when the parameter γ gets high values, we can achieve better results on preferences' satisfaction, but the average extra distance is also inevitably increased.

4.2.2.2 Agios Nikolaos

We are now going to present the results of our approach when applied on Agios Nikolaos. The city graph is slightly larger than the graph of Rethymno and it contains 2185 vertices. Hence, in these experiments we tried different numbers of agents. More specifically, we consider 700, 1200 and 2000 agents. Similar to Rethymno, the average distance for drivers in Agios Nikolaos is 1.11 km. This value was produced via a sample of thousands of drivers with standard deviation of 0.03.

# of Agents		35% *drivers*	50% *drivers*	80% *drivers*
700	AED	+111.6%	+87.4%	+49.4%
	PPC	2.8	2.33	1.47
	CR	−82.1%	−76.64%	−67%
	CPR	46.3%	46.43%	45.53%
1200	AED	+105.7%	+83.2%	+46.7%
	PPC	2.88	2.34%	1.5
	CR	−81.8%	−76.4%	−67%
	CPR	46.11%	44.78%	45.49%
2000	AED	+103.9%	+82.1%	+43.7%
	PPC	2.89	2.4	1.5
	CR	−81.66%	−76.5%	−66.42%
	CPR	46.71%	46.49%	46.34%

Table 4.5: Agios Nikolaos 1^{st}*scenario*

# of Agents		35% *drivers*	50% *drivers*	80% *drivers*
700	AED	+108.5%	+88.3%	+48.6%
	PPC	2.8	2.32	1.5
	CR	−81.37%	−76.75%	−67%
	CPR	60.38%	64.33%	69.92%
1200	AED	+106%	+85.3%	+46.7%
	PPC	2.85	2.37	1.51
	CR	−81.63%	−76.77%	−66.91%
	CPR	63.45%	68.5%	74.46%
2000	AED	+102.9%	+82.8%	+46.62%
	PPC	2.87	2.4	1.53
	CR	−76%	−76.8%	−67.36%
	CPR	68.63%	72.9%	79.31%

Table 4.6: Agios Nikolaos $2^{nd}scenario$

# of Agents		35% *drivers*	50% *drivers*	80% *drivers*
700	AED	+111.5%	+89.5%	+51%
	PPC	2.8	2.33	1.5
	CR	−81.8%	−76.8%	−67.5%
	CPR	67.05%	70.59%	80.21%
1200	AED	+112.3%	+80.4%	+49.9%
	PPC	2.84	2.38	1.52
	CR	−81.8%	−76.8%	−67.7%
	CPR	70.99%	75.85%	82.22%
2000	AED	+106.5%	+84.8%	$+48.8\%km$
	PPC	2.9	2.4	1.55
	CR	−81.8%	−76.9%	−67.6%
	CPR	74.88%	79%	85.42%

Table 4.7: Agios Nikolaos $3^{rd}scenario$

# of Agents		35% *drivers*	50% *drivers*	80% *drivers*
700	AED	+118%	+95.4%	+57.5%
	PPC	2.84	2.4	1.56
	CR	−82.5%	−77.9%	−69%
	CPR	74.83%	79.8%	85.58%
1200	AED	+117.2%	+93.1%	+55.5%
	PPC	2.88	2.42	1.57
	CR	−82.6%	−77.76%	−69%
	CPR	78.5%	82.22%	88.06%
2000	AED	+113.6%	+90.4%	+56.4%
	PPC	2.91	2.4	1.65
	CR	−82.5%	−77.9%	−69.5%
	CPR	80.56%	84.1%	90.84%

Table 4.8: Agios Nikolaos 4^{th}*scenario*

In the first scenario, we observe that the percentage of preferences (CPR) which are satisfied randomly is about 44% to 46%, similarly to the first scenario for Rethymno. Concerning the average extra distance for drivers (AED), we notice that it is inversely proportional to the number of agents and the percentage of drivers. When drivers constitute 35% of the total set, the average extra distance is slightly higher than 100%, which means that drivers need to travel almost double distance in order to pick up and drop off all their passengers. However, according to PPC, each driver offers transportation to 2.8 passengers on average. For 50%, AED ranges from 80% to 90%, it depends on the number of agents and the percentage of drivers, the number of passengers per car (PPC) is reduced to ∼2.4 compared to the case of 35% drivers. When 80% of the agents are drivers, AED is limited between ∼40% to ∼60%. For the second scenario, where $\gamma = 0.1$, we observe an important improvement in preference rates, while AED is not affected. Furthermore, in the third scenario where $\gamma = 0.2$, we notice an even bigger improvement on satisfied preferences along with a tiny rise of AED. Finally, in the last scenario, we observe high CPR; however, AED is also inevitably enlarged. It seems conclusively that the third scenario is the most efficient, in terms of preferences satisfaction while at the same time the detours remain short.

4.2.2.3 Chania

Subsequently, we continue to larger city graphs and for higher numbers of agents. Chania is the second biggest city of Crete. The city graph consists of 3278 nodes. The experimental results contain 800, 1600 and 2500 agents respectively. Among thousands of agents in our experiments, it was calculated that the average distance for a driver without passengers in 1.87 kilometers. We are interested in AED and how it is affected in larger cities. We search for the optimal scenario in order to balance acceptable extra distances and satisfying preference rates.

# of Agents		35% *drivers*	50% *drivers*	80% *drivers*
800	AED	+107.1%	+84.8%	+47.9%
	PPC	2.85	2.37	1.47
	CR	−81.8%	−76.9%	−66.8%
	CPR	46.36%	45%	45.17%
1600	AED	+102.8%	+83.3%	+45.8%
	PPC	2.87	2.4	1.5
	CR	−81.6%	−76.6%	−66.8%
	CPR	46.82%	46.3%	44.51%
2500	AED	+103.4%	+80.2%	+44.4%
	PPC	2.9	2.38	1.52
	CR	−81.8%	−76.3%	−66.7%
	CPR	46.83%	46.23%	46.34%

Table 4.9: Chania $1^{st} scenario$

# of Agents		35% *drivers*	50% *drivers*	80% *drivers*
800	AED	+108.2%	+85.2%	+49.8%
	PPC	2.82	2.35	1.51
	CR	−81.65%	−76.5%	−63%
	CPR	60.84%	63.23%	70.19%
1600	AED	+105.5%	+82.5%	+47.9%
	PPC	2.85	2.35	1.53
	CR	−81.7%	−76.6%	−67.3%
	CPR	65.27%	69.05%	76.81%
2500	AED	+105.3%	+84.7%	+45.7%
	PPC	2.92	2.44	1.51
	CR	−82%	−77.13%	−67.1%
	CPR	68%	72.57%	79.57%

Table 4.10: Chania 2^{nd}*scenario*

# of Agents		35% *drivers*	50% *drivers*	80% *drivers*
800	AED	+104.6%	+84%	+48.8%
	PPC	2.9	2.4	1.52
	CR	−81.9%	−76.9%	−67.1%
	CPR	67.23%	71.46%	79.58%
1600	AED	+100%	+81.7%	+46.2%
	PPC	2.95	2.43	1.55
	CR	−81.8%	−76.6%	−67.1%
	CPR	72.16%	76.74%	82.97%
2500	AED	+98.6%	+79.3%	+46.4%
	PPC	2.96	2.46	1.6
	CR	−81.7%	−76.7%	−67.2%
	CPR	75%	79.26%	86.8%

Table 4.11: Chania 3^{rd}*scenario*

# of Agents		35% *drivers*	50% *drivers*	80% *drivers*
800	AED	+109.6%	+89.3%	+55.92%
	PPC	2.84	2.37	1.62
	CR	−81.7%	−77%	−68.9%
	CPR	75.89%	80.9%	87.21%
1600	AED	+108%	+89.4%	+52.1%
	PPC	2.9	2.45	1.61
	CR	−82.1%	−77.6%	−68.3%
	CPR	79.56%	84%	89.95%
2500	AED	+104.7%	+87.6%	+51.8%
	PPC	2.97	2.51	1.66
	CR	−82.2%	−77.9%	−68.7%
	CPR	81.49%	85.2%	91.06%

Table 4.12: Chania 4th*scenario*

Similar to both previous cities, we observe that for $\gamma = 0$, preferences are satisfied by 44% to 46%. In the second and third scenarios, we notice that they both have similar AED but the CPR in scenario 3 is higher. This means that without deviation in extra distance for drivers, we can achieve greater satisfaction for the participants. The fourth scenario shows that when the parameter γ is further increased, the CPR can be improved. However, that causes an increase in AED which we want to avoid. Hence, we conclude that the most profitable scenario is the third one, since it offers the best balance between AED and CPR.

Furthermore, we observe that generally the AED is lower in Chania compared to the previous, smaller cities. In every case, although the distances that drivers need to cover in Chania are longer, due to the city's extent, AED remains lower.

4.2.2.4 Heraklion

The fourth city of our case study is Heraklion. It is the biggest city of Crete and its corresponding graph is much larger than all the previous. It consists of 9084 vertices. For this city, our experiments contain 1000, 2500 and 4000 agents. Our purpose is to evaluate the performance of our implementation when it is applied on a large city with a significant amount of agents.

# of Agents		35% *drivers*	50% *drivers*	80% *drivers*
1000	AED	+127.8%	+99.9%	+55.2%
	PPC	2.88	2.4	1.52
	CR	−82.4%	−77.5%	−67.5%
	CPR	46.55%	47%	46.15%
2500	AED	+128.9%	+99.3%	+52.5%
	PPC	2.98	2.44	1.53
	CR	−83.1%	−77.4%	−67%
	CPR	46.68%	46.76%	46.03%
4000	AED	+127.9%	+96.6%	+51.1%
	PPC	3	2.44	1.51
	CR	−82.9%	−77.2%	−66.9%
	CPR	47.27%	46.54%	45.51%

Table 4.13: Heraklion 1^{st}*scenario*

# of Agents		35% *drivers*	50% *drivers*	80% *drivers*
1000	AED	+133.9%	+105%	+55.4%
	PPC	2.92	2.43	1.53
	CR	−83.1%	−77.9%	−68.1%
	CPR	62.32%	64.98%	73.53%
2500	AED	+128.3%	+101.2%	+55.3%
	PPC	2.93	2.46	1.55
	CR	−82.5%	−77.9%	−68%
	CPR	69.21%	73.33%	79.72%
4000	AED	+129.4%	+100.5%	+55.2%
	PPC	3	2.5	1.56
	CR	−83.2%	−78%	−67.9%
	CPR	71.48%	77.16%	83.32%

Table 4.14: Heraklion 2^{nd}*scenario*

# of Agents		35% *drivers*	50% *drivers*	80% *drivers*
1000	AED	+132.5%	+106.5%	+57.9%
	PPC	2.87	2.45	1.55
	CR	−82.8%	−78.46%	−68.3%
	CPR	69.71%	74.42%	79.82%
2500	AED	+132.8%	+103.4%	+57.1%
	PPC	2.97	2.5	1.58
	CR	−83.2%	−78.2%	−68.4%
	CPR	74.86%	80.61%	86.26%
4000	AED	+131.2%	+103.4%	+58.3%
	PPC	2.98	2.52	1.63
	CR	−83.2%	−78.5%	−68.9%
	CPR	78%	82.34%	88.36%

Table 4.15: Heraklion $3^{rd}scenario$

# of Agents		35% *drivers*	50% *drivers*	80% *drivers*
1000	AED	+137.3%	+111.5%	+62.2%
	PPC	2.8	2.46	1.57
	CR	−82.8%	−79%	−69%
	CPR	78.06%	81.76%	90.2%
2500	AED	+137.3%	+108.4%	+63.7%
	PPC	2.97	2.52	1.68
	CR	−83.6%	−79%	−70%
	CPR	81.81%	85.73%	90.76%
4000	AED	+137.4%	+109.3%	+64.1%
	PPC	3	2.55	1.7
	CR	−83.7%	−79.3%	−70.1%
	CPR	83.4%	86.88%	92.75%

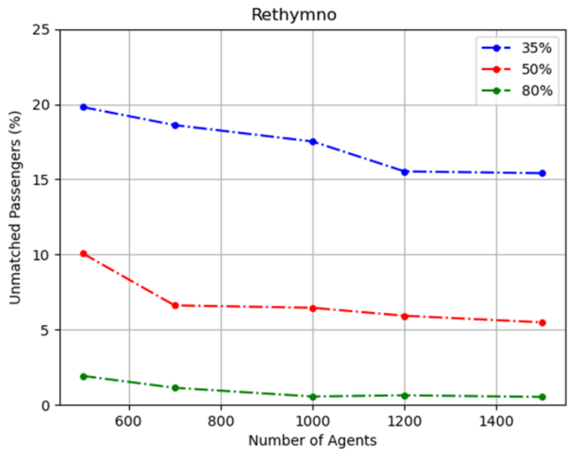
Table 4.16: Heraklion $4^{th}scenario$

As we mentioned, the city of Heraklion extends over a large area. Nevertheless, we observe that the performance follows the same pattern as the previous cities. Essentially, the most profitable results are produced in the second and the third scenarios, where the preference rate is satisfying while at the same time the extra distance is acceptable. We notice that second and third scenarios have almost the same AED, which means that in both scenarios the drivers need to cover the same extra distance in order to complete the route. However, the third scenario has a more efficient performance because CPR is greater, more preferences are satisfied. Hence, we would chose the third scenario among all four.

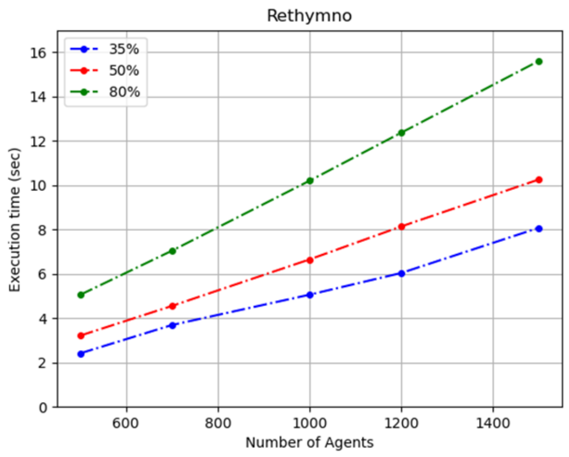
The main concern of this thesis is the satisfaction of preferences, since we aim to create an application which facilitates the transportation, but at the same time it offers interpersonal relationships. Our purpose in this work is matching agents with collaborators with whom they feel more comfortable. According to the experiments above, our implementation achieves that goal in a large extent. Alongside, we pursue to maintain short routes in order to convince drivers to take part. Our experiments show that the extra distance depends on the city, and its extent, as well as on our tolerance. In our approach, we are flexible concerning the distance in order to achieve satisfying results about preferences.

Subsequently, we are going to evaluate the efficacy of our implementation concerning the percentage of unmatched passengers, i.e., passengers who did not join any coalition due to incompatibility or because other passengers were more convenient that them. As we explained in section 3.2.1, we aim at matching every passenger to a suitable coalition regarding spatial requirements and restrictions. However, drivers should not be forced to cover much longer distances. Hence, some passengers who are not very compatible, they will not end up to any coalition, thus they will form singletons. Of course, we aim to keep the percentage of those agents as low as possible. In the next experiments, we focus on this factor alongside with the execution time. We provide results for various numbers of agents at each city.

We are now going to present the diagrams of time and efficiency evaluation. For each city the number of agents is proportional to its size. The execution time concerns the determination of the coalition structure. In other words, it is the required time in order for the PACR-CF algorithm to be completed. The execution time of the CFRE algorithm is not included. Finally, we consider the percentage of unmatched passengers in relation to the number of agents and the percentage of drivers.



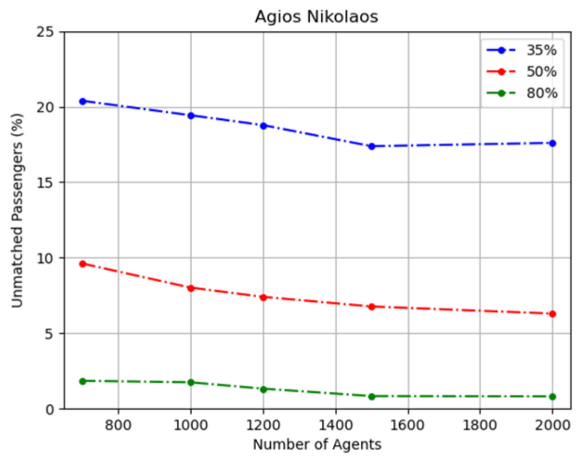
(a) Unmatched Passengers Results



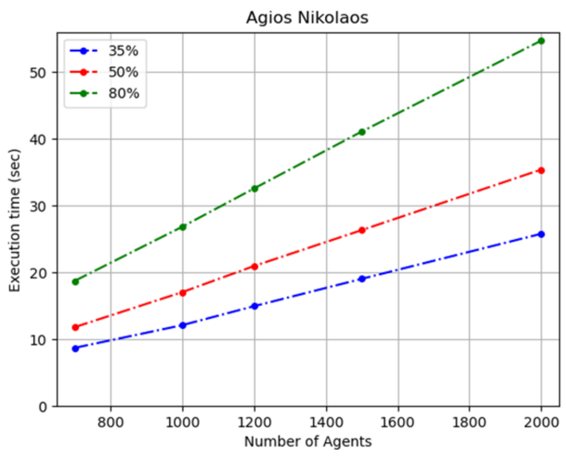
(b) Execution Time Results

Figure 4.3: Unmatched Passengers and Execution time for Rethymno

For the city of Rethymno, the results were measured for a number of agents between 500 and 1500. We observe that for 35% drivers, the percentage of the unmatched passengers is 15% to 20% and it is inversely proportional to the number of agents. For 50%, it is reduced significantly between 5% and 10%, while for 80% drivers the percentage of unmatched passenger is slightly higher that 0%. Regarding the execution time, we notice that it increases linearly according to the number of agents. In addition, higher percentage of drivers leads to an increase of the execution time. In the case of 1500 agents, for 80% drivers our algorithm produces the coalition structure in less than 16 seconds. For 50%, the results are given in almost 10 seconds, while for 35% in just 8 seconds.



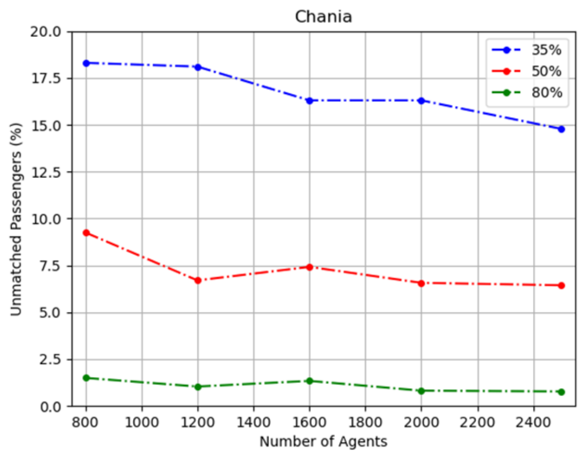
(a) Unmatched Passengers Results



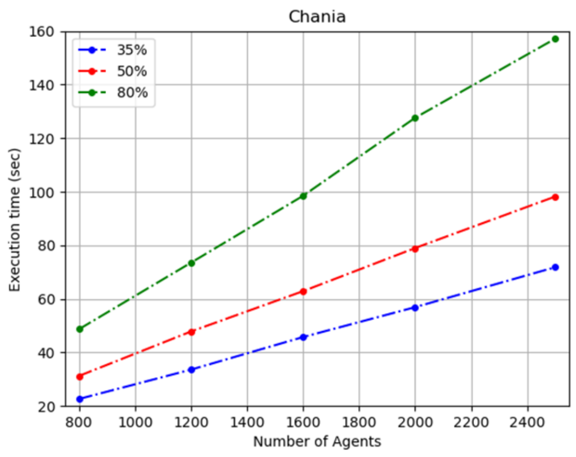
(b) Execution Time Results

Figure 4.4: Unmatched Passengers and Execution time for Agios Nikolaos

Agios Nikolaos has a slightly larger city graph than Rethymno, thus the number of agents ranges from 700 to 2000. We observe similar results to Rethymno about the percentage of unmatched drivers concerning the three different cases. Since the city graph is wider, the pathfinding algorithm needs more time in order to compute the routes. This is the reason why the execution time is increased in comparison to the execution time for Rethymno. For 2000 agents, our algorithm produces the results in less than a minute for the case of 80% drivers. For 50%, it needs almost 35 seconds, and finally for 35% it needs just 25 seconds.



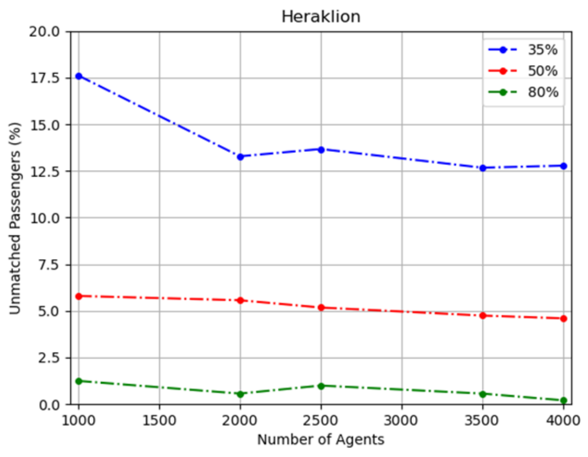
(a) Unmatched Passengers Results



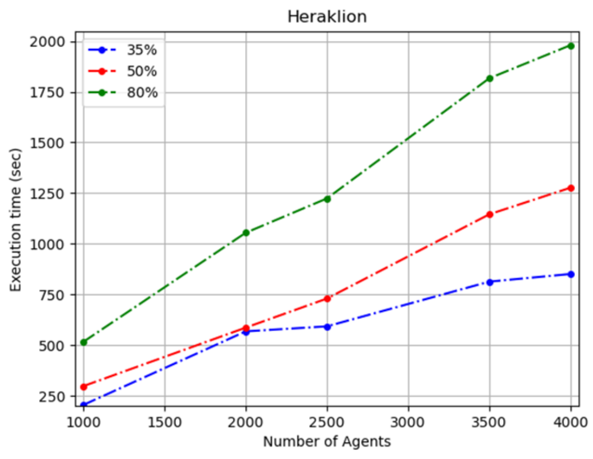
(b) Execution Time Results

Figure 4.5: Unmatched Passengers and Execution time for Chania

Similarly to the previous cities, the percentage of unmatched passengers is satisfying. For 35%, it is between 15% and 18% which is a positive outcome. For 50%, it is about 7.5%, while for 80% it is almost 1%. Regarding the execution time, we observe the same linear increase. However, the time has been increased in comparison to the previous cities because the city graph is larger. In particular, for 2500 agents where 80% of them constitute the set of drivers, the algorithm needs almost 2.6 minutes in order to produce the results. For 50%, 1.6 minutes are needed, while for 35% just 75 seconds. So far, we notice that the time of computation for the final outcome is short, for all three cities. This is necessary for an application of short notice rides where the distribution of agents into vehicles must be quick. Nevertheless, we need to consider the scenario of a much larger city where the city graph is dense and the number of agents corresponds to its size. Heraklion represents this kind of cities and the number of agents in this case study, ranges from 1000 to 4000.



(a) Unmatched Passengers Results



(b) Execution Time Results

Figure 4.6: Unmatched Passengers and Execution time for Heraklion

In this case, we observe high-quality efficacy concerning the unmatched passengers where the percentage is almost 12.5% when the drivers constitute 35% of agents. For 50% drivers, it is just 5%, while for 80% it is 1%, similarly to the previous cities. Hence, we conclude that in terms of efficiency, our algorithm has the similar, satisfying performance when applied to cities of various sizes. However, we observe a significant increase in execution time which restricts the ability to apply our method to large scale cities for short notice rides. When 4000 agents participate, the execution time for 35% drivers is almost 14 minutes. For 50%, it is 20.8 minutes, and finally for 80%, the algorithm needs almost 33 minutes. These execution times are not prohibitive; however, they are not short enough in order to determine the coalition structure instantly. Hence, we conclude that our method is recommended for large scale cities when the distribution of agents into cars needs to be determined for later on.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we present the Ridesharing problem and we propose an approach for the solution. Our method is based on characteristic function games and it aims at finding a coalition structure which corresponds to an overall welfare that approaches the optimal. Hypergraphs are used in order to partition the city and create an initial grouping for the participants. Simple graphs depict the interaction between agents. Most of the related works focus on minimising the overall cost by merging similar routes. In this work, we consider that drivers are ordinary people who are willing to share their cars with others in order to facilitate them and at the same time to reduce their own expenses. In addition, we take into consideration agents' preferences about collaborators and we pursue to satisfy most of them. In our method, we introduce a main algorithm called *PASR-CF* which acts in a greedy manner and its purpose is to construct an efficient coalition structure. We also introduce two more algorithms which are used in order to determine the optimal subset among a set of potential passengers, and to compute the final route for each vehicle. Finally, we systematically evaluate our approach when applied on four cities of different sizes. We come to the conclusion that our approach is highly effective for small and middle-size cities which can be depicted by succinct city graphs. The coalitions have high quality in terms of preferences satisfaction and mobility similarity among members. However, for large scale cities our method is still effective, but the program needs to be executed in computers with high processing power.

5.2 Future Work

In this thesis, we approach the Ridesharing problem for short notice rides, thus we did not include the factor of time. However, this constitutes an extension which could be considered in a future work in order to implement an integrated application. Each agent sets her own departure time apart from starting and destination points. This is an extra restriction which has to be satisfied in realistic scenarios. Our method uses a greedy algorithm, hence the outcome is sub-optimal. Therefore, more work to come up with even more effective and efficient algorithm is required. In order to emphasise on preferences satisfaction, we did not set an upper bound for drivers' detours. A different approach could set some limits on the accepted extra distance for drivers. Furthermore, if the procedure is repeated, meaning that the application is used daily, a potential improvement would be to apply learning of agents' preferences over time in order to achieve better matching. In addition, input data could be used to learn preferences patterns in order to group efficiently the passengers without expressed preferences. Moreover, a more elaborate preference aggregation process is in order, employing related methods provided by *social choice theory* [34]. Finally, an extension of the Ridesharing problem is the payment distribution among members in a fair and representative way. In other words, each member covers a part of the overall cost which is proportional to her contribution.

References

[1] Drews, F., Luxen, D.: Multi-hop ride sharing. Proceedings of the 6th Annual Symposium on Combinatorial Search, SoCS 2013 (01 2013) 71–79 1, 11

[2] Bistaffa, F., Farinelli, A., Chalkiadakis, G., Ramchurn, S.D.: A cooperative game-theoretic approach to the social ridesharing problem. Artificial Intelligence **246** (2017) 86–117 1, 2, 3, 11, 19, 21

[3] Pelzer, D., Xiao, J., Zehe, D., Lees, M., Knoll, A., Aydt, H.: A partition-based match making algorithm for dynamic ridesharing. IEEE Transactions on Intelligent Transportation Systems **16** (10 2015) 1–12 1, 2, 18

[4] Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., Rus, D.: On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. Proceedings of the National Academy of Sciences **114** (01 2017) 201611675 1, 2, 3, 18

[5] Bicocchi, N., Mamei, M., Sassi, A., Zambonelli, F.: On recommending opportunistic rides. IEEE Transactions on Intelligent Transportation Systems **18**(12) (2017) 3328–3338 1, 18

[6] Riley, C., Legrain, A., Hentenryck, P.V.: Column generation for real-time ride-sharing operations. In Rousseau, L., Stergiou, K., eds.: Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 16th International Conference, CPAIOR 2019, Thessaloniki, Greece, June 4-7, 2019, Proceedings. Volume 11494 of Lecture Notes in Computer Science., Springer (2019) 472–487 2, 3, 19

[7] Wooldridge, M.: An Introduction to MultiAgent Systems. 2nd edn. Wiley Publishing (2009) 2

[8] Chalkiadakis, G., Elkind, E., Wooldridge, M.: Computational Aspects of Cooperative Game Theory (Synthesis Lectures on Artificial Inetlligence and Machine Learning). 1st edn. Morgan &amp; Claypool Publishers (2011) 2, 15, 16, 17

[9] Deo, N.: Graph Theory with Applications to Engineering and Computer Science (Prentice Hall Series in Automatic Computation). Prentice-Hall, Inc., USA (1974) 6, 8

[10] Wilson, R.J.: Introduction to Graph Theory. John Wiley & Sons, Inc., USA (1986) 7

[11] Joshi, K.D.: Foundations of Discrete Mathematics. John Wiley & Sons, Inc., USA (1989) 7

[12] Applegate, D.L., Bixby, R.E., Chvatal, V., Cook, W.J.: The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics). Princeton University Press, USA (2007) 7, 45

[13] Williamson, E.: Lists, Decisions and Graphs. S. Gill Williamson 8

[14] Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach (2nd Edition). Prentice Hall (December 2002) 8

[15] Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische mathematik **1**(1) (1959) 269–271 10

[16] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, Third Edition. 3rd edn. The MIT Press (2009) 10

[17] Dial, R.B.: Algorithm 360: Shortest-path forest with topological ordering [h]. Commun. ACM **12**(11) (November 1969) 632–633 10

[18] Mehlhorn, K., Sanders, P.: Algorithms and data structures: The basic toolbox. Algorithms and Data Structures: The Basic Toolbox (01 2008) 10, 11

[19] Berge, C.: Hypergraphs - combinatorics of finite sets. Volume 45 of North-Holland mathematical library. North-Holland (1989) 13

[20] Valdivia, P., Buono, P., Plaisant, C., Dufournaud, N., Fekete, J.D.: Analyzing dynamic hypergraphs with parallel aggregated ordered hypergraph visualization. IEEE Transactions on Visualization and Computer Graphics **27**(1) (2021) 1–13 13

[21] Zhou, D., Huang, J., Schölkopf, B.: Learning with hypergraphs: Clustering, classification, and embedding. Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference, 1601-1608 (2007) **19** (01 2006) 1601–1608 14

[22] Wooldridge, M.: An Introduction to MultiAgent Systems. 2nd edn. Wiley Publishing (2009) 15

[23] Rahwan, T., Ramchurn, S., Jennings, N., Giovannucci, A.: An anytime algorithm for optimal coalition structure generation. Journal of Artificial Intelligence Research **34** (04 2009) 15

[24] Myerson, R.: Game Theory: Analysis of Conflict. Harvard University Press (1997) 15

[25] Sandholm, T., Larson, K., Andersson, M., Shehory, O., Tohmé, F.: Coalition structure generation with worst case guarantees. Artificial Intelligence **111**(1) (1999) 209–238 15

[26] Shapley, L.S.: A Value for N-Person Games. RAND Corporation, Santa Monica, CA (1952) 16

[27] Deng, X., Papadimitriou, C.H.: On the complexity of cooperative solution concepts. Math. Oper. Res. **19**(2) (May 1994) 257–266 17

[28] Bistaffa, F., Chalkiadakis, G., Farinelli, A.: Efficient coalition structure generation via approximately equivalent induced subgraph games. IEEE Transactions on Cybernetics (2021) 1–11 17

[29] Kendall, J.: Hard and soft constraints in linear programming. Omega **3**(6) (1975) 709–715 22

[30] Bruckner, A.: Tests for the superadditivity of functions. Proceedings of The American Mathematical Society - proc amer math soc **13** (02 1962) 37

[31] Binshtok, M., Brafman, R.I., Shimony, S.E., Martin, A., Boutilier, C.: Computing optimal subsets. In: Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 2. AAAI'07, AAAI Press (2007) 1231–1236 39, 41, 43

[32] Dang, V.D., Dash, R.K., Rogers, A., Jennings, N.R.: Overlapping coalition formation for efficient data fusion in multi-sensor networks. In: Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1. AAAI'06, AAAI Press (2006) 635–640 43

[33] Bistaffa, F., Farinelli, A., Ramchurn, S.D.: Sharing rides with friends: A coalition formation algorithm for ridesharing. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. AAAI'15, AAAI Press (2015) 608–614 49

[34] Moulin, H.: Handbook of Computational Social Choice. Cambridge University Press (2016) 78