# OPTIMIZATION AND MECHANISM DESIGN FOR RIDESHARING SERVICES

A Dissertation

by

WEI LU

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

| | |
|---|---|
| Chair of Committee, | Luca Quadrifoglio |
| Committee Members, | Sivakumar Rathinam |
| | Xiubin Bruce Wang |
| | Yunlong Zhang |
| Head of Department, | Robin Autenrieth |

December 2015

Major Subject: Civil Engineering

# ABSTRACT

Ridesharing services, whose aim is to gather travelers with similar itineraries and compatible schedules, are able to provide substantial environmental and social benefits through reducing the use of private vehicles. When the operations of a ridesharing system is optimized, it can also save travelers a significant amount of transportation cost. The economic benefits associated with ridesharing in turn attract more travelers to participate in ridesharing services and thereby improve the utilization of transportation infrastructure capacity.

This study addresses two of the most challenging issues in designing an efficient and sustainable ridesharing service: ridesharing optimization and ridesharing market design. The first part of the dissertation formally defines the large-scale ridesharing optimization problem, characterizes its complexity and discusses its relation to classic relevant problems like the traveling salesman problem (TSP) and the vehicle routing problem (VRP). A mixed-integer program (MIP) model is developed to solve the ridesharing optimization problem. Since the ridesharing optimization problem is NP-hard, the MIP model is not able to solve larger instances within a reasonable time. An insertion-based heuristic is developed to get approximate solutions to the ridesharing optimization problem. Experiments showed that ridesharing can significantly reduce the system-wide travel cost and vehicle trips. Evaluation of the heuristic solution method showed that the heuristic approach can solve the problem very fast and provide nearly-optimal (98%) solutions, thus, confirming its efficiency and accuracy.

From a societal perspective, the ridesharing optimization model proposed in this dissertation provided substantial system-wide travel cost saving (25%+) and vehicle-

trip saving (50%) compared to non-ridesharing situation. However, the system-level optimal solution might not completely align with individual participant interest. The second part of this dissertation formulates this issue as a fair cost allocation problem through the lens of the cooperative game theory.

A special property of the cooperative ridesharing game is that, its characteristic function values are calculated by solving an optimization problem. We characterize the game to be monotone and subadditive, but non-convex. Several concepts of fairness are investigated and special attention is paid to a solution concept named nucleolus, which aims to minimize the maximum dissatisfaction in the system. However, finding the nucleolus is very challenging because it requires solving the ridesharing optimization problem for every possible coalition, whose number grows exponentially as the number of participants increases in the system. We break the cost allocation (nucleolus finding) problem into a master-subproblem structure and two subproblems are developed to generate constraints for the master problem. We propose a coalition generation procedure to find the nucleolus and approximate nucleolus of the game. When the game has a non-empty core, in the approximate nucleolus scheme the coalitions are computed only when it is necessary, and the approximate nucleolus scheme produces the actual nucleolus. Experimental results showed that, when the game has an empty core, the approximate nucleolus is close to the actual nucleolus. Results also showed that, regardless of the emptiness of the game, by using our algorithm, only a small fraction (1.6%) of the total coalition constraints were generated to compute the approximate nucleolus, and the approximate nucleolus is close to the actual nucleolus.

# DEDICATION

To my mother and father.

TABLE OF CONTENTS

Page

ABSTRACT . . . . . ii

DEDICATION . . . . . iv

TABLE OF CONTENTS . . . . . v

LIST OF FIGURES . . . . . vii

LIST OF TABLES . . . . . viii

1. INTRODUCTION . . . . . 1

1.1 Background and Motivation . . . . . 1
1.2 Outline . . . . . 3

2. LITERATURE REVIEW . . . . . 5

2.1 Mechanism Design . . . . . 5
2.2 Mechanism Design in Ridesharing . . . . . 8
2.3 Algorithmic Studies on Ridesharing-related Problems . . . . . 9

3. DEFINITIONS AND PROBLEM DESCRIPTION . . . . . 12

3.1 Ridesharing Settings . . . . . 12
3.1.1 Ridesharing system objective . . . . . 14
3.2 Mechanism Design Problem . . . . . 14
3.2.1 Value of a shared-ride for an agent . . . . . 14
3.2.2 Mechanism design preliminaries . . . . . 16
3.3 Mechanism Design Problem in Ridesharing . . . . . 16

4. RIDESHARING OPTIMIZATION PROBLEM . . . . . 19

4.1 Introduction . . . . . 19
4.1.1 Ridesharing system objectives . . . . . 19
4.1.2 Problem definition . . . . . 19
4.1.3 Cost and value of a shared-ride for an agent . . . . . 21
4.2 Modeling . . . . . 23
4.2.1 Setting . . . . . 23

4.2.2 Hardness . . . 24
4.2.3 Transformation . . . 25
4.2.4 Integer program . . . 25
4.3 Solution Methods . . . 28
4.3.1 Profitability of ridesharing . . . 28
4.3.2 One-to-one match . . . 29
4.3.3 An insertion heuristic . . . 30
4.4 Experiments . . . 32
4.5 Conclusions . . . 33

5. COOPERATIVE RIDESHARING GAME . . . 37

5.1 Introduction . . . 37
5.2 Cooperative Game Theory Background . . . 39
5.2.1 Solution concepts . . . 40
5.3 RSP From A Game Theory Perspective . . . 43
5.4 Fairness and Stability in RSP Game . . . 50
5.4.1 The core and the nucleolus . . . 50
5.4.2 An algorithm to find the nucleolus . . . 56
5.4.3 Coalition generation subproblem – general . . . 59
5.4.4 Coalition generation subproblem – non-empty core . . . 62
5.5 Experiments . . . 64
5.5.1 Approximate nucleolus . . . 64
5.5.2 Nucleolus vs. approximate nucleolus . . . 66
5.6 Conclusions . . . 67

6. CONCLUSIONS . . . 71

BIBLIOGRAPHY . . . 73

LIST OF FIGURES

FIGURE Page


3.1 A matching with detour . . . . . 13

4.1 (a) Individual trips vs. (b) organized ridesharing trips . . . . . 24

4.2 Corresponding feasible solution for SDMTSP-PD on the transformed graph . . . . . 26

4.3 A two-player example . . . . . 29

4.4 A profitable insertion . . . . . 31

5.1 Profitable vs. nonprofitable coalition . . . . . 49

5.2 A three player example . . . . . 53

5.3 Solution path – problem8a . . . . . 69

5.4 Solution path – problem8b . . . . . 69

LIST OF TABLES

TABLE Page


4.1 Experiment results . . . . . 36

4.2 Performance of algorithms - saving cost . . . . . 36

4.3 Performance of algorithms - saving vehicles . . . . . 36

5.1 Calculation of nucleolus - the bankruptcy game . . . . . 42

5.2 Calculation of nucleolus - empty core . . . . . 54

5.3 Calculation of nucleolus - nonempty core . . . . . 55

5.4 Data and approximate nucleolus of prob10c . . . . . 65

5.5 Data and approximate nucleolus of prob10d . . . . . 66

5.6 Nucleolus vs. approximate nucleolus – problem8a . . . . . 68

5.7 Nucleolus vs. approximate nucleolus – problem8b . . . . . 68

# 1. INTRODUCTION

## 1.1 Background and Motivation

Travelers today have a number of options when choosing transportation modes to go from origins to destinations. When selecting a transportation mode, people usually consider criteria such as cost-efficiency, time, reliability, convenience, etc. Usually people cannot achieve both cost-efficiency and convenience. For example, the fixed-route transit (FRT) systems are considered to be cost-efficient because of their ridesharing attribute and sufficient loading capacity. However, they are inconvenient because the fixed stops and schedule cannot cater to individual passenger's demand. This lack of flexibility is the most significant constraint of FRT. At the other end of the spectrum, private cars and taxis (demand-responsive transit, or DRT more generally) are much more flexible and faster offering convenient door-to-door transportation, at a much higher cost than FRT. Among the transportation modes, ridesharing (also referred to as carpooling) lies somewhere between the two ends. Ridesharing combines the fast travel time and convenience of private cars and the cost-efficiency of fixed-route transit to provide an attractive and viable alternative.

Ridesharing by definition occurs when travelers share both a private vehicle and the associated travel cost with others that have similar itineraries and compatible time schedules. On the one hand, the popularity of smart phones and other ubiquitous computing powers make the efficient sharing and communication of personal information possible (e.g., location through global positioning systems (GPS)). On the other hand, ridesharing arises as a viable urban transportation option in the context of finite oil supplies, rising gas prices, never-ending traffic congestion, and

environmental concerns. The aim of ridesharing is to improve the efficiency of transportation by bringing together travelers with similar itineraries and schedules. The following facts make ride-sharing services promising. The private car occupancy rates are surprisingly low: According to recent reports (see European Environmental Agency, 2005; Santos et al., 2011), the average car occupancies in Europe and the US range from 1.8 to 1.1, meaning the vast majority of the trips are actually transporting "empty seats". The low-occupancies, together with the large demand for automobile transportation at peak-hours lead to traffic congestion in many urban areas. According to recent reports by the Texas A&M Transportation Institute (Schrank and Lomax, 2007; Schrank et al., 2012), the economic loss associated with congestion is as high as 78 billion and 121 billion dollars in 2007 and 2012, respectively, indicating a significant increase over years. Besides, private automobiles are also a major source of fuel consumption and carbon dioxide emissions, which contribute to air pollution and climate change.

Effective usage of ride-sharing can potentially increase occupancy rates, and thus mitigate the above-mentioned problems (Morency, 2007). Moreover, the ridesharing system has "scale effects." As shown by Dailey et al. (1999), the relationship between the number of ridesharing participants and the number of carpools formed is quadratic, meaning that ridesharing can have a large effect on traffic demand management (TDM) if large segments of the population are attracted to the service.

There are already some successful stories in the emerging ridesharing market. Ridesharing market leader, Uber, who started as a "glorified" taxi company whose motto was "Everybody's Private Driver," is currently shifting its focus to ridesharing services in major markets. Recent data shows that half of all Uber rides in San Francisco are for UberPool, which is Uber's ridesharing service. Other players in the market, such as Lyft and Sidecar share the same trend.

Although ridesharing as a transportation solution is promising in a lot of ways, the operations of ridesharing systems introduce new challenges to both industry and academia. First, an important objective of a ridesharing system is to minimize the system-wide total vehicle mileage by gathering travelers with similar itineraries and compatible schedules. Based on ride sharer information, the system should be able to make decisions regarding which travelers will be assigned the role of a driver and what route the drivers should follow regarding picking up and dropping off riders. An efficient ridesharing system should be able to make these decisions automatically with the objective of minimizing the system-wide total vehicle mileage. However, this is a very difficult optimization problem to solve. Second, as the travel costs are reallocated among the customers, a ridesharing system needs to carefully design a fair cost allocation scheme to ensure customer satisfaction and induce more travelers to participate in the system. This problem corresponds to mechanism design in game theory and again, this is a very challenging problem because the calculation of the fair cost allocation scheme is highly related to the ridesharing optimization problem. Neither of the above problems can be solved by existing methodologies. As a result, this dissertation develops innovative mathematical models and algorithms to handle these challenges, in the hope of making ridesharing a sustainable and attractive alternative to private cars.

## 1.2 Outline

This dissertation is organized as the following. Section 2 gives a comprehensive review of three subjects: (1) the models and solution methods for routing and scheduling problems in transportation, (2) models, applications and the algorithmic aspects of game theory, and (3) recent progress in ridesharing study. Section 3 summaries the problems in optimization and mechanism design for ridesharing services

and provides some background in game theory and mechanism design. Section 4 gives a formal definition of the ridesharing optimization problem, characterizes its complexity and develops models and algorithms to solve the problem, both accurately and approximately. Experiments are conducted to evaluate the quality of the developed algorithms. Section 5 is dedicated to the mechanism design problem of ridesharing. The ridesharing cost allocation problem is formulated through the lens of cooperative game theory. Several solution concepts about "fairness" in cost allocation are investigated. In particular, an algorithm is developed to find the nucleolus of the ridesharing game. Conclusions and future research direction are presented in Section 6.

# 2. LITERATURE REVIEW

## 2.1 Mechanism Design

Mechanism design lies in the intersection of computer science and economics. It looks for overall good solutions in a distributed system where each participant acts as a self-interested agent with private preferences. Recently mechanism design has been successfully applied in many areas such as electronic market design and resource allocation problems. A typical goal in mechanism design is to provide incentives to participants to promote truth revelation from agents.

In one sentence, mechanism design is the inverse of game theory: It is the art of designing a game (a mechanism) in which a given desired outcome is achieved in equilibrium, i.e., the state from which no player has the incentive to deviate.

Mechanism designers have many "desired outcomes" to choose from. However, most of the mechanisms are focused on two objectives:

1. In social welfare-maximizing (i.e., efficient) mechanism design, the goal is to assign the items to the bidders with the highest value.

2. In revenue-maximizing (i.e., optimal) mechanism design, the goal is to achieve as much revenue as possible for the agency in charge of creating and awarding bids, i.e., the auctioneer.

Vickrey (1961), in his seminal paper, gave an efficient mechanism for a single-item auction: Assign the item to the bidder with the highest bid, and charge him the second highest price. Naturally, this kind of auction is coined as the second-price or Vickrey auction. A Vickrey auction has two desirable properties: It achieves social welfare optimality and at the equilibrium, all bidders give their true values for the

item. For its simple allocation and payment rules, the Vickrey auction is easy to implement.

Groves mechanisms (Groves, 1973) have played and will continue to play an important role in mechanism design. Groves mechanisms can be implemented to maximize the total value over all participants — meaning Groves mechanisms are efficient. Besides, the Groves mechanisms are strategy-proof, meaning truth-revealing is a dominant strategy for each participant, regardless of the strategies taken by other players. However, the value-maximization problem can only be solved after all the participants report their complete information. Because of its combinatorial nature, the Groves mechanism is computationally intractable. And any attempts trying to increase its computability (e.g., relax the requirement of complete information revelation) could easily compromise the mechanism's game-theoretic properties (e.g., the strategy-proofness). A step up is the Vickrey, Clarke and Groves (VCG) mechanism (Vickrey, 1961; Clarke, 1971; Groves, 1973), which is generalized from the Vickrey auction for multiple items. The VCG mechanism charges each bidder the harm they cause to other bidders (i.e., their negative externality) and is both efficient and incentive compatible. However, implementing VCG mechanism is computationally infeasible (see Nisan and Ronen, 2000). The computational issues of mechanism design has been gaining the attention of theoretical computer scientists since Nisan and Ronen (1999) opened up the field of algorithmic mechanism design. Algorithmic mechanism design addresses the mechanism design problem from a computer scientist's perspective in which worst-case analysis and approximation techniques are used. Recent progress has been focused on the gap between approximability and incentive compatibility (see Papadimitriou et al., 2008; Dobzinski, 2011; Dobzinski and Vondrak, 2012; Dughmi et al., 2011)

For optimal mechanism design, which is focused on revenue-maximization, My-

erson (1981) in his landmark paper solved the auction design problem to achieve the highest expected revenue at equilibrium. Inspired by the results of Myerson (1981) showing that the optimal single-item auction for players with independent private values is simply the Vickrey auction with reserve prices, computer scientists followed up by generalizing the auction settings. These efforts include Chawla et al. (2007) and Hartline and Roughgarden (2009) who relaxed the assumptions on bidders' value distribution and proposed VCG-based mechanisms that reasonably approximated the optimal expected revenue. Unlike mechanism design problems, whose results are dependent on the distribution of bidders' valuations, prior-free auctions further relax the settings and assume no distribution over bidders' valuations before the bid. Worst-case approximation analysis is commonly seen in prior-free auction design problems and the resultant revenues are often compared to the well-known benchmark mechanism (Goldberg et al., 2006). Hartline and Roughgarden (2008) proposed a Bayesian optimal mechanism-based framework that is near-optimal and prior-free. Leonardi and Roughgarden (2012) designed a prior-free multiple goods auction with ordered bidders and gave its expected revenue compared with some benchmarking results.

Note that most of the optimal mechanism design research in the literature was focused on single-item auctions due to the complexity issues. This is one of the challenges we'll face in the mechanism design for ridesharing services because in the ridesharing context we generally have multiple auctioneers (drivers) with multiple items (passenger seats) for sale. Another challenge is due to the impossibility of finding an efficient mechanism which is both incentive compatible and individually rational without external subsides (Myerson and Satterthwaite, 1983).

The most business-wise successful application of mechanism design might be the generalized second-price (GSP) mechanism. The GSP mechanism is the current in-

dustry standard for search engine (e.g., Google, Yahoo!) advertisement allocation as it practically generates stable allocation results. This mechanism was first proposed in a static framework in which a single auction with a small number of slots were allocated at the same time. Varian (2007) and Edelman et al. (2007) independently developed this mechanism and analyzed its equilibrium. Note that unlike VCG mechanism, the GSP is not truthful, meaning truth-revealing is not a dominant strategy for players.

## 2.2 Mechanism Design in Ridesharing

This research area has been largely ignored by the transportation research community until very recently. The author only found two applications of mechanism design in ridesharing. One is due to Kamar and Horvitz (2009) who proposed the agent-based carpooling (ABC) system that dynamically generates ride-share plans and encourages fair payments based on a VCG mechanism. However, this VCG mechanism needs to compute optimal outcomes to ensure truthfulness, and the problem itself is NP-complete (Nisan and Ronen (2000)). Thus, it is computationally infeasible to implement such a mechanism in a dynamic environment in which frequent computation on payments is necessary. To address the intractability, the ABC system makes a series of approximations such as calculating local VCG payments instead of global VCG payments and calculating the payments based on an approximation method used by Nisan and Ronen (2000). In this way however, their payment mechanism is not truthful.

Kleiner et al. (2011) proposed an auction scheme for dynamic ride-sharing with one-driver-one-passenger setting. They showed that their system is incentive-compatible and allows trade-off between the minimization of vehicle mileage traveled and the overall successful matching rate. However, their work needs to be extended to

multiple-passenger-multiple-driver before its can be implemented in the real world. More recently, Wang (2013) studied the stability issue in the optimal ridesharing matching problem. To the extent of our knowledge, no previous work has been published on the general ridesharing problems with a multiple-passenger-multiple-driver-flexible-role setting.

## 2.3 Algorithmic Studies on Ridesharing-related Problems

As will be shown later, the mechanism design problem is closely related to the ridesharing optimization problem with specific objective functions. So there's a need to conduct a literature review on algorithmic studies related to rideharing problems. The ridesharing problem is closely related to the pickup and delivery problem (PDP, see Savelsbergh and Sol (1995) for a review). Extensive studies of PDP can be found in the literature. Many of the studies involve integer programming-based exact algorithms. Sexton and Bodin (1985) reported an exact algorithm based on Bender's decomposition. Cordeau (2006) developed an MIP formulation for a related problem of PDP – the multi-vehicle Dial-a-Ride Problem (DARP). A branch-and-cut algorithm using valid inequalities was proposed by him. Cordeau and Laporte (2003) examined and compared different mathematical formulations and solution methods. Lu and Dessouky (2004) developed an MIP formulation for the multiple-vehicle PDP. New valid inequalities were utilized to develop a branch-and-cut algorithm to optimally solve the problem. Cortes et al. (2010) presented a strict MIP formulation for the PDP and allow passengers to transfer. Berbeglia et al. (2010) gave a comprehensive review on dynamic PDP and discussed solution strategies. Quadrifoglio et al. (2008) developed an MIP formulation for the static single-vehicle Mobility Allowance Shuttle Transit (MAST) system (a variant of the PDP system). Logic cuts were proposed by the authors to strengthen the formulation and solve the problem.

Other exact algorithms include dynamic programming. Psaraftis developed dynamic programming techniques to solve the DARP (Psaraftis, 1980) and DARP with time windows (Psaraftis, 1983). These two algorithms have a complexity of $O(N^2 3^N)$ (here $N$ stands for the number of customers) and can solve instances of up to 20 customers within a reasonable time. Besides, Teodorovic and Radivojevic (2000) developed a fuzzy logic approach for the DAR problem. Wang (2013) modeled the single-driver, single-rider ridesharing optimization problem as a maximum-weight bipartite matching problem.

Due to the fact that PDP is strongly $\mathcal{NP}$-hard (Lenstra and Kan, 1981), besides exact solution methods, the research community have been focusing on heuristic approaches that can solve large instances of PDP in polynomial time, while maintaining the quality of the solution. Insertion heuristics are very important and popular among the heuristic approaches, not only because of their fast running time, but also due to their applicability and implementability in dynamic environments (Campbell and Savelsbergh, 2004). Efforts in insertion heuristics include a insertion-based construction heuristic for multiple-vehicle PDP by Lu and Dessouky (2006). One disadvantage of insertion heuristics is that it is difficult to quantitatively evaluate their performance. Another disadvantage is due to its myopic and greedy nature when searching for local optimum. Quadrifoglio et al. (2007) resolved this disadvantage efficiently by introducing an insertion heuristic with the concept of "usable slack time". Some efforts have been put to evaluate the performance of heuristics through worst-case analysis. These efforts can be found in PDP and more fundamental problems such as TSP. Through constructing a minimum spanning tree and an Euler tour, Christofides (1976) developed a $O(n^3)$ heuristic with worst-case ratio of 3/2 for metric-TSP (TSP whose cost matrix satisfies the triangle inequality). Rosenkrantz et al. (1977) analyzed the approximation ratio of the cheapest insertion

heuristic and several other heuristics for TSP. Archetti et al. (2003) did research on the re-optimization version of TSP in which a new node is added to or removed from an optimal solution. They proved that the worst-case ratio of the cheapest insertion heuristic decreases from 2 (Rosenkrantz et al., 1977) to 3/2 when applied to the re-optimization version of TSP. Arora (1998) developed a polynomial time approximation scheme (PTAS) for Euclidean TSP, which is currently the best result on approximating TSP.

# 3. DEFINITIONS AND PROBLEM DESCRIPTION

We begin this section by summarizing the challenges of optimization and mechanism design for ridesharing:

1. The valuations of the customers largely depend on the actual assignment of the ridesharing plan.

2. The mechanism design problem is closely related to the optimization problem of ridesharing. For example, a VCG-based mechanism has its desired properties (efficiency, individually-rational, and strategy-proof) only if the ridesharing assignment solved from the optimization model is actually optimal.

## 3.1 Ridesharing Settings

There are two types of participants in ridesharing: riders and drivers. Drivers use their vehicles to provide ride-sharing offers. Riders place requests to be matched to an offer. A service is a procedure to make those matches happen, which could be a website based on social networks.

There are four basic arrangements of ridesharing services, namely:

- Single Rider, Single Driver
- Single Driver, Multiple Riders
- Single Rider, Multiple Drivers
- Multiple Driver, Multiple Riders

This research will focus on the multiple-driver-multiple-rider setting. The following notation is commonly used to address the optimization matching problem of ridesharing.

Given a set of locations $L$, travel times $t_{ij}$ and distances $d_{ij}$ between each pair of locations $i, j \in L$, let $D$ be the set of drivers; $R$, the set of riders, and $P = D \cup R$, the set of all agents in a ridesharing system. Each driver $d \in D$ (rider $r \in R$) wants to travel from his origin (start location) $l_{d,s} \in L$ ($l_{r,s} \in L$) to his destination (end location) $l_{d,e} \in L$ ($l_{r,e} \in L$). Each driver $d \in D$ (rider $r \in R$) has an earliest time $et(d)$ ($et(r)$) from which she may leave her origin and a latest time $lt(d)$ ($lt(r)$) at which she may arrive at his destination. The range $[et(d), lt(d)]$ is called the time window.

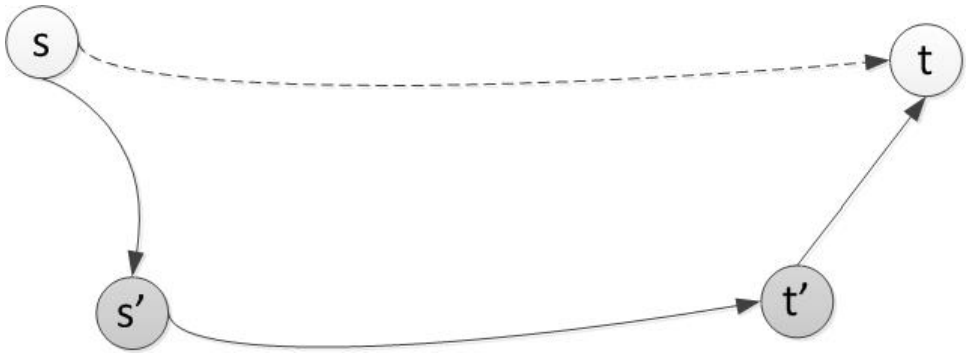



Figure 3.1: A matching with detour

Figure 3.1 illustrates the detouring cost associated with a successful ridesharing matching. The original cost of the driver without ridesharing with the rider is: $d_{st}$. The cost of the rider is $d_{s't'}$. With ridesharing, the joint trip length becomes $d_{ss'} + d_{s't'} + d_{t't}$. From the driver's point of view, the detouring cost is thus $d_{ss'} + d_{s't'} + d_{t't} - d_{st}$. Since the driver is always suffering a loss equal to the detouring cost, the runner of the service has to pay the drivers for their loss. The business runner can use the fees collected from the riders to subsidize the drivers. From the perspective of the business runner, an important issue of the mechanism design is

how to subsidize the driver to attract more users to participate in the service, and at the same time, maximize the profit of ridesharing matching.

### *3.1.1 Ridesharing system objective*

The objective of the system is to minimize the system-wide mileage, which is the total mileage traveled by all users. This objective is meaningful from a social perspective because total vehicle mileage is critically related to air pollution (emissions) and road congestion. Also, note that this objective is closely related to minimizing total travel costs, or alternatively speaking, maximizing total travel cost savings, which is the direct motivation of ridesharing participants.

## 3.2 Mechanism Design Problem

The mechanism designs how much each participant has to pay for the shared ride. We design mechanisms that have the following properties: For drivers, cost is shared and perhaps, some income is gained; for passengers, their transportation needs are satisfied without driving their own cars.

### *3.2.1 Value of a shared-ride for an agent*

Let $P$ be the set of participants in a ridesharing system and $S \in P$ be a ridesharing group. Denote by $\mathcal{R}(S)$ the set of all the feasible ridesharing tours for $S$. Here a ridesharing tour $R \in \mathcal{R}(S)$ decides which participant in $S$ will be assigned as the driver and gives the pick-up and drop-off sequence for all the riders. The value of a shared-ride for an agent $p_i \in S$ is defined as the cost savings associated with switching from driving a vehicle from origin to destination to participating in a ridesharing tour. Let $C^0(p_i)$ be $p_i$'s cost for an individual trip if he doesn't participate in any ridesharing. Let $C(p_i, R)$ be the cost of $p_i$ for ridesharing tour $R$, then $p_i$ puts a

value on ridesharing tour $R$ that is equal to

$$v_i(R) = C^0(p_i) - C(p_i, R)$$

It is noted that the cost of agent $i$ should include the following components:

1. Agent $i$'s driving cost including fuel cost and vehicle usage.

2. A cognitive inconvenience cost for being a driver.

Note that the fuel cost and the vehicle usage is closely related to the driving miles. It is also observed that agent $i$'s cognitive inconvenience cost for being a driver can also be measured by the mileage he drives, as the inconvenience cost should increase as the driving miles accumulate. As a result, it is reasonable to assume that each participant $i$'s travel cost is proportional to his driving miles. Since in any ridesharing tour $R \in \mathcal{R}(S)$, the only one that is driving is the driver $p_d \in S$, then we have

$$C(p_i, R) = 0, \quad \forall p_i \in S \backslash \{p_d\}.$$

Then it is not hard to see that for every agent who is not matched to any ridesharing tour, his value is

$$v_i(R) = C^0(p_i) - C(p_i, R) = C^0(p_i) - C^0(p_i) = 0$$

The cumulative value of a ridesharing tour $R$ is the summation of the values of all the agents for participating in the ridesharing.

$$V(R) = \sum_{p_i \in S} v_i(R)$$

### *3.2.2 Mechanism design preliminaries*

Before getting into the mechanism design for ridesharing, we first introduce some definitions for the desired properties of the ridesharing mechanism.

**Definition 3.1** (Truthfulness)**.** *A mechanism is truthful if any agent's equilibrium strategy is to report his true value for shared-ride plan $R$ to the system,*

$$\hat{v}_i(R) = v_i(R), \quad \textit{for } \forall p_i \in P$$

**Definition 3.2** ((Ex post) Individual Rationality)**.** *A mechanism is ex post individual rational if no agent loses money by participating in the ridesharing,*

$$v_i(R^*) \geq \rho_i, \quad \textit{for } \forall p_i \in P,$$

*where $R^*$ is the equilibrium ridesharing plan and $\rho_i$ is the value of $p_i$ if there is no ridesharing.*

**Definition 3.3** (Weak budget-balance)**.** *A mechanism is weakly budget balanced when no external subsidy is required to maintain the operations of the ridesharing system,*

$$\sum_{p_i \in P} \rho_i \geq 0$$

## 3.3 Mechanism Design Problem in Ridesharing

Many online ridesharing matching agencies use simple payment mechanisms such as sharing fuel costs among passengers (see Furuhataa et al., 2013). These simple mechanisms are easy to implement but are not capable of incorporating personal preferences because of individually-different cognitive costs. Moreover, these pay-

ment schemes are not truth-revealing since participants may deceptively report their own value in hopes of biasing the ridesharing plans for their own good.

Designing a payment mechanism that has the game theoretic virtues for a dynamic ridesharing system is challenging. First of all, Myerson and Satterthwaite (1983) proved the impossibility of finding an efficient mechanism that is both incentive compatible and individually rational without external subsides. Second, computationally expensive payment mechanisms that require optimal outcome calculations are not practically implementable in a dynamic ridesharing system.

According to the VCG mechanism (Vickrey, 1961; Clarke, 1971; Groves, 1973), each participant of the ridesharing service is charged the harm he causes to the social welfare (his externality). Let $V_{-i}^{**}$ be the cumulative value of the rideshare plans of all the agents except $p_i$, and $V_{-i}^{*}$ is the cumulative value of the system when $p_i$ is not participating in ridesharing. Then agent $p_i$'s payment is:

$$\rho_i = V_{-i}^{**} - V_{-i}^{*}$$

On the condition that the rideshare plan calculated by the optimization problem is optimal, this VCG-based plan has the following properties:

- Efficiency – Social welfare is maximized. This is achieved by solving the optimization problem.
- Individual rationality – All agents have non-negative utility by participating in ridesharing.
- Strategy-proof – Truth-telling is a dominant strategy for each agent regardless of what strategy other players are taking.

There are several challenges when solving the mechanism design problem. First,

payments usually need to be repeatedly calculated in a dynamic ridesharing setting. Second, to ensure truthfulness, the embedded optimization problem, which maximizes social benefits (NP-hard in its nature) needs to be solved.

# 4. RIDESHARING OPTIMIZATION PROBLEM

## 4.1 Introduction

In a large-scale ridesharing system, we assume that all the customers that are considered in the optimization problem are time-compatible. This assumption is realistic when the time windows are sufficiently large and travelers are less sensitive to time windows (e.g, end of business days). Furthermore, the proposed optimization problem has a nice sub-structure. That is, it can be easily divided into smaller subproblems with similar structure whose customers share similar time schedule and solved in a parallel algorithm fashion.

### *4.1.1 Ridesharing system objectives*

The objective of the ridesharing organizer is to minimize the system-wide mileage, which is the total mileage traveled by all users. This objective is meaningful from a social perspective because total vehicle mileage is critically related to the air pollution (emissions) and road congestion. Note that this objective is also closely related to minimizing total travel costs, or alternatively speaking, maximizing total travel cost savings, which is the direct motivation of ridesharing participants. And we will later show that this system-wise objective has an alignment with individual participant's interest - minimizing personal travel cost.

### *4.1.2 Problem definition*

Let $P$ be the set of participants in a ridesharing system. Each participant $i \in P$ wants to travel from his origin $s_i$ to his destination $t_i$. Let $V_s = \{s_i | i \in P\}$, $V_t = \{t_i | i \in P\}$, and the entire location set $V = L_s \cup L_t$. Let $A = V \times V$ denote the edge set connecting all the vertices in $V$ and $C \in \mathbb{R}^{|V| \times |V|}$ denote the cost matrix

with $c_{ij}$ representing traveling cost from location $i$ to $j$. Then we have a complete graph $G = (V, A)$ and its edge cost matrix $C$ as input. To formally introduce the ridesharing optimization problem, we first present some definitions.

**Definition 4.1** (ridesharing tour)**.** *Let $S \subseteq P$ be a ridesharing group and let $V(S)$ denote the location set of customers in $S$, therefore $V(S) = \cup_{i \in S}(s_i \cup t_i)$. A ridesharing tour for ridesharing group $S$, $R(S)$ is a directed Hamiltonian path on the graph $G(S) = (V(S), A(S))$ where $A(S) = V(S) \times V(S)$ such that*

1. *$R(S)$ starts from agent $d$'s origin $s(d)$ and ends at $d$'s destination $t(d)$.*

2. *Let $S_{-d} = S \backslash \{d\}$. For every agent $i$ in $S_{-d}$, $s(i)$ precedes $t(i)$.*

Note that the above definition implies that agent $d$ is assigned as the driver in the ridesharing group $S$.

**Definition 4.2** (ridesharing partition)**.** *A ridesharing partition $SP = \{S_1, \ldots, S_m\}$ is a set of ridesharing groups such that*

1. *$\cup_{S_j \in SP} S_j = P$*

2. *$S_j \cap S_k = \emptyset \quad 1 \leq j \neq k \leq m$*

**Definition 4.3** (ridesharing plan)**.** *A ridesharing plan for a ridesharing partition $SP$ is a set of ridesharing tours $RP = \{R(S_j) | S_j \in SP\}$*

Define $f(RP)$ as the value of ridesharing plan $RP$ that corresponds to a function $f$. Define the objective function of the ridesharing optimization problem as:

$$max\{f(RP)\}$$

In this research $f$ is the accumulated social values of all the ridesharing tours, which is defined by:

$$f(RP(SP)) = \sum_{S_j \in SP} V(R(S_j)) \tag{4.1}$$

**Definition 4.4** (ridesharing optimization problem (RSP)). *An optimization problem of ridesharing is a 4-tuple $\langle I_Q, S_Q, f_Q, opt_Q \rangle$, where:*

- *$I_Q$: the set of the participants $P$ and the corresponding graph $G = (V, A)$*
- *$S_Q$: the set of all ridesharing plans for all the ridesharing partitions of $P$*
- *$f_Q$: $f(RP)$ is the value of ridesharing plan $RP$*
- *$opt_Q$: max.*

### *4.1.3 Cost and value of a shared-ride for an agent*

Recall from Section 3, for every agent who is not matched to any ridesharing tour, his value is

$$v_i(R) = C^0(p_i) - C(p_i, R) = C^0(p_i) - C^0(p_i) = 0$$

On the other hand, for an agent that is matched in a ridesharing tour, his value is

$$v_i(R) = \begin{cases} C^0(p_i), & \forall p_i \in S \backslash \{p_d\}, \\ C^0(p_i) - d(R), & p_i = p_d. \end{cases}$$

Here $d(R)$ is the tour length of $R$.

The cumulative value of a ridesharing tour $R$ is the summation of the values of

all the agents for participating in the ridesharing.

$$\begin{aligned} V(R) &= \sum_{p_i \in S} v_i(R) \\ &= \sum_{p_i \in S} C^0(p_i) - d(R) \\ &= \sum_{p_i \in S} d(p_i) - d(R) \end{aligned}$$

Given a set of ridesharing group $S \subseteq P$ and suppose the ridesharing organizer wants to maximize the cumulative value of $S$. Since $d(p_i)$ is a constant for every $p_i \in S$ when $S$ is given, this objective is equivalent to minimizing $d(R)$. And minimizing $d(R)$ is equivalent to finding the optimal solution for the corresponding TSP on $S$ with pick-up and drop-off precedence constraints. We call the optimal solution for this problem the optimal ridesharing tour of $S$ and denote it as $C^*(S)$. The associated value of $C^*(S)$ is called the optimal value of $S$ and is denoted by $V^*(S)$.

Now suppose the given set of ridesharing participants $P$ has formed $m$ ridesharing groups. Let $SP = \{S_1, \ldots, S_m\}$ denote the set of these groups. Then we must have $\cup_{S_j \in SP} S_j = P$ and $S_j \cap S_k = \emptyset$ for every $1 \leq j \neq k \leq m$. Note that here $SP$ is also known as a partition of $P$. We define the cumulative value of the participant set $P$ under a partition $SP$ as the sum of the optimal values of sets in $SP$, that is

$$V_{SP}(P) = \sum_{S_j \in SP} V^*(S_j)$$

Now suppose the ridesharing organizer is more ambitious, not only it wants to maximize $V(R)$ for a given ridesharing group $S \subseteq P$, instead it aims to maximize the cumulative value of $P$. This objective is equivalent to find the optimal set partition

$SP^*$ such that

$$SP^* = \arg \max_{SP_i \in \mathcal{SP}(P)} V_{SP_i}(P)$$

## 4.2 Modeling

### *4.2.1 Setting*

The ridesharing organizer seeks to minimize system-wide total vehicle traveling miles, namely the total driven miles by all the ridesharing users, either in a shared ride or in a solo drive if unmatched.

Unlike cab-drivers that are hired by a company, drivers in a ridesharing service are independent users. Suppose we have $n$ ridesharing participants $P = \{1, 2, \ldots, n\}$. Each of them has a origin location and a destination location. Denote the node set of origin and destination locations as $V_O$ and $V_D$, respectively. Let node $i$ be customer $i$'s origin node ($1 \leq i \leq n$) and $i+n$ be his destination node, then we have $V_O = \{1, 2, \ldots, n\}$ and $V_D = \{n+1, n+2, \ldots, 2n\}$.

Then we have a complete digraph $G_R = (V_R, A_R)$, where $V_R = V_O \cup V_D$ is the set of all nodes and $A_R = V_R \times V_R$ is the set of all edges. Let $C_R \in \mathbb{R}^{2n \times 2n}$ be the cost matrix with $c^0_{ij}$ representing the travel cost from node $i$ to node $j$.

Figure 4.1 provides an illustration showing two feasible solutions to RSP on $G_R$. Note that the number in a node indicates its associated customer index. Nodes with a rectangular shape and a "+" label represent the destinations. Figure 4.1(a) is a solution that consists of individual trips (no ridesharing at all). On the other hand, in Figure 4.1(b) customers 1 and 2 form a ridesharing group and customers $4, 5$ and 6 form a ridesharing group. Nodes in red belong to the customers that are assigned as drivers.

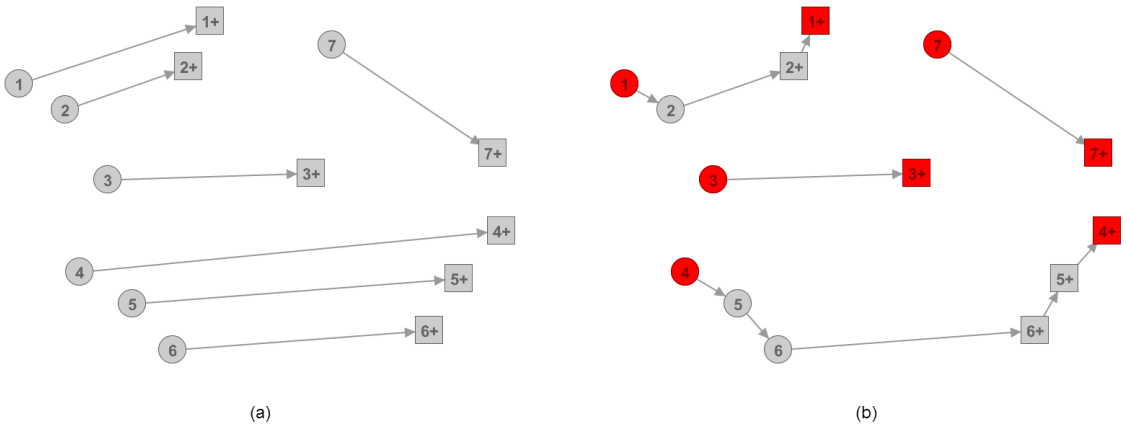



Figure 4.1: (a) Individual trips vs. (b) organized ridesharing trips

### *4.2.2 Hardness*

**Theorem 4.1.** *RSP is NP-hard.*

*Proof.* Given a case in the 3-set covering problem, we can construct a RSP case like this. For a given case $P$ in the 3-partition problem, where $U = \{e_1, e_2, \ldots, e_n\}$ is the finite element set, and $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ is the collection of sets of $U$. Each $S_j \in \mathcal{S}$ is associated with a cost $c(S_j)$. Note that this includes singletons. For any $\mathcal{X} \subseteq \mathcal{S}$, let $C(\mathcal{X})$ denote the total costs of the sets in $\mathcal{X}$, i.e., $C(\mathcal{X}) = \sum_{S_j \in \mathcal{X}} c(S_j)$. The objective function of problem $P$ is to find the $\mathcal{X} \subseteq \mathcal{S}$ with smallest cost that covers each element exactly once. It is known that this problem is NP-hard (Karp, 1972), and $P$ can be reduced to RSP in the following way.

For any given case of $P$, we construct a case of RSP $P_0$ like this. We build a graph $G = (V, E)$ where $V = U + U'$, here $U$ is still the finite element set as in $P$, and $U'$ is the set formed by corresponding "destination" of each element in $U$. For every element $e \in U$, we have a corresponding destination as $e'$, and we set the distance $c_0(ee') = c(e)$. For every duplet, triplet and above in $\mathcal{S}$, that we denote it by $S$ in P,

in $P_0$ we have $c_0(S) = c(S)$. Here $c_0(S)$ denotes the cost of the optimal ridesharing tour (as defined before) of the nodes in $S$. Because a solution to the RSP on $G$ ($P_0$) includes a solution to the optimal set covering problem on $U$ ($P$), and the latter is NP-hard, we can conclude that RSP is NP-hard as well. □

### *4.2.3 Transformation*

The RSP is transformed to the single-depot multiple traveling salesman problem with pickup and delivery constraints (SDMTSP-PD) in the following manner. Let $V_0 = \{0\}$ be a "dummy" depot. The transformed graph is represented by $G = (V, A)$ where $V = V_R \cup \{V_0\}$ and $A$ is the set of all the directed edges connecting any two vertices in $V$. The cost of the arcs in $A$ is defined as

$$c_{ij} = \begin{cases} 0, & \text{if } i = 0 \text{ or } j = 0, \\ c^0_{ij}, & \text{otherwise.} \end{cases}$$

Then the solution in Figure 4.1(b) is equivalent to the solution in Figure 4.2, where dash lines indicate zero-cost arcs. Since SDMTSP-PD is NP-hard, RSP is NP-hard too.

### *4.2.4 Integer program*

In this section, we introduce an integer program for the transformed SDMTSP-PD problem.

For each edge $(i, j) \in A$ we define a binary variable $x_{ij}$ such that

$$x_{ij} = \begin{cases} 1, & \text{if } (i, j) \in A \text{ is in the solution,} \\ 0, & \text{otherwise.} \end{cases}$$

Then we can immediately perform some reductions on the problem.

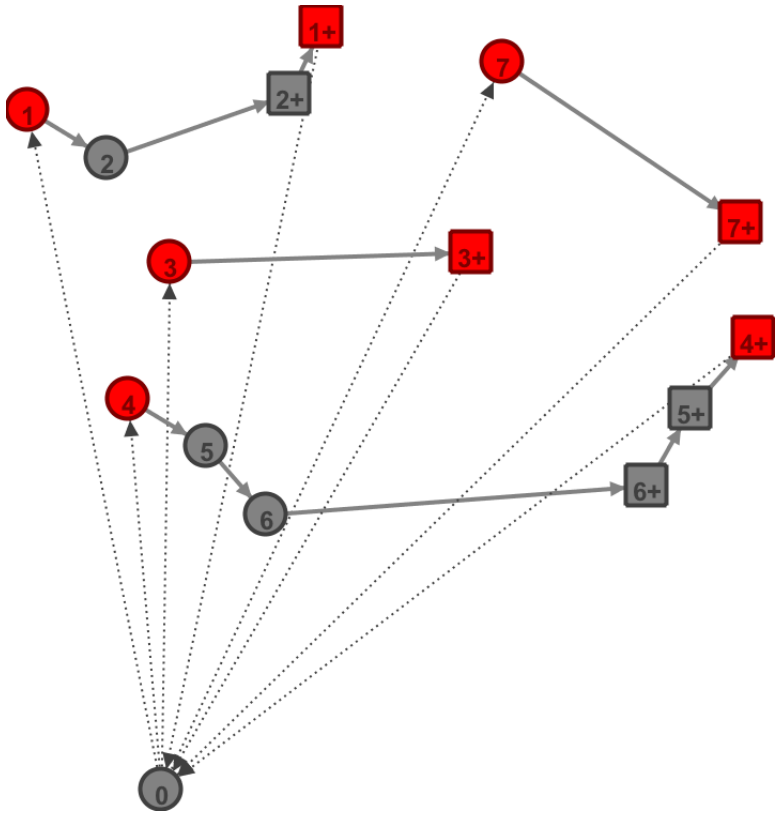



Figure 4.2: Corresponding feasible solution for SDMTSP-PD on the transformed graph

**Lemma 4.1.** *Given* $G = (V, A)$ *and* $P = (V, R)$*, then*

$(I) \quad x_{0j} = 0 \quad \forall (i,j) \in R$

$(II) \quad x_{i0} = 0 \quad \forall (i,j) \in R$

$(III) \quad x_{ji} = 0 \quad \forall (i,j) \in R$

*Proof.* Obvious. □

Additionally, we define a binary variable $y_{ik}$ as follows

$$y_{ik} = \begin{cases} 1, & \text{if node i is visited by driver k,} \quad i \in V \backslash \{0\}, k \in V_O \\ 0, & \text{otherwise.} \end{cases}$$

Note that here $y_{ik} = 1$ implies customer $i$ is a driver. Thus constraints that connect $y$ variables and $x$ variables need to be added in the integer program.

$$\min \quad Z = \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{4.2}$$

$$\sum_{i=0}^{2n} x_{ij} = 1, \quad j = 1, 2, \ldots, n \tag{4.3}$$

$$\sum_{j=0}^{2n} x_{ij} = 1, \quad i = 1, 2, \ldots, n \tag{4.4}$$

$$x_{0i} - x_{(i+n)0} = 0, \quad i = 1, 2, \ldots, n \tag{4.5}$$

$$u_i - u_j + px_{ij} \leq p - 1, \quad 1 \leq i \neq j \leq 2n \tag{4.6}$$

$$u_i < u_{i+n}, \quad i = 1, 2, \ldots, n \tag{4.7}$$

$$\sum_{k} y_{ik} = 1, \quad i = 1, \ldots, 2n; \quad k = 1, \ldots, n \tag{4.8}$$

$$y_{ik} = y_{(i+n)k}, \quad i = 1, \ldots, n; \quad k = 1, \ldots, n \tag{4.9}$$

$$x_{0i} = y_{ii}, \quad i = 1, \ldots, n \tag{4.10}$$

$$x_{ij} \leq M_1(1 - z_{ij}), \quad 1 \leq i \neq j \leq 2n \tag{4.11}$$

$$-M_2 z_{ij} \leq y_{ik} - y_{jk} \leq M_2 z_{ij}, \quad 1 \leq i \neq j \leq 2n; \quad k = 1, \ldots, n \tag{4.12}$$

$$x_{ij} \in \{0, 1\}, \quad 0 \leq i \neq j \leq 2n \tag{4.13}$$

$$y_{ik} \in \{0, 1\}, \quad i = 1, \ldots, 2n; \quad k = 1, \ldots, n \tag{4.14}$$

$$z_{ij} \in \{0, 1\}, \quad 1 \leq i \neq j \leq 2n \tag{4.15}$$

Constraints (4.3) and (4.4) are the continuity constraints. Constraints (4.5) make sure that a tour starts at its driver's origin and ends at his destination. Constraints (4.6) are a group of subtour-elimination constraints (SECs) first proposed by Miller

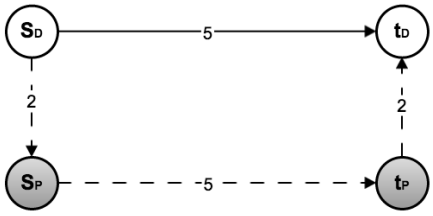
et al. Miller et al. (1960). Here $u_i$ are continuous variables called *node potentials* that indicate the visit order of node $i$ in the tour, while $p$ denotes the maximal number of nodes a driver can visit in a tour. This parameter can be used to specify the seat capability of drivers. Generally in a typical dynamic ridesharing setting where most participating vehicles are private vehicles whose seats are fewer that 5, $p$ won't exceed 10. Constraints (4.7) ensure that a customer's origin precedes his destination. Constraints (4.8) ensure that each node is visited by exactly one driver. Constraints (4.9) make sure that a customer's origin and destination are visited by the same driver. Constraints (4.10) mean that customer $i$ is selected as a driver if and only if his origin is visited by himself. The intuitive meaning of constraints (4.11) and (4.12) is that if edge $(i, j)$ is selected in the solution then nodes $i, j$ must be served by the same driver. This set of constraints serve as a bridge between $x$ variables and $y$ variables. Here $M_1$ and $M_2$ are large numbers.

## 4.3 Solution Methods

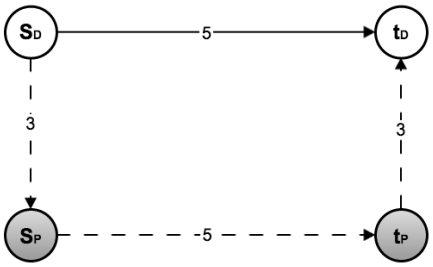
### *4.3.1 Profitability of ridesharing*

Since the ridesharing problem aims to combine trips with similar itineraries together to save total travel cost, a natural question to ask is, when does ridesharing makes the greatest economic sense, and thus should be encouraged, and in what case should inefficient ridesharing be discouraged. The following example illustrates this thought.

In Figure 4.3(a), the total travel cost assuming no ridesharing happens is $d_{S_D t_D} + d_{S_P t_P} = 10$. When $P, D$ decide to do share a ride, the optimal route yields a travel cost of 9 (e.g. $S_D - S_P - t_P - t_D$), resulting a cost saving equal to 1. In Figure 4.3(b) however, the optimal route has a travel cost of 11, resulting a cost loss equal to 1 instead of cost saving.




(a) Profitable ridesharing




(b) Nonprofitable ridesharing

Figure 4.3: A two-player example

From this example we know that, the profitability of ridesharing depends largely on the relative geographical location of the participants. To generalize our observation, we conclude that, for a two-player ridesharing to be profitable, the combined route cost must be less than the sum of the two solo trip costs. That is, $d_{S_D S_P} + d_{t_D t_P} + d_{S_P t_P} < d_{S_D t_D} + d_{S_P t_P}$, i.e. $d_{S_D S_P} + d_{t_D t_P} < d_{S_D t_D}$. This is equivalent to say, the cost saving of ridesharing, $d_{S_D t_D} - (d_{S_D S_P} + d_{t_D t_P})$, has to be greater than 0.

### *4.3.2 One-to-one match*

Recall that the objective of minimizing total travel cost is equivalent to maximizing total cost saving. The above observation motivates Wang (2013) to solve an

alternate version of ridesharing optimization problem: the optimal rideshare matching problem. In the optimal rideshare match problem, each customer can be matched with at most another customer to form a shared-ride. Obviously, the solution to this problem must have a greater cost than the solution to the ridesharing optimization problem. Nevertheless, the solution to this problem can provide insight to solving the ridesharing optimization problem and can serve as a benchmark to our proposed heuristic solution methods.

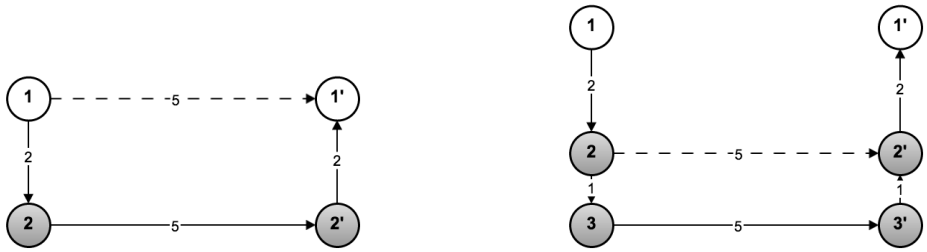
The optimal ridesharing matching problem (see Wang, 2013) can be modeled on a graph. Let $G = (V, A)$ be a digraph with $V = \{1, \ldots, N\}$ standing for the set of customers and $A = \{(i, j)|i, j \in V\}$ representing the possible rideshare match between agent $i$ and $j$. A directed arc $(i, j), \forall i, j \in V$ is associated with an edge cost $c_{ij}$ equal to the cost saving when $i$ serves as the driver and $j$ as the rider in the $i - j$ rideshare match. The objective function of this problem aims to maximize the cost savings of rideshare matches over all possibilities: $\sum_{(i,j)\in A} c_{ij}x_{ij}$. Here $x_{ij}$ is a binary decision variable that is defined as

$$x_{ij} = \begin{cases} 1, & \text{if rideshare match } (i, j) \in A \text{ is selected,} \\ 0, & \text{otherwise.} \end{cases}$$

The constraints of this optimization model can be represented by $\sum_{j\in V\setminus\{i\}} x_{ij} + \sum_{j\in V\setminus\{i\}} x_{ji} \leq 1 \quad \forall i \in V$.

### *4.3.3 An insertion heuristic*

Note that the optimal matching solution in Section 4.3.2 is not necessarily the optimal solution to the RSP. Apparently, as long as the vehicle capacity is not reached, the route plan could be further improved by inserting unmatched customers. The following example shows that the further improvement through insertion is possible.




(a) An optimal match (b) Insertion

Figure 4.4: A profitable insertion

The optimal match solution by solving the optimization model in Section 4.3.2 is represented in Figure 4.4(a). This solution can be potentially further improved. As is shown in Figure 4.4(b), when a third customer (customer 3) participates in the system, the total travel cost can be saved by including 3 into the existing route of 1 and 2. The saved cost by including traveler 3 is $d_{22} - d_{23} - d_{2'3'} = 3$.

The observation from Figure 4.4 motivates the researchers to develop an insertion-based heuristic to improve the solution to the ridesharing optimization problem. Insertion heuristic is an efficient algorithm for solving transportation routing and scheduling problems. The basic idea of insertion-based heuristic involves finding the "cheapest" feasible insertion position in an existing route. Jaw et al. (1986) first adapted the traditional insertion approach to solve the multi-vehicle dial-a-ride problem with time windows. Insertion-based constructive heuristic has then been developed to solve the pickup and delivery problem with time windows (Lu and Dessouky, 2006), the single-vehicle mobility allowance shuttle transit service (MAST) scheduling problem (Quadrifoglio et al., 2007), and the multiple-vehicle MAST problem (Lu et al., 2011).

The insertion-based heuristic algorithm we develop is based on the optimal match-

ing solution found in Section 4.3.2. After the optimal matching solution is obtained, for each of the unmatched participant, the algorithm loop through all the existing routes to find a feasible insertion position that has the greatest cost saving. This process repeats until either no unmatched participant is left or an insertion position with positive cost saving cannot be found. The input of the algorithm is the locations of the customers and the optimal matching solution as found in Section 4.3.2. The output of the algorithm is the improved ridesharing solution (a route plan). This insertion heuristic is formally described in Algorithm 4.1.

## 4.4 Experiments

We implemented the algorithms in Java with CPLEX 12.6 and the Concert library. The data set[1] we used in the experiments were selected from Dumitrescu et al. (2010). The origins and destinations of customers were randomly generated in the square $[0, 1000] \times [0, 1000]$. The Euclidean distances were used. The instances are named probnX, where $n$ is the customer size and $X$ stands for different instances with the same customer size. The (partial) experimental results are summarized in Table 4.1.

Both the optimal matching and the insertion heuristic are able to find the solutions instantly (in less than 1 second). So in Table 4.1 we focus on how much time the MIP model spends on finding the optimal solutions. Because RSP is NP-hard, not all prob10 instances can be solved to optimality within the 3600s time limit. For those could not be solved to optimality, the integral gap ranged from 5% to 16%. It is noteworthy to point out that the solution obtained from the insertion heuristic is not far from the optimal solution via solving MIP: for all the prob5X instances and the instance of prob10c, the insertion heuristic actually found the optimal solution;

[1]The data sets can be downloaded from `http://www.diku.dk/~sropke/`

for prob10d, the gap between the insertion heuristic solution and the MIP solution was only 1.64%.

To further compare the solution qualities of insertion heuristic (Insertion) and optimal matching (Match) in terms of cost saving, the average solution values for each problem size are summarized in Table 4.2. The percentage in parenthesis for each solution method indicates the cost saving percentage compared to the non-ridesharing route plan (Solo). It can be seen that Insertion always outperforms Match, and can further save about 5 percent of the total travel cost – a non-trivial amount of mileage.

Table 4.3 summarizes the saved vehicle trips by adopting ridesharing. As can be seen, Match and Insertion can save 20% - 38% and 24% - 55% vehicle trips on road respectively, depending on the problem size. Once again, Insertion always beats Match by a significant margin (as high as 18%). Also note that the percentage of vehicle trips saving increases as the problem size – a fact that confirms the scale effect of ridesharing.

## 4.5 Conclusions

In this section we approached ridesharing problem from the service provider's point of view. Particularly, we try to answer the question that, given the participants' origin and destination, how should the service provider organize (assign driver/rider role and suggest a route) the ridesharing, with the objective of minimizing system-wide travel cost? This problem may well arise in the context of large-scale, dynamic ridesharing. Although ridesharing service provider may face this challenging problem, it has never been studied in the literature. The authors formally defined this problem as the ridesharing optimization problem (RSP) and showed how to transform RSP to the single-depot multiple traveling salesman problem with pickup and

delivery constraints (SDMTSP-PD). A mixed-integer problem (MIP) model was then developed for RSP.

Since RSP is NP-hard, we resort to approximation algorithms to solve the problem with larger size. An insertion-based heuristic is developed based on the optimal matching solution (see Wang, 2013). The insertion heuristic was then compared with optimal matching solution and the optimal solution of RSP MIP. We found that the insertion heuristic can consistently save more mileage than optimal matching, and the gap between the heuristic solution and the MIP optimal solution was very small, thus confirmed the effectiveness of the insertion heuristic. It is also found out that it can save a significantly large amount of vehicle trips by adopting ridesharing. In our experiments, the insertion algorithm can save vehicle trips by up to 55%, putting more than half of the cars on road back to garage. We also found that ridesharing has scale effect, as the percentage of vehicle trip-saving increases as more customers join the ridesharing service.

The researchers identify several future research directions on the ridesharing optimization problem. The special structure of the RSP MIP model is still to be investigated and exploited, thus developing valid inequalities and logic cuts is a promising direction. The insertion heuristic has the potential to be further improved by better and deeper understanding of the characteristics of RSP. We note that the experiments conducted in this section were using totally randomly-generated geographical locations. But this can be far from the real-world scenario in which travelers' origins and destinations are more likely to be clustered. In these situations, the insertion heuristic can potentially perform even better. So it will be interesting to collect real-world data and further evaluate the models and solution methods developed for RSP.

**Algorithm 4.1:** An insertion algorithm based on the optimal matching solution

```
input  : Geolocations of customers and the optimal match solution
output: An improved ridesharing solution
SAVING = 0 ;
while true do
    if UNMATCHED= ∅ then
        break ;
    end if
    SOLUTION = ∅ ;
    SAVING = 0 ;
    for a ∈ UNMATCHED do
        for r ∈ all_routes do
            for each feasible insertion position do
                Δ = cost saving after insertion ;
                if Δ < SAVING then
                    SAVING = Δ ;
                    record the incumbent solution ;
                end if
            end for
        end for
    end for
    if SAVING< 0 then
        update UNMATCHED ;
        update all_routes ;
        update SOLUTION ;
    end if
    else
        break ;
    end if
end while
```

Table 4.1: Experiment results

| Instance | n | Total cost | | | | Time (s) | Gap |
|---|---|---|---|---|---|---|---|
| | | Solo | Match | Insertion | MIP | MIP | |
| prob5a | 5 | 2722.0 | 2338.2 | 2338.2 | 2338.2 | 0.3 | - |
| prob5b | 5 | 2378.4 | 2115.1 | 2115.1 | 2115.1 | 0.5 | - |
| prob5c | 5 | 3189.2 | 2856.0 | 2662.8 | 2662.8 | 0.3 | - |
| prob5d | 5 | 2086.3 | 1842.0 | 1842.0 | 1842.0 | 0.0 | - |
| prob5e | 5 | 2171.4 | 2171.4 | 2171.4 | 2171.4 | 0.8 | - |
| prob10a | 10 | 6109.6 | 4680.9 | 4680.9 | 4267.0 | 3600 | 8.95% |
| prob10b | 10 | 5576.9 | 4965.5 | 4618.4 | 4487.1 | 3600 | 15.9% |
| prob10c | 10 | 5514.1 | 4109.3 | 3591.9 | 3591.9 | 501.9 | 0.0% |
| prob10d | 10 | 4126.2 | 3662.0 | 3662.0 | 3603.5 | 239.4 | 0.0% |
| prob10e | 10 | 5302.8 | 4964.9 | 4810.3 | 4545.4 | 3600 | 5.0% |

Table 4.2: Performance of algorithms - saving cost

| Problem size | Total cost (saving%) | | | | |
|---|---|---|---|---|---|
| | Solo | Match | | Insertion | |
| 5 | 2509.5 | 2264.5 | (9.8%) | 2225.9 | (11.3%) |
| 10 | 5325.9 | 4476.5 | (15.9%) | 4272.7 | (19.8%) |
| 15 | 7929.6 | 6654.7 | (16.1%) | 6499.7 | (18.0%) |
| 20 | 10561.8 | 8454.6 | (19.9%) | 8201.7 | (22.4%) |
| 25 | 12695.5 | 10430.8 | (17.8%) | 9826.4 | (22.6%) |
| 30 | 16490.1 | 12975.2 | (21.3%) | 12190.0 | (26.1%) |
| 35 | 18367.0 | 14327.1 | (22.0%) | 13576.6 | (26.1%) |

Table 4.3: Performance of algorithms - saving vehicles

| Problem size | Vehicle trips (saving%) | | | | |
|---|---|---|---|---|---|
| | Solo | Match | | Insertion | |
| 5 | 5 | 4.0 | (20.0%) | 3.8 | (24.0%) |
| 10 | 10 | 7.2 | (28.0%) | 5.8 | (42.0%) |
| 15 | 15 | 9.8 | (34.7%) | 8.2 | (45.3%) |
| 20 | 20 | 12.2 | (39.0%) | 9.8 | (51.0%) |
| 25 | 25 | 16.2 | (35.2%) | 11.6 | (53.6%) |
| 30 | 30 | 18.2 | (39.3%) | 13.8 | (54.0%) |
| 35 | 35 | 21.8 | (37.7%) | 15.8 | (54.9%) |

# 5. COOPERATIVE RIDESHARING GAME

## 5.1 Introduction

In Section 4 the ridesharing optimization problem is modeled as a mixed-integer program and the optimal solution is found to minimize the system-wide travel cost. This is beneficial in the society's point of view, assuming each agent accepts the system's assignment. This is, however, a strong assumption considering agents might form their own ridesharing groups if they find doing so is more of their own interest.

Recall that the agents of ridesharing system participate in this system in the hope of saving travel cost. So it is up to the ridesharing service provider to decide how the travel cost would be shared among customers after a ridesharing plan is proposed and accepted by the customers. This is a non-trivial task because if the agents find the cost allocation scheme unfair, they may leave the system and form their own ridesharing group in the long run. This fair cost-allocation situation is critical to the sustainability of a ridesharing system and thus is the motivation of the study in this section.

The ridesharing cost allocation problem is modeled as a cooperative game. Cooperative game theory, due to its close relation to combinatorial optimization, has drawn significant attention of the operations research community. Since its introduction by von Neumannn and Morgenstern (1944), cooperative game theory has developed several solution concepts that aim to resolve the benefits (cost) allocation issues among cooperative players. In this section, we are primarily concerned with a particular cost allocation solution concept - the *nucleolus*. The nucleolus of a cooperative game has several nice properties. Intuitively, it is a solution to the cost allocation problem that *minimizes the maximal dissatisfaction* among the customers.

The concept of nucleolus was first suggested by Schmeidler (1969) and since then was developed by Shapley (1967) and Maschler et al. (1979). Although the nucleolus has several game theoretic virtues, the computation of nucleolus is very difficult. In fact, for a n-player game, as the size of the characteristic function grows exponentially with the number of players, any enumeration algorithm that computes the nucleolus that requires the entire information of the characteristic function takes $O(2^n)$ time, assuming the characteristic function is readily available. Moreover, as will be shown in later section, finding the characteristic function value of ridesharing game involves solving an optimization problem related to TSP, which is NP-hard itself. This means the computation of the nucleolus of RSG can easily become intractable and more efficient algorithm needs to be developed.

In this section, we utilize the nucleolus-finding procedure by successively solving a number of linear programs. This technique was first proposed by Dragan (1981) and Kopelowitz (1967). We combine this technique with a constraint generation framework proposed in Hallefjord et al. (1995), such that the explicit information of the characteristic function of a coalition is only computed when it is necessary. In this way the computational burden is significantly reduced.

Note that the constraint generation approach was first proposed in Gilmore and Gomory (1961) and was successfully applied to solving the cutting stock problem. Utilizing a similar idea, Göthe-Lundgren et al. (1996) studied the basic vehicle routing game (VRG) in which a fleet with homogeneous capacity are available. The authors analyzed the properties of this game and proposed a nucleolus-finding procedure based on coalition generation. Engevall et al. (2004) generalized the model of Göthe-Lundgren et al. (1996) to consider vehicles with heterogeneous capacities and studied a real-world case based on their model.

This section is organized as the following. In Section 5.2 the basics of cooperative

game theory are covered. In Section 5.3, a formulation of the RSP cost allocation problem is developed from a game theory perspective and the properties of the characteristic function are analyzed. Section 5.4 discusses the fairness issues in the RSP game regarding the core and the nucleolus. A coalition generation scheme is then developed to compute the nucleolus. The constraint generation subproblem is explicitly formulated by a mathematical formulation related to the ridesharing optimization problem. In Section 5.5, numerical experiments are conducted and the performance of the proposed nucleolus procedure is evaluated. Finally, conclusions and future research ideas are presented in Section 5.6.

## 5.2 Cooperative Game Theory Background

In this section we discuss specifically cooperative games with transferable utility (TU game), in which the earnings (costs) of a coalition can be expressed by one number. A cooperative game with transferable utility is given by specifying a value for every coalition. The game is defined by a tuple $(N, v)$ where $N = \{1, 2, \ldots, n\}$ is a finite set of players and $v$ is a characteristic function $v : 2^S \to \mathbb{R}$ from the set of coalitions of players in $N$ to a set of payment schemes satisfying $v(\emptyset) = 0$. Here $2^N$ denotes the power set of $N$.

**Definition 5.1** (Superadditivity)**.** *A TU game is superadditive if and only if*

$$v(s \cup t) \geq v(s) + v(t), \quad \forall s, t \subseteq S \text{ satisfying } s \cap t = \emptyset \tag{5.1}$$

**Definition 5.2** (Monotonicity)**.** *Larger coalitions have higher values.*

$$v(s) \leq v(t), \quad \forall s \subseteq t \subseteq S \tag{5.2}$$

**Definition 5.3** (Cohesiveness). *A TU game is cohesive if we have*

$$v(N) \geq \sum_{k=1}^{K} v(S_k) \tag{5.3}$$

*for every partition* $\{S_1, \ldots, S_K\}$ *of* $N$.

Note that cohesiveness ensures that the equilibrium coalition will form.

### *5.2.1 Solution concepts*

In a value game in which players have net profit, a solution concept is a payoff vector $x \in \mathbb{R}^{|S|}$ that dictates the value allocated to each player in $S$, i.e. the earnings each player gets. In a cost game the payoff vector is the cost each player pays.

**Definition 5.4** (Efficiency). *A solution is efficient if the total value is divided exactly by the payoff vector:* $\sum_{i \in N} x_i = v(N)$.

**Definition 5.5** (Individual rationality). *A solution is individually rational if any player gets at least as much as what he would get on his own:* $x_i \geq v(\{i\}), \forall i \in N$

#### *5.2.1.1 Imputation*

**Definition 5.6** (Imputation). *A payoff vector is an imputation if it's individually rational and efficient at the same time.*

#### *5.2.1.2 The core*

**Definition 5.7** (Core). *The core of a TU game is the set of payoff vectors*

$$C(v) = \left\{ x \in \mathbb{R}^N : \sum_{i \in N} x_i = v(N); \sum_{i \in S} x_i \geq v(S), \forall S \subseteq N \right\}. \tag{5.4}$$

In other words, the core of a game is the set of imputations under which no coalition $S$ has a value $x(S)$ that is larger than its members' payoffs' summation.

Thus no coalition has the incentive to leave the grand coalition.

#### *5.2.1.3 The nucleolus*

Let $v : 2^S \to \mathbb{R}$ denote the payoff characteristic function of a cooperative profit game. Then the function gives the amount of collective payoff a set of players could get through forming a coalition. The *excess* of $x$ for a coalition $S \subseteq N$ is defined as $e(x, S) = v(S) - \sum_{i \in S} x_i$. Let $\theta(x) \in \mathbb{R}^{2^N}$ be the excess vector of $x$, with elements $v(S) - \sum_{i \in S} x_i$ arranged in non-increasing order, that is, $\theta_i(x) \geq \theta_j(x), \forall i < j$. Note that a cost allocation vector $x$ is in the core if and only if it is efficient and $\theta_1(x) \leq 0$. Consider the lexicographic ordering of excess vectors: for any two payoff vectors $x, y$, $\theta(x)$ is said to be lexicographically smaller than $\theta(y)$ if $\exists k$ such that $\theta_i(x) = \theta_i(y), \forall i < k$ and $\theta_k(x) < \theta_k(y)$. We denote this ordering by $\theta(x) \prec \theta(y)$.

**Definition 5.8** (Nucleolus)**.** *The nucleolus of a cooperative game is the lexicographically minimal imputation. Denote the nucleolus by $x$ and let $\bar{X}$ be the set of imputations, then we have*

$$\theta(x) \preceq \theta(x'), \quad \forall x' \in \overline{X} \backslash (x). \tag{5.5}$$

The following bankruptcy game gives an example that clarifies the definition of nucleolus.

**Example 5.1** (The Bankruptcy Game)**.** *A company goes bankrupt and owes \$10,000 to creditor A, \$20,000 to creditor B and \$30,000 to creditor C. If only \$36,000 is available for this company to cover these debts, how should this company split its money to pay the creditors?*

If the money is divided proportionally, the company would pay \$6,000 to A, \$12,000 to B, and \$18,000 to C. We denote this allocation by $x = (6, 12, 18)$. We will calculate the nucleolus of this game and compare it with the pro rata allocation.

First, we decide the characteristic function of this game. Apparently, $v(\emptyset) = 0$ and $v(ABC) = 36$ in thousands of dollars. By himself A would not get anything because B and C would split the whole amount; therefore we have $v(A) = 0$. $v(B) = 0$ similarly. For creditor C, even if A and B get the whole amount of their claim, that is $30,000$, C would still get $36,000 - 30,000 = 6,000$. Therefore we have $v(C) = 6$. It is not hard to get $v(AB) = 6$, $v(AC) = 16$, and $v(BC) = 26$.

Table 5.1: Calculation of nucleolus - the bankruptcy game

| S | v(S) | e(x, S) | (6, 12, 18) | (5, 12, 19) | (5, 10.5, 20.5) |
|---|---|---|---|---|---|
| A | 0 | $-x_1$ | -6 | -5 | -5 |
| B | 0 | $-x_2$ | -12 | -12 | -10.5 |
| C | 6 | $6 - x_3$ | -12 | -13 | -14.5 |
| AB | 6 | $6 - x_1 - x_2$ | -12 | -11 | -9.5 |
| AC | 16 | $16 - x_1 - x_3$ | -8 | -8 | -9.5 |
| BC | 26 | $26 - x_2 - x_3$ | -4 | -5 | -5 |

Let $x = (x_1, x_2, x_3)$ be an efficient allocation (i.e., $x_1 + x_2 + x_3 = 36$), we find the excesses in Table 5.1. Note that we can ignore the empty set and the grand coalition since they always have zero excesses. We start from the proportional allocation $(6, 12, 18)$. As is in Table 5.1, the excesses vector is $\theta = (-6, -12, -12, -12, -8, -4)$, with the largest number $-4$ belonging to the coalition BC. The coalition of BC will complain that every other coalition has a better excess value than it does. As a result the next step is to improve BC's excess by increasing $x_2 + x_3$ (i.e., decreasing $x_1$ as $x_1 = 36 - x_2 + x_3$). However, as we make the excess of BC smaller, the excess of A will get larger synchronously. When $x_1 = 5$, these two excesses meet at $-5$. It is not hard to see that no matter how we select the values of $x$, the maximal excess is at least $-5$ because at least one of A or BC would have an excess that is greater

than or equal to $-5$. Therefore, $x_1 = 5$ is in the nucleolus.

Since $x_1$ is fixed, we choose $x_2$ and $x_3$ to make the next largest excess smaller. Choosing the solution $x = (5, 12, 19)$, we find that the largest excess excluding the $-5$'s is $-8$ which belongs to coalition AC. In order to make $e(x, AC)$ smaller, we have to make $x_3$ larger (make $x_2$ smaller). However, when we do so the excesses of B and AB will increase synchronously. Because coalition AB's excess $(-11)$ is closer to $-8$, the nucleolus must be found when $e(x, AB) = e(x, AC)$. Solving the equations,

$$16 - x_1 - x_3 = 6 - x_1 - x_2$$

$$x_2 + x_3 = 31$$

$$x_1 = 5,$$

we find the nucleolus as $(5, 10.5, 20.5)$.

Recall that we defined $\theta(x)$ as the excesses vector which is arranged in non-increasing order. In this example, $\theta(x) = (-4, -6, -8, -12, -12, -12)$ when $x = (6, 12, 18)$. Similarly, let $x' = (5, 12, 19)$ and $x'' = (5, 10.5, 20.5)$, then $\theta(x') = (-5, -5, -8, -11, -12, -13)$ and $\theta(x'') = (-5, -5, -9.5, -9.5, -10.5, -14.5)$. Recall the definition of lexicographical ordering, since $\theta_1(x') = -5 < \theta_1(x) = -4$ we have $\theta(x') \prec \theta(x)$. Because $\theta_3(x'') = -9.5 < \theta_3(x') = -8$ we have $\theta(x'') \prec \theta(x)$. Therefore, $\theta(x'') \prec \theta(x') \prec \theta(x)$, meaning $x''$ is the lexicographically minimal imputation and thus the nucleolus.

## 5.3 RSP From A Game Theory Perspective

Consider a set of ridesharing participants and denote it by $N$. Each participant wants to travel from her origin $s_i$ to her destination $t_i$. Denote the capacity of a vehicle by $Q$. Consider the subsets of participants that do not exceed the vehicle

capacity, i.e. $|S| \leq Q$. For each such participant subset $s$, assume the *feasible* route with minimum cost is known. Here by feasible it means the following conditions are met

1. the route $r$ starts from an agent $d's$ origin and ends at his destination.

2. Let $s_{-d} = s \backslash \{d\}$. For every agent $i \in s_{-d}$, $s_i$ precedes $t_i$ in $r$.

Denote by $c_r$ the cost of such a feasible route and by $R$ the set of feasible routes with minimal cost. Let $a_{ir} = 1$ if participant $i$ (both $s_i$ and $t_i$) is served by route $r$ and 0 otherwise. The RSP can be formulated as

$$\text{(RSP)} \quad z = \quad \min \quad \sum_{r \in R} c_r x_r \tag{5.6}$$

$$\text{s.t.} \quad \sum_{r \in R} a_{ir} x_r = 1, \quad i \in N \tag{5.7}$$

$$x_r \in \{0, 1\}, r \in R \tag{5.8}$$

In the formulation $x_r = 1$ if feasible route $r$ is selected and 0 otherwise. Constraints (5.7) guarantee that each participant is covered by exactly one route. Note that the coefficient $c_r$ in the objective function is obtained by finding the minimal cost route that covers the participants for which $a_{ir} = 1$, that is, by finding the solution to the corresponding TSP with precedence constraints.

It is noted that this formulation is characterized by its large number of columns. Therefore, this formulation is practically solvable by a column generation solution method. Similar approaches were successfully applied to the vehicle routing problems (VRP) (Balinski and Quandt, 1964; Desrochers et al., 1992, see). When we solve the

RSP with a column generation approach, it is of our interest to reduce the number of columns. We show this is possible as follows.

We first introduce the definition of the *profitable* ridesharing route.

**Definition 5.9** (profitability)**.** *Denote by $r(S)$ the corresponding minimum cost feasible route of participant subset $S$. $r(S)$ is* non-profitable *if there exists two non-empty subsets $S_1 \cup S_2 = S, S_1 \cap S_2 = \emptyset$ such that $c_{r(S)} > c_{r(S_1)} + c_{r(S_2)}$. A route is defined profitable otherwise.*

Intuitively, a shared-ride route becomes non-profitable if by ridesharing the participants end up spending more money on the transportation cost.

The following proposition shows that we only need to consider a subset of the columns when solving RSP.

**Proposition 5.1.** *Let $X = \{x_{r_1}, x_{r_2}, \ldots, x_{r_m}\}$ be an optimal solution to RSP, i.e. $x_{r_i} = 1, i = 1, \ldots, m$. Then $r_i, \forall i = 1, \ldots, m$ must be a profitable route.*

*Proof.* Proof by contradiction. Let $X = \{x_{r_1}, x_{r_2}, \ldots, x_{r_m}\}$ be an optimal solution to RSP. Suppose there exists $i^*$ such that $r_{i^*}$ is a non-profitable route. Let $S^*$ be the corresponding participants that are covered by this route. Then by definition there must be two non-empty subsets $S_1^* \cup S_2^* = S^*, S_1^* \cap S_2^* = \emptyset$ such that $c_{r(S^*)} > c_{r(S_1^*)} + c_{r(S_2^*)}$. Since all the customers that are covered by $r_{i^*}$ are also covered by $r(S_1^*)$ and $r(S_2^*)$, we can get a new feasible solution $X'$ to RSP by substituting $x_{r_{i^*}} = 1$ with $x_{r(S_1^*)} = 1$, $x_{r(S_2^*)} = 1$ and $x_{r_{i^*}} = 0$ while keep all the other $x$ variables unchanged. This feasible solution has a strictly less cost than $X$. Contradiction. □

From a game theory perspective, we denote each ridesharing participant, $i \in N$, by a *player* and each subset of participants, $S \subseteq N$, by a *coalition*.

The ridesharing cost allocation problem is the problem of finding a "fair" cost allocation scheme for the ridesharing optimization problem (RSP).

A cooperative ridesharing game is defined by specifying a travel cost for each coalition. The game is defined by a ridesharing group $S$, and a *characteristic function* $c(S) : 2^S \to \mathbb{R}$ from the set of all possible coalitions (sub-ridesharing group) of players in $S$ to a set of payment schemes satisfying $c(\emptyset) = 0$. Here $2^S$ denotes the power set of $S$. In the context of RSP game the characteristic function can be seen as the travel cost occurring if coalition $S \subseteq N$ is formed. Each coalition can be defined by a binary vector $s$ as

$$s_i = \begin{cases} 1, & \text{if customer } i \text{ is a member of the coalition,} \\ 0, & \text{otherwise,} \end{cases}$$

Define $c$ as the objective value of a mathematical program. For all coalitions $S \subseteq N, S \neq \emptyset$, let $c(S)$ be the solution to the following mathematical program

$$c(S) = \quad \min \quad \sum_{r \in R} c_r x_r \tag{5.9}$$

$$\text{s.t.} \quad \sum_{r \in R} a_{ir} x_r = s_i, \quad i \in N \tag{5.10}$$

$$x_r \in \{0, 1\}, \quad r \in R \tag{5.11}$$

Intuitively, $c(S)$ represents the cost of an optimal route that covers the players in $S$, i.e. the players for which $s_i = 1$. It is noted that this program is very similar to VRP. In fact, the columns of this program can be reduced in a similar fashion as VRP. This is stated in the following proposition.

**Proposition 5.2.** *Let $X = \{x_{r_1}, x_{r_2}, \ldots, x_{r_m}\}$ be an optimal solution to $C(S)$, i.e.*

$x_{r_i} = 1, i = 1, \ldots, m$. *Then* $r_i, \forall i = 1, \ldots, m$ *must be a profitable route.*

*Proof.* Similar to the proof of Proposition 5.1. □

When studying a cooperative game, it is of great interest to study the properties of its characteristic function. Assuming that the singleton coalitions have a positive cost, we show the RSP game has the following properties. From here on we denote by $C(\cdot)$ the mathematical program that defines the characteristic function value of $\cdot$, i.e. $c(\cdot)$.

**Proposition 5.3** (Monotonicity)**.** *The characteristic function of the RSP game is* monotone*, that is,* $c(S) \leq c(T), S \subset T \subset N$.

*Proof.* Proof by contradiction. Suppose there exists $S \subset T \subset N$ and $c(S) > c(T)$. Let $X = \{x_r | r \in R\}$ be an optimal solution to $C(T)$. Let $R_T = \{r_i | x_{r_i} = 1\}$. For $r \in R$, we construct a feasible solution to $C(S)$ in the following manner. Let

$$x_r = \begin{cases} 1, & \text{if } \exists i \in S \text{ such that } a_{ir} = 1, \\ 0, & \text{otherwise.} \end{cases}$$

Let $X'$ be the solution constructed in the above way. Denote by $R_S$ the set of selected routes. Intuitively, we keep those routes in $R_T$ that covers at least one player in $S$ and discard those don't.

It is known that $X$ must satisfy

$$\sum_{r \in R} a_{ir} x_r = 1, \quad i \in T$$
$$\sum_{r \in R} a_{ir} x_r = 0, \quad i \in N - T$$

Because $S \subset T$, then $X'$ must satisfy

$$\sum_{r \in R} a_{ir} x_r = 1, \quad i \in S$$
$$\sum_{r \in R} a_{ir} x_r = 0, \quad i \in T - S$$
$$\sum_{r \in R} a_{ir} x_r = 0, \quad i \in N - T$$

This is equivalent to

$$\sum_{r \in R} a_{ir} x_r = s_i, \quad i \in N$$

So $X'$ is a feasible solution to $C(S)$. In addition, since the cost matrix $\{c_{ij}\}$ is positive, the route cost is also positive. Therefore the cost of $X'$ is less than $c(T)$, which is less than $c(S)$. Note that $c(S)$ is the optimal cost, so this is a contradiction. □

**Proposition 5.4** (Subadditivity)**.** *The characteristic cost function of RSP game is* subadditive*, i.e.,* $c(S) + c(T) \geq c(S \cup T), S, T \subset N, S \cap T = \emptyset$.

*Proof.* Let $R_S$, $R_T$ be the optimal solution to $C(S)$, $C(T)$, respectively. Because $S, T \subset N$ and $S \cap T = \emptyset$, $R_{S,T} = R_S \cup R_T$ must cover all the players in $S \cup T$, i.e. $R_{S,T}$ is a feasible solution to $C(S \cup T)$. Since this solution has an objective value equal to $c(S) + c(T)$, we have $c(S) + c(T) \geq c(S \cup T)$. □

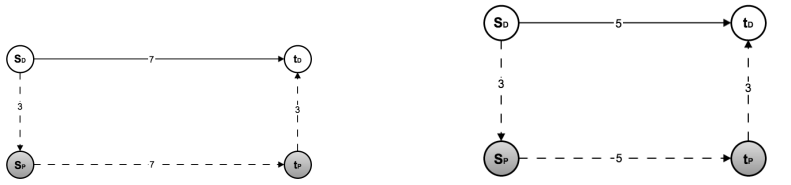
It is noteworthy that subadditivity implies larger coalitions save more. So it is always beneficial to include more people to participant in ridesharing and this is a desirable property of the RSP game.

Denote a coalition $S$ whose cardinality is smaller than the vehicle capacity ($|S| \leq Q$) as a *feasible coalition* and otherwise as an *infeasible coalition*. Denote by $\mathbb{S}$ the

set of feasible coalitions. Then we get

$$c(S) = c_r, \quad \forall r \in R \text{ and } S \in \mathbb{S} \text{ such that } a_{ir} = s_i, i \in N.$$

In addition, denote a coalition such that $\sum_{i \in S} c(i) \geq c(S)$ by a *profitable coalition* and otherwise by a *nonprofitable coalition*.




(a) Profitable Coalition, $c(S) = 13$ (b) Nonprofitable Coalition, $c(S) = 11$

Figure 5.1: Profitable vs. nonprofitable coalition

Figure 5.1 gives an example where forming a coalition will not always produce desirable results: instead of reducing total transportation cost as Figure 5.1(a), Figure 5.1(b) actually increases the total cost, meaning it doesn't make much sense to form such a coalition. In this case the players are better off on their own. Note that the profitability of forming a coalition in a large extent depends on the relative geo-locations of the players.

## 5.4 Fairness and Stability in RSP Game

### *5.4.1 The core and the nucleolus*

#### *5.4.1.1 The core*

Let $y_i$ be the cost allocated to agent $i$, $i \in N$. The *core* of the RSP game is the set of the cost allocation plans $y$, such that

$$\sum_{i \in N} y_i = c(N), \tag{5.12}$$

$$\sum_{i \in S} y_i \leq c(S), \forall S \subset N. \tag{5.13}$$

The above inequalities can be interpreted as no single player or coalition should make a payment that is greater than their cost on their own. A cost allocation scheme that is in the core is a good allocation as no coalition has the incentive to leave the grand coalition. An inequality in (5.13) is called a *core defining inequality* (CDI).

It is observed that the number of CDIs is in the scale of $O(2^N)$. As will be shown in later sections, in order to find the core and the nucleolus efficiently, it is important and of our great interest to reduce the number of CDIs. This is possible through the following propositions.

**Proposition 5.5.** *Any CDI with a nonprofitable coalition $S, S \neq N$, is not needed in* (5.13)*.*

*Proof.* Consider any nonprofitable coalition $\hat{S}$, $\hat{S} \neq N$. Denote by $\{1, 2, \ldots, m\}$ the players in $\hat{S}$. By definition of nonprofitable coalition we have $\sum_{i \in \hat{S}} c(i) < C(\hat{S})$.

Note that all individual players are also singleton coalitions. It follows that

$$y_i \leq c(i), \forall i \in \hat{S} \quad \Longrightarrow \quad \sum_{i \in \hat{S}} y_i \leq \sum_{i \in \hat{S}} c(i) < c(\hat{S}).$$

□

**Proposition 5.6.** *(Göthe-Lundgren et al., 1996) For a RSP game with non-empty core, any CDI with an infeasible coalition $S, S \neq N$, is not needed in* (5.13).

*Proof.* Let $\hat{S}$, $\hat{S} \neq N$ be an infeasible coalition. Denote by $\{r_1, \ldots, r_m\}$ the corresponding optimal routes and $\{s_1, \ldots, s_m\}$ the disjoint feasible coalitions corresponding to the optimal routes. Since we have $\sum_{j=1}^{m} \sum_{i \in S_j} y_i = \sum_{i \in \hat{S}} y_i$ and $\sum_{j=1}^{m} c(S_j) = c(\hat{S})$, then we have the following

$$\sum_{i \in S_j} y_i \leq c(S_j), \forall j = 1, \ldots, m \quad \Longrightarrow \quad \sum_{i \in \hat{S}} y_i \leq c(\hat{S}).$$

□

From Proposition 5.1 and Proposition 5.6 we have

$$C = \{y | \sum_{i \in S} y_i \leq c(S), S \in \mathbb{S}; \quad \sum_{i \in N} y_i = c(N)\}.$$

Thus, when the core of the RSP game is non-empty, the only characteristic function values of our interests are those corresponding to profitable and feasible coalitions. This, as will be stated in later sections, reduces the size of the coalition-generating subproblem dramatically. Note that the calculation of $c(S)$ for a coalition $S$ is equivalent to solving the corresponding TSP with pick-up and drop-off constraints for the customers for which $s_i = 1$.

#### *5.4.1.2 The nucleolus*

Note that the previous definitions in Section 5.2 are based on value game. In a cooperative ridesharing game, in which players share travel cost, allocations are the payments each player need to pay. That is to say, cooperative ridesharing game (RSP game) is a cost game.

Let $c : 2^S \to \mathbb{R}$ denote the cost characteristic function of a cooperative ridesharing game. Then the function gives the amount of collective cost a group of players need to pay through forming a coalition. Before we define the nucleolus of the RSP game, first recall the definition of *excess*. In an RSP game, the excess of $y$ for a coalition $S \subseteq N$ is defined as $e(y, S) = c(S) - \sum_{i \in S} y_i$ and measures the amount of cost-saving of coalition $S$ in the allocation $y$, compared to $c(S)$. Note that when $e(y, S)$ is negative, it means the sum of the cost of $S$ in the allocation $y$ must exceed $c(S)$. Thus $e(y, S)$ measures the dissatisfaction of $S$ in the allocation $y$. Recall that the core is defined as the set of imputations such that $c(S) \geq \sum_{i \in S} y_i$ for all coalitions $S$, then we have that an imputation $y$ is in the core if and only if all its excesses are positive or zero. Denote by $\theta(y) \in \mathbb{R}^{2^N}$ the excess vector of $y$ whose elements $c(S) - \sum_{i \in S} y_i$ are arranged in non-decreasing order, that is, $\theta_i(y) \leq \theta_j(y), \forall i < j$. Then a cost allocation vector $y$ is in the core if and only if it is efficient and $\theta_1(y) \geq 0$. Consider the lexicographic ordering of excess vectors: for two payment vectors $x, y$, we say $\theta(x)$ is lexicographically greater than $\theta(y)$ if $\exists k$ such that $\theta_i(x) = \theta_i(y), \forall i < k$ and $\theta_k(x) > \theta_k(y)$. Denote this ordering by $\theta(x) \succ \theta(y)$.

**Definition 5.10** (Nucleolus)**.** *The nucleolus of a RSP game is the lexicographically maximal imputation. Denote the nucleolus by $y$ and let $\bar{Y}$ be the set of imputations, then we have*

$$\theta(y) \succeq \theta(y'), \quad \forall y' \in \overline{Y} \backslash y.$$

Intuitively, the nucleolus minimizes the maximal dissatisfaction of all the coalitions in the ridesharing system. As a result, on the condition that the core is nonempty, the nucleolus is the *center* of the core. It is of our interest to investigate the non-emptiness of the core of RSP game because the definitions of nucleolus and the core are related. In fact, the following example shows that the core of RSP game may be empty.

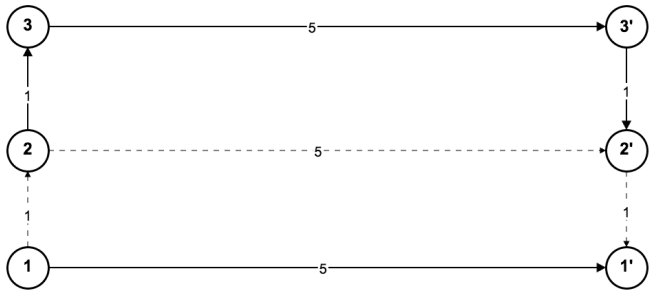



Figure 5.2: A three player example

The transportation costs of three players $1, 2, 3$ are given in Figure 5.2. Assuming that each player's vehicle has a capacity of one extra passenger seats, the characteristic function of this 3-player game is then defined by $c(\{1\}) = c(\{2\}) = c(\{5\}) = 5$, $c\{1, 2\} = c(\{2, 3\}) = 7$ (e.g., $1 - 2 - 2' - 1'$), $c(\{1, 3\}) = 9$ (e.g., $1 - 3 - 3' - 1'$) and $c(\{1, 2, 3\}) = 12$. The optimal route configuration is, for example, $2 - 3 - 3' - 2'$ and $1 - 1'$. We show the calculation of nucleolus of this example in Table 5.2.

As an initial guess, we try $(4, 4, 4)$. In Table 5.2, we find that the minimum

Table 5.2: Calculation of nucleolus - empty core

| S | c(S) | e(y, S) | (4, 4, 4) | $(4\frac{2}{3}, 2\frac{2}{3}, 4\frac{2}{3})$ |
|---|---|---|---|---|
| 1 | 5 | $5 - y_1$ | 1 | $\frac{1}{3}$ |
| 2 | 5 | $5 - y_2$ | 1 | $2\frac{1}{3}$ |
| 3 | 5 | $5 - y_3$ | 1 | $\frac{1}{3}$ |
| 1 and 2 | 7 | $7 - y_1 - y_2 = y_3 - 5$ | -1 | $-\frac{1}{3}$ |
| 2 and 3 | 7 | $7 - y_2 - y_3 = y_1 - 5$ | -1 | $-\frac{1}{3}$ |
| 1 and 3 | 9 | $9 - y_1 - y_3 = y_2 - 3$ | 1 | $-\frac{1}{3}$ |

excess happens at coalition $(1, 2)$ and $(2, 3)$. These are the coalitions with maximum dissatisfaction. To improve this, we must increase both $y_1$ and $y_3$. This involves decreasing $y_2$, and will decrease the excess for $(1, 3)$ at the same rate. Also note that player 1 and player 3 have symmetrical roles in this game, thus we can conclude that the best scenario occurs when the excesses for $(1, 2)$, $(2, 3)$ and $(1, 3)$ are all equal. Solving the equations,

$$y_3 - 5 = y_1 - 5 = y_2 - 3$$

$$y_1 + y_2 + y_3 = 12,$$

we find the nucleolus of this game is $y = (4\frac{2}{3}, 2\frac{2}{3}, 4\frac{2}{3})$ and $C = \emptyset$ (note that $c(1, 2) < y_1 + y_2$).

Interestingly, if we increase the capacity of the vehicles to two extra seat, then we obtain a game with a nonempty core. In this case, the characteristic function is defined in the following fashion. For the singleton coalitions and the 2-coalitions, characteristic function values remain the same. However, for the grand coalition $c(\{1, 2, 3\}) = 9$. The optimal route configuration is, for example, $1-2-3-3'-2'-1'$. The calculation of nucleolus of this example is shown in Table 5.3.

Initially, we try $(3, 3, 3)$. In Table 5.3, we find that the minimum excess happens

Table 5.3: Calculation of nucleolus - nonempty core

| S | c(S) | e(y, S) | (3, 3, 3) | $(\frac{11}{3}, \frac{5}{3}, \frac{11}{3})$ |
|---|---|---|---|---|
| 1 | 5 | $5 - y_1$ | 2 | 4/3 |
| 2 | 5 | $5 - y_2$ | 2 | 10/3 |
| 3 | 5 | $5 - y_3$ | 2 | 4/3 |
| 1 and 2 | 7 | $7 - y_1 - y_2 = y_3 - 2$ | 1 | 5/3 |
| 2 and 3 | 7 | $7 - y_2 - y_3 = y_1 - 2$ | 1 | 5/3 |
| 1 and 3 | 9 | $9 - y_1 - y_3 = y_2$ | 3 | 5/3 |

at coalition $(1, 2)$ and $(2, 3)$. These are the coalitions with maximum dissatisfaction. To improve this, we must increase both $y_1$ and $y_3$. This involves decreasing $y_2$, and will decrease the excess for $(1, 3)$ at the same rate. Also note that player 1 and player 3 have symmetrical roles in this game, therefore we can conclude that the best scenario we can achieve happens when the excesses for $(1, 2)$, $(2, 3)$ and $(1, 3)$ are all equal. Solving the equations,

$$y_3 - 2 = y_1 - 2 = y_2$$

$$y_1 + y_2 + y_3 = 9,$$

we find the nucleolus of this game is $y = (11/3, 5/3, 11/3)$. Here, $C \neq \emptyset$ and the nucleolus of this game is the center of the core.

These observations can be generalized below.

**Proposition 5.7.** *(Göthe-Lundgren et al., 1996) Let $R$ be an optimal route configuration of $N$, i.e. $R$ consists of routes serving the players of feasible disjoint coalitions $S_1, S_2, \ldots, S_m$. Then we have*

$$\sum_{j \in S_r} y_j = c(S_r), \quad \forall y \in C \text{ and } 1 \leq r \leq m.$$

*Proof.* Note that $c(N) = \sum_{r=1}^{m} c(S_r)$ and $\sum_{r=1}^{m} \sum_{j \in S_r} y_j = \sum_{j \in N} y_j, \forall y \in R^n$. In addition any $y \in C$ satisfies $\sum_{j \in N} y_j = c(N)$ and $\sum_{j \in S_r} y_j, \forall 1 \leq r \leq m$. Therefore, $c(N) = \sum_{j \in N} y_j = \sum_{r=1}^{m} \sum_{j \in S_r} y_j \leq \sum_{r=1}^{m} c(S_r) = c(N)$. It implies that all inequalities must be equalities for this equation to hold, meaning $\sum_{j \in S_r} y_j = c(S_r), \quad \forall y \in C$ and $1 \leq r \leq m$. □

Therefore, if the core of RSP game is nonempty, then the cost of an optimal route should be split only among the players served by that route.

### *5.4.2 An algorithm to find the nucleolus*

#### *5.4.2.1 The master problem*

Since the nucleolus is the cost allocation which minimizes the maximal dissatisfaction, it can be represented by the solution to

$$\max_{y \in Y} \min_{\forall S \subset N} (c(S) - \sum_{i \in S} y_i),$$

which can be transformed to a linear program

$$(P^1) \quad \max \quad w \tag{5.14}$$

$$\text{s.t.} \quad w \leq c(S) - \sum_{i \in S} y_i, \quad S \in \mathbb{S} \tag{5.15}$$

$$\sum_{i \in N} y_i = c(N). \tag{5.16}$$

Notice that the LP program has $O(|\mathbb{S}|)$ constraints, and computing $c(S), S \in \mathbb{S}$ involves solving the corresponding TSPPD. So the LP program can easily become intractable. We therefore approach this problem with a constraint generation procedure. Hallefjord et al. (1995) has suggested such an approach for linear programming games. Göthe-Lundgren et al. (1996) has used a similar approach to solve the vehicle

routing problem (VRP) game.

Since before searching for the nucleolus, we should already know the solution to the corresponding RSP, thus the optimal route configuration, we can start $(P_1)$ with the coalitions corresponding to the optimal routes. Besides, the singleton coalitions' cost values are readily available. Denote by $\Omega \in \mathbb{S}$ the available coalitions, then $(P_1)$ can be replaced by the following relaxed problem

$$(P_M^1) \quad \max \quad w \tag{5.17}$$

$$\text{s.t.} \quad w \leq c(S) - \sum_{i \in S} y_i, \quad S \in \Omega \tag{5.18}$$

$$\sum_{i \in N} y_i = c(N). \tag{5.19}$$

If the solution to $(P_M^1)$ is unique, let it be $(y^*, w^*)$, i.e. $\theta_1(y^*) > \theta_1(y'), \forall y' \in Y \backslash \{y^*\}$, then $y^*$ is the nucleolus of the game. If the solution to $(P_M^1)$ is not unique, we continue to find the greatest $\theta_2(y)$ among the $y \in Y$ with $\theta_1(y) = w^*$. We continue this process until the solution to the linear program is unique. At stage $t$ the master LP problem to be solved is

$$(P_M^t) \quad \max \quad w_t \tag{5.20}$$

$$\text{s.t.} \quad w_t \leq c(S) - \sum_{i \in S} y_i, \quad S \in \mathbb{S} \backslash \bigcup_{\tau=1}^{t-1} \Gamma_\tau, \tag{5.21}$$

$$w_\tau = c(S) - \sum_{i \in S} y_i, \quad S \in \Gamma_\tau, \tau = 1, \ldots, t-1, \tag{5.22}$$

$$\sum_{i \in N} y_i = c(N). \tag{5.23}$$

The solution to the last program in this series is the nucleolus of this game. Let $\Pi_{t,S}$ be the dual variable corresponding to constraint $w \leq c(S) - \sum_{i \in S} y_i$. Let $\Gamma_t$

denote the set of coalitions whose corresponding constraints are *binding*, that is, $\Gamma_t = \{S \in \mathbb{S} \bigcup_{\tau=1}^{t-1} \Gamma_\tau | \Pi^*_{t,S} > 0\}$.

The essential idea of constraint generation approach is trying to find the nucleolus with explicit information of only a small portion of the entire coalition set. This goal is realized by finding the most violated constraint that is not yet included in $\Omega$ via a subproblem after the master problem is solved at each stage. Denote the optimal solution to $(P^1_M)$ by $y^* = (y^*_1, \ldots, y^*_n)$. The constraint that is violated the most, aka the most unhappy coalition given the cost allocation scheme $y^*$, is obtained through solving the following subproblem

$$(P_S) \min_{S \in \mathbb{S} \setminus \Omega} c(S) - \sum_{i \in S} y^*_i - w^*$$

This nucleolus-finding procedure for a ridesharing game is developed based on the theories and techniques proposed in Dragan (1981) and Kopelowitz (1967) and a general constraint generation framework proposed in Hallefjord et al. (1995). The pseudocode of this procedure is given in Algorithm 5.1. First, at stage $t$ the master LP problem $P^t_M$ is solved and both the primal and dual solutions are returned. Second, a subproblem $P_S$ is solved and the least satisfied constraint ($s^*$) that is not yet included is identified. If $c^* \leq 0$, then we include $s^*$ in $\Omega_{INEQ}$ and resolve $P^t_M$ with newly included constraint $w_t \leq c(s^*) - \sum_{i \in s^*} y^*_i$. This stage iterates between the master problem and the subproblem until no coalition violates the rationality constraints of the master problem (i.e. $c^* \geq 0$). When this is achieved, we identify the active and binding constraints, reformulate the master problem (modify $\Omega_{INEQ}$ and $\Omega_{EQ}$) and proceed to the next stage ($t = t + 1$). This process continues until the solution $y^*$ to the master problem is unique. And this last solution is the nucleolus of the RSP game. Note that in the procedure SP.addCut($s^*$), a cut of the type

**Algorithm 5.1:** Procedure of finding the nucleolus of a cooperative ridesharing game

**input** : Geolocations of customers
**output**: Nucleolus of the ridesharing game

t := 1 ;
$\Omega_{INEQ}$ ;
STOP := false ;
**while** *!STOP* **do**
  Solve a master problem $P_M^t$ ;
  Solve a subproblem $P_S$ ;
  $c^* = \min_S c(S) - \sum_{i \in S} y_i^* - w^*$ ;
  $s^* = \arg\min c(S) - \sum_{i \in S} y_i^* - w^*$ ;
  **if** $c^* \leq 0$ **then**
    SP.addCut($s^*$) ;
    $\Omega_{INEQ} := \Omega_{INEQ} \cup \{s^*\}$ ;
  **end if**
  **else**
    STOP := true ;
    **for** *every active and binding constraint s* **do**
      STOP := false ;
      $\Omega_{INEQ} := \Omega_{INEQ} \backslash \{s\}$ ;
      $\Omega_{EQ} := \Omega_{EQ} \cup \{s\}$ ;
    **end for**
    $t := t + 1$ ;
  **end if**
**end while**

of inequality (5.46) is added to the subproblem to prevent the duplication of row associated with coalition $s^*$.

### *5.4.3 Coalition generation subproblem – general*

Recall in the nucleolus-finding scheme described in Algorithm 5.1, it involves finding the most violated constraint in the subproblem. This is equivalent to finding the "least satisfied" subset of customers with a given allocation proposal. A general

formulation of the subproblem is thus

$$(P_S^0) \min_{S \in \mathbb{S} \setminus \Omega} c(S) - \sum_{i \in S} y_i^* - w^* \tag{5.24}$$

$$\sum_{\{i|s_i^j=0\}} s_i + \sum_{\{i|s_i^j=1\}} (1 - s_i) \geq 1, \quad j|S_j \in \Omega \tag{5.25}$$

$$s_i \in \{0, 1\}, i \in N \tag{5.26}$$

Constraints (5.25) are preventing the re-generation of constraints.

Note that calculating $c(S)$ is equivalent to solving the RSP model for customers $i \in S$, i.e. those $s_i = 1$. This implies that we can formulate the subproblem $P_S^0$ explicitly. Denote by $G = (V, E)$ the graph of the RSP game with vertex set $V = V_O \cup V_D \cup \{0\}$ and edge set $E = \{(i, j)|i, j \in V, i \neq j\}$. Here vertex 0 is the "dummy" depot such that any edge incident with it has a cost of 0. $V_O(V_D)$ is the origin (destination) vertex set of players in $N$. Each player is associated with a profit (prize) equal to $y_i^*$. The subproblem of the constraint generation procedure is to find a subset of customers in $N$ which maximizes the total prize minus the total cost, while conforming to certain constraints.

$$(P_S^1) \quad \pi = \quad \max \quad \sum_{k\in N} y_k^* s_k - \sum_{i\in V}\sum_{j\in V} c_{ij}\lambda_{ij} + w^* \tag{5.27}$$

$$\sum_{\{i|s_i^j=0\}} s_i + \sum_{\{i|s_i^j=1\}} (1-s_i) \geq 1, \quad j|S_j \in \Omega \tag{5.28}$$

$$\lambda_{0i} - \lambda_{(i+n)0} = 0, \quad i = 1, 2, \ldots, n \tag{5.29}$$

$$\sum_{i=0}^{2n} \lambda_{ik} = s_k, \quad k \in N \tag{5.30}$$

$$\sum_{i=0}^{2n} \lambda_{ki} = s_k, \quad k \in N \tag{5.31}$$

$$\sum_{i=0}^{2n} \lambda_{i,k+n} = s_k, \quad k \in N \tag{5.32}$$

$$\sum_{i=0}^{2n} \lambda_{k+n,i} = s_k, \quad k \in N \tag{5.33}$$

$$u_i - u_j + p\lambda_{ij} \leq p-1, \quad 1 \leq i \neq j \leq 2n \tag{5.34}$$

$$u_i < u_{i+n}, \quad i = 1, 2, \ldots, n \tag{5.35}$$

$$\sum_k y_{ik} = 1, \quad i = 1, \ldots, 2n; \quad k = 1, \ldots, n \tag{5.36}$$

$$y_{ik} = y_{(i+n)k}, \quad i = 1, \ldots, n; \quad k = 1, \ldots, n \tag{5.37}$$

$$\lambda_{0i} = y_{ii}, \quad i = 1, \ldots, n \tag{5.38}$$

$$\lambda_{ij} \leq M_1(1 - z_{ij}), \quad 1 \leq i \neq j \leq 2n \tag{5.39}$$

$$-M_2 z_{ij} \leq y_{ik} - y_{jk} \leq M_2 z_{ij}, \quad 1 \leq i \neq j \leq 2n; \quad k = 1, \ldots, n \tag{5.40}$$

$$x_{ij} \in \{0, 1\}, \quad 0 \leq i \neq j \leq 2n \tag{5.41}$$

$$y_{ik} \in \{0, 1\}, \quad i = 1, \ldots, 2n; \quad k = 1, \ldots, n \tag{5.42}$$

$$z_{ij} \in \{0, 1\}, \quad 1 \leq i \neq j \leq 2n \tag{5.43}$$

Easily to see, this program ($P_S^1$) is closely related to the RSP model in Section 4. This problem can be termed as *prize-collecting RSP* (see (Balas, 1989) for an analogy of TSP and prize-collecting TSP). An explanation of the constraints in the model can be found in Section 4.

### *5.4.4 Coalition generation subproblem – non-empty core*

Recall that in Algorithm 5.1 searching the most violated constraint in each iteration is a non-trivial task. Also recall that when the RSP game has a non-empty core, the only coalitions that are non-redundant are the feasible coalitions. Notice that $c(S), S \in \mathbb{S}$ is the minimum cost of a feasible route that covers the origin and destination of all the players in $S$, that is, those $s_i = 1, i \in N$. This inspires us to formulate the subproblem explicitly. Let $G = (V, E)$ be the graph with vertex set $V = V_O \cup V_D \cup \{0\}$ and edge set $E = \{(i, j)|i, j \in V, i \neq j\}$. Here vertex 0 is the "dummy" depot such that any edge incident with it has a cost of 0. $V_O(V_D)$ is the origin (destination) vertex set of players in $N$. Each player is associated with a profit (prize) equal to $y_i^*$. The constraint generation subproblem finds a feasible route in $G$ which maximizes the total prize minus cost, while conforming to the following constraints

1. Exactly one player is assigned as the driver in the route
2. The route length does not exceed two times the seat capacity of a passenger car $Q$
3. The route starts from the driver's origin and ends at his/her destination
4. The pick-up drop-off precedence constraints are respected

Denote the edge selection variable in the graph by $\lambda$. This problem is represented by

$$(P_S^2) \quad \pi = \quad \max \quad \sum_{k\in N} y_k^* s_k - \sum_{i\in V}\sum_{j\in V} c_{ij}\lambda_{ij} + w^* \tag{5.44}$$

$$\text{s.t.} \quad \sum_{k\in N} s_k \leq Q \tag{5.45}$$

$$\sum_{\{i|s_i^j=0\}} s_i + \sum_{\{i|s_i^j=1\}} (1-s_i) \geq 1, \quad j|S_j \in \Omega \tag{5.46}$$

$$\lambda_{0i} - \lambda_{(i+n)0} = 0, \quad i = 1,2,\ldots,n \tag{5.47}$$

$$\sum_{k\in N} \lambda_{0k} = 1 \tag{5.48}$$

$$\sum_{k\in N} \lambda_{k+n,0} = 1 \tag{5.49}$$

$$\sum_{i=0}^{2n} \lambda_{ik} = s_k, \quad k \in N \tag{5.50}$$

$$\sum_{i=0}^{2n} \lambda_{ki} = s_k, \quad k \in N \tag{5.51}$$

$$\sum_{i=0}^{2n} \lambda_{i,k+n} = s_k, \quad k \in N \tag{5.52}$$

$$\sum_{i=0}^{2n} \lambda_{k+n,i} = s_k, \quad k \in N \tag{5.53}$$

$$u_i - u_j + p\lambda_{ij} \leq p-1, \quad 1 \leq i \neq j \leq 2n \tag{5.54}$$

$$u_i < u_{i+n}, \quad i = 1,2,\ldots,n \tag{5.55}$$

$$\lambda_{ij} \in \{0,1\}, \quad i,j \in V, i \neq j \tag{5.56}$$

$$s_k \in \{0,1\}, \quad k \in N \tag{5.57}$$

Constraints (5.46) put the restriction that a constraint that is generated before is not generated again. Constraints (5.45) are the capacity constraints and (5.47)

forces a tour to start at a driver's origin and end at a driver's destination. Constraints (5.48) and (5.49) stipulate that exactly one player is assigned as the driver. Constraints (5.50), (5.51), (5.52) and (5.53) are the flow balancing constraints for each vertex. Constraints (5.54) are the subtour elimination constraints and (5.55) are the precedence constraints.

It is not hard to notice that this program is closely related to the RSP model developed in Section 4 and the explicit formulation $P_S^1$ in the previous subsection. Note that although related, $P_S^2$ is much easier to solve than $P_S^1$.

## 5.5 Experiments

We have implemented the nucleolus algorithm (with both $P_S^1$ and $P_S^2$ as subproblem) in Java with CPLEX 12.6 and the Concert library. In this section, we first show the results of nucleolus algorithm with $P_S^2$ as subproblem. Because $P_S^2$ is much easier to solve than $P_S^1$, and the coalitions needed are much fewer in $P_S^2$ than in $P_S^1$, nucleolus algorithm with $P_S^2$ can not only calculate the nucleolus when the RSP has a non-empty core, but can also be used to find an approximate nucleolus when the corresponding RSP has an empty core. Next, we show a comparison between the nucleolus and the approximate nucleolus, obtained by using the nucleolus algorithm with $P_S^1$ and $P_S^2$ as the subproblem, respectively.

### *5.5.1 Approximate nucleolus*

We report results for two instances of the 10-player problem, which is the largest problem we have solved. As will be shown later, the computational bottleneck is not at the nucleolus algorithm but at solving the corresponding ridesharing optimization problem (RSP). The data set used here is the same as the data set in the experiments of Section 4. Table 5.4 shows the geographical locations of the players. After solving the RSP MIP model, the optimal ridesharing plan is $\{1\}$, $\{3,5\}$, $\{4,6,7\}$, $\{8\}$, $\{2,9\}$,

$\{10\}$. These, along with other singleton coalitions are used to generate the initial constraints of the master LP problem.

At stage 1 three constraints are generated by solving the subproblem. They correspond to the coalitions of $\{3,4,6\}$, $\{4,6\}$, and $\{2,3,4,5,9\}$. Active constraints corresponding to coalitions $\{8\}$, $\{10\}$ and $\{1\}$ are then identified at the end of stage 1. At stage 2, we notice that the solution to the master problem $(P_M^2)$ is unique, thus we find the approximate nucleolus. The approximate nucleolus of this game is listed in the last column of Table 5.4.

In total, $10+3+3=16$ out of $2^{10}-2=1022$ constraints are needed to compute the approximate nucleolus, which is only a very small fraction (1.6%).

Table 5.4: Data and approximate nucleolus of prob10c

| Customer Number | Pickup Coordinates | Drop-off Coordinates | Nucleolus Cost |
|---|---|---|---|
| 1 | (387, 137) | (918, 786) | 346.2 |
| 2 | (595, 4) | (852, 236) | 267.1 |
| 3 | (514, 483) | (9, 481) | 627.5 |
| 4 | (342, 655) | (609, 55) | 729.7 |
| 5 | (715, 887) | (372, 215) | 434.3 |
| 6 | (111, 687) | (777, 91) | 250.4 |
| 7 | (692, 933) | (203, 173) | 97.5 |
| 8 | (791, 847) | (488, 312) | 226.1 |
| 9 | (702, 762) | (928, 755) | 520.8 |
| 10 | (543, 443) | (90, 700) | 92.4 |

In our second experiment of prob10d (see Table 5.5), the optimal ridesharing configuration is $\{1\}$, $\{2,3,4,6\}$, $\{7\}$, $\{5,8\}$, $\{9\}$ and $\{10\}$. At stage 1, four constraints are generated after via solving the subproblem. They correspond to the coalitions of $\{2,3,4,9\}$, $\{3,6\}$, $\{2,9\}$ and $\{2,3,4,5,6\}$. At stage 2, the master LP problem is

found to have a unique solution. This solution is thus the approximate nucleolus of this game. The approximate nucleolus is listed in the last column of Table 5.5. In total, $(10+2+4)/1022 \approx 1.6\%$ constraints were needed to compute the approximate nucleolus.

Table 5.5: Data and approximate nucleolus of prob10d

| Customer Number | Pickup Coordinates | Drop-off Coordinates | Nucleolus Cost |
|---|---|---|---|
| 1 | (60, 742) | (34, 697) | 52.0 |
| 2 | (730, 471) | (390, 845) | 444.1 |
| 3 | (964, 151) | (39, 78) | 808.7 |
| 4 | (336, 763) | (11, 332) | 481.2 |
| 5 | (330, 593) | (570, 862) | 326.9 |
| 6 | (496, 333) | (88, 346) | 374.1 |
| 7 | (168, 403) | (432, 341) | 271.2 |
| 8 | (343, 502) | (525, 846) | 173.3 |
| 9 | (600, 534) | (585, 615) | 83.0 |
| 10 | (18, 952) | (494, 605) | 589.0 |

It is noteworthy to point out that the computational time for both instances is very small (less than 10s), indicating the bottleneck is the optimization solution method (recall this takes more than 100 seconds).

### *5.5.2 Nucleolus vs. approximate nucleolus*

In this subsection we conduct two experiments to compare the actual nucleolus and the approximate nucleolus. Finding the actual nucleolus by using Algorithm 5.1 with $P_S^1$ as subproblem is a very time-consuming process. In our first example, problem8a, it takes 5 hours to find the actual nucleolus. In our second example, problem8b, the time it takes to find the actual nucleolus goes up to 20 hours.

The actual nucleolus cost and approximate nucleolus cost are summarized in

Table 5.6 and Table 5.7. As we can see, the solutions obtained by the approximate nucleolus algorithm are a close approximation for the actual nucleolus in both cases.

It is of our interest to see the computational performance of Algorithm 5.1 using $P_S^1$ and $P_S^2$. In problem8a, a total of 193 constraints are generated by $P_S^1$, comparing to a total of 9 constraints generated by $P_S^2$. In problem10d, a total of 184 coalitions are generated by $P_S^1$, comparing to a total of only 9 coalitions generated by $P_S^2$. Note that the total number of coalitions (not including the empty set and the universal set) is $2^8 - 2 = 254$ for problem8a and problem8b. Therefore, Algorithm 5.1 using $P_S^1$ generated $193/254 = 76\%$ and $184/254 = 72\%$ of the total constraints to find the nucleolus in problem8a and problem8b, respectively. So Algorithm 5.1 using $P_S^1$ is more like an enumeration procedure. Thus $P_S^2$ is much more computationally efficient than $P_S^1$. Since the number of constraints generated by $P_S^1$ are significantly higher than that of $P_S^2$. This along with the fact that $P_S^1$ is much harder to solve than $P_S^2$ explains the significant time difference between Algorithm 5.1 using $P_S^1$ and $P_S^2$.

In Figure 5.3 and 5.4 we measure the Euclidean distance between the incumbent nucleolus and the actual nucleolus ($\sqrt{\sum_{i \in N}(y_i^* - y_i)^2}$) as the algorithm iterates. These two figures show the solution path of the algorithm in both cases. As can be seen, in both cases, the algorithm found the nucleolus before it stopped. This happened before the 20th iteration in problem8a, and before 75 constraints were generated in problem8b. It means the majority of the running time of this algorithm is consumed after the actual nucleolus is found.

## 5.6 Conclusions

In this section, we studied an important problem faced by ridesharing service provider: how to allocate cost among ridesharing participants to ensure sustainabil-

Table 5.6: Nucleolus vs. approximate nucleolus – problem8a

| Customer Number | Nucleolus Cost | Approximate Nucleolus Cost |
|---|---|---|
| 1 | 442.1 | 449.9 |
| 2 | 505.0 | 512.8 |
| 3 | 639.2 | 636.6 |
| 4 | 409.3 | 406.7 |
| 5 | 527.6 | 525.0 |
| 6 | 465.8 | 463.2 |
| 7 | 228.4 | 225.8 |
| 8 | 519.4 | 516.8 |

Table 5.7: Nucleolus vs. approximate nucleolus – problem8b

| Customer Number | Nucleolus Cost | Approximate Nucleolus Cost |
|---|---|---|
| 1 | 366.5 | 366.5 |
| 2 | 507.1 | 457.7 |
| 3 | 594.1 | 617.1 |
| 4 | 387.6 | 456.7 |
| 5 | 1012.5 | 1035.6 |
| 6 | 258.1 | 235.1 |
| 7 | 545.3 | 571.8 |
| 8 | 235.9 | 166.8 |

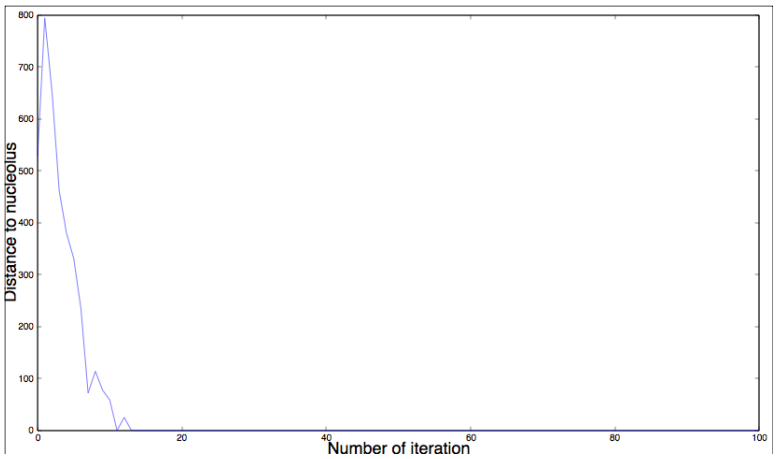



Figure 5.3: Solution path – problem8a

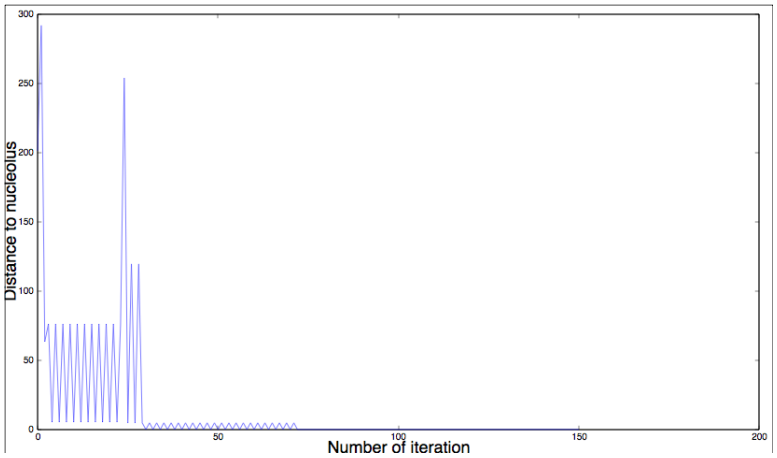



Figure 5.4: Solution path – problem8b

ity and fairness. This fair cost allocation problem was modeled as a cooperative game. A special property of the cooperative ridesharing game is that, its characteristic function values are calculated by solving an optimization problem. To better understand this game, we further studied the characteristic function and proved it to be monotone, subadditive, but non-convex. We then proposed an iterative constraint-generation algorithm (Algorithm 5.1) for calculating the nucleolus of the

RSP game in two situations – the game has an empty core and the game has a non-empty core. In both cases the algorithm utilizes an explicitly formulated MIP as the subproblem to generate constraints. When the game has an empty core, the algorithm uses $P_S^1$ as the subproblem and becomes an enumeration procedure to find the nucleolus of the game. When the game has a non-empty core, this algorithm uses $P_S^2$ as the subproblem which utilizes the special properties of the RSP game such that the characteristic function values are computed only when they are needed. Therefore the number of subproblems (an NP-hard optimization problem) that need to be solved is significantly reduced. Experiments showed that by adopting this algorithm with $P_S^2$ only a small fraction (1.6%) of the coalition constraints were needed to find the nucleolus. It is also found in the experiments that when the emptiness of the RSP game is unclear, the algorithm with $P_S^2$ can be used to find an approximate nucleolus that is close to the actual nucleolus. This indicates that our proposed algorithm is promising in finding nucleolus of dynamic, large-scale RSP game.

# 6. CONCLUSIONS

Ridesharing services, whose aim is to gather travelers with similar itineraries and compatible schedules, are able to provide substantial environmental and social benefits through reducing the use of private vehicles. When the operations of a ridesharing system is optimized, it can also save travelers a significant amount of transportation cost. The economic benefits associated with ridesharing in turn attract more travelers to participate in ridesharing services and thereby improve the utilization of transportation infrastructure capacity.

The first part of the dissertation formally defined the large-scale ridesharing optimization problem (RSP), characterized its complexity and discussed its relation to classic relevant problems such as TSP and VRP. A mixed-integer program is developed to solve RSP to optimality. Since RSP is NP-hard, heuristic algorithms are then developed to efficiently solve larger instances of RSP. The quality of heuristic solutions are evaluated by using the optimal matching solutions (see Wang, 2013) and the MIP solutions as benchmarks. Experimental results showed that adopting ridesharing can save a significant amount of travelers' cost and vehicle trips. It was also found that solutions produced by heuristic are good-enough approximations of the optimum and outperformed the matching solution by a non-trivial margin.

From a societal perspective, the RSP models can provide a ridesharing plan that minimizes the system-wide travel cost. However, the system-level optimal solution might not completely align with individual participants' interest. The second part of this dissertation formulates the fair cost allocation problem in ridesharing through the lens of cooperative game theory. An algorithm based on coalition generation techniques is developed to find the nucleolus and approximate nucleolus of the game.

This algorithm utilizes a master-slave problem solving structure with the subproblem explicitly formulated. Different subproblems are used in different situations. Experiments showed that this algorithm can save a significant amount of computational resources compared to the enumeration method.

Future research directions might include (1) developing heuristic or algorithms to solve the RSP MIP model more efficiently and (2) investigating other solution concepts in cooperative game theory such as the Shapley value. Anticipating that a dynamic large-scale ridesharing system might emerge in the foreseeable future, it is not computationally practical to find the optimal solution repeatedly when agents consistently join and leave the system. So it is of interest to study nearly-fair cost allocation schemes under a nearly-optimal ridesharing plan.

# BIBLIOGRAPHY

Claudia Archetti, Luca Bertazzi, and M. Grazia Speranza. Reoptimizing the traveling salesman problem. *Networks*, 42(3):154–159, 2003. ISSN 1097-0037. doi: 10.1002/net.10091.

Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45:753–782, September 1998. ISSN 0004-5411.

Egon Balas. The prize collecting traveling salesman problem. *Networks*, 19(6):621–636, 1989. ISSN 1097-0037. doi: 10.1002/net.3230190602. URL `http://dx.doi.org/10.1002/net.3230190602`.

M. L. Balinski and R. E. Quandt. On an integer program for a delivery problem. *Operations Research*, 12(2):300–304, 1964. doi: 10.1287/opre.12.2.300. URL `http://dx.doi.org/10.1287/opre.12.2.300`.

Gerardo Berbeglia, Jean-Franois Cordeau, and Gilbert Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8 – 15, 2010.

Ann Melissa Campbell and Martin Savelsbergh. Efficient insertion heuristics for vehicle routing and scheduling problems. *Transportation Science*, 38(3):369–378, 2004. doi: 10.1287/trsc.1030.0046.

Shuchi Chawla, Jason D. Hartline, and Robert Kleinberg. Algorithmic pricing via virtual valuations. In *Proceedings of the 8th ACM conference on Electronic commerce*, EC '07, pages 243–251, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-653-0. doi: 10.1145/1250910.1250946. URL `http://doi.acm.org/`

10.1145/1250910.1250946.

N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.

EdwardH. Clarke. Multipart pricing of public goods. *Public Choice*, 11(1):17–33, 1971. ISSN 0048-5829. doi: 10.1007/BF01726210. URL http://dx.doi.org/10.1007/BF01726210.

Jean-Francois Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54(3):573–586, 2006.

Jean-Franois Cordeau and Gilbert Laporte. The dial-a-ride problem (darp): Variants, modeling issues and algorithms. *4OR: A Quarterly Journal of Operations Research*, 1:89–101, 2003.

Cristian E. Cortes, Martin Matamala, and Claudio Contardo. The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *European Journal of Operational Research*, 200(3):711 – 724, 2010.

D.J. Dailey, D. Loseff, and D. Meyers. Seattle smart traveler: dynamic ridematching on the world wide web. *Transportation Research Part C: Emerging Technologies*, 7(1):17 – 32, 1999. ISSN 0968-090X. URL http://www.sciencedirect.com/science/article/pii/S0968090X99000078.

Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992. doi: 10.1287/opre.40.2.342. URL http://dx.doi.org/10.1287/opre.40.2.342.

Shahar Dobzinski. An impossibility result for truthful combinatorial auctions with

submodular valuations. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, STOC '11, pages 139–148, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0691-1. doi: 10.1145/1993636.1993656. URL http://doi.acm.org/10.1145/1993636.1993656.

Shahar Dobzinski and Jan Vondrak. The computational complexity of truthfulness in combinatorial auctions. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, pages 405–422, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1415-2. doi: 10.1145/2229012.2229044. URL http://doi.acm.org/10.1145/2229012.2229044.

I. Dragan. A procedure for finding the nucleolus of a cooperativen person game. *Zeitschrift für Operations Research*, 25(5):119–131, 1981. ISSN 0340-9422. doi: 10.1007/BF01919297. URL http://dx.doi.org/10.1007/BF01919297.

Shaddin Dughmi, Tim Roughgarden, and Qiqi Yan. From convex optimization to randomized mechanisms: toward optimal combinatorial auctions. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, STOC '11, pages 149–158, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0691-1. doi: 10.1145/1993636.1993657. URL http://doi.acm.org/10.1145/1993636.1993657.

Irina Dumitrescu, Stefan Ropke, Jean-François Cordeau, and Gilbert Laporte. The traveling salesman problem with pickup and delivery: polyhedral results and a branch-and-cut algorithm. *Mathematical Programming*, 121(2):269–305, 2010.

B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2007. cited By (since 1996) 206.

Stefan Engevall, Maud Göthe-Lundgren, and Peter Värbrand. The heterogeneous vehicle-routing game. *Transportation Science*, 38(1):71–85, February 2004. ISSN 1526-5447. doi: 10.1287/trsc.1030.0035.

European Environmental Agency. Indicator: Occupancy rates of passenger vehicles. Technical report, European Environmental Agency, 2005.

Masabumi Furuhataa, Maged Dessouky, Fernando Ordonez, Marc-Etienne Brunet, Xiaoqing Wang, and Sven Koenig. Ridesharing: the state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 2013.

P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859, 1961. doi: 10.1287/opre.9.6.849. URL http://dx.doi.org/10.1287/opre.9.6.849.

Andrew V. Goldberg, Jason D. Hartline, Anna R. Karlin, Michael Saks, and Andrew Wright. Competitive auctions. *Games and Economic Behavior*, 55(2):242 – 269, 2006. ISSN 0899-8256. doi: 10.1016/j.geb.2006.02.003. URL http://www.sciencedirect.com/science/article/pii/S0899825606000303. ¡ce:title¿Mini Special Issue: Electronic Market Design¡/ce:title¿.

Maud Göthe-Lundgren, Kurt Jörnsten, and Peter Värbrand. On the nucleolus of the basic vehicle routing game. *Mathematical Programming*, 72(1):83–100, 1996. ISSN 0025-5610. doi: 10.1007/BF02592333. URL http://dx.doi.org/10.1007/BF02592333.

Theodore Groves. Incentives in teams. *Econometrica*, 41(4):pp. 617–631, 1973. ISSN 00129682. URL http://www.jstor.org/stable/1914085.

Åsa Hallefjord, Reidun Helming, and Kurt Jørnsten. Computing the nucleolus when

the characteristic function is given implicitly: A constraint generation approach. *International Journal of Game Theory*, 24(4):357–372, 1995. ISSN 0020-7276. doi: 10.1007/BF01243038. URL http://dx.doi.org/10.1007/BF01243038.

Jason D. Hartline and Tim Roughgarden. Optimal mechanism design and money burning. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, pages 75–84, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-047-0. doi: 10.1145/1374376.1374390. URL http://doi.acm.org/10.1145/1374376.1374390.

Jason D. Hartline and Tim Roughgarden. Simple versus optimal mechanisms. In *Proceedings of the 10th ACM conference on Electronic commerce*, EC '09, pages 225–234, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-458-4. doi: 10.1145/1566374.1566407. URL http://doi.acm.org/10.1145/1566374.1566407.

Jang-Jei Jaw, Amedeo R. Odoni, Harilaos N. Psaraftis, and Nigel H.M. Wilson. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B: Methodological*, 20(3):243 – 257, 1986. ISSN 0191-2615. doi: http://dx.doi.org/10.1016/0191-2615(86)90020-2. URL http://www.sciencedirect.com/science/article/pii/0191261586900202.

Ece Kamar and Eric Horvitz. Collaboration and shared plans in the open world: studies of ridesharing. In *Proceedings of the 21st international jont conference on Artifical intelligence*, IJCAI'09, pages 187–194, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

RichardM. Karp. Reducibility among combinatorial problems. In RaymondE. Miller, JamesW. Thatcher, and JeanD. Bohlinger, editors, *Complexity of Computer*

*Computations*, The IBM Research Symposia Series, pages 85–103. Springer US, 1972. ISBN 978-1-4684-2003-6. doi: 10.1007/978-1-4684-2001-2_9. URL http://dx.doi.org/10.1007/978-1-4684-2001-2_9.

Alexander Kleiner, Bernhard Nebel, and Vittorio A. Ziparo. A mechanism for dynamic ride sharing based on parallel auctions. In Toby Walsh, editor, *IJCAI*, pages 266–272. IJCAI/AAAI, 2011. ISBN 978-1-57735-516-8.

A. Kopelowitz. *Computation of the Kernels of Simple Games and the Nucleolus of N-person Games*. Defense Technical Information Center HEBREW UNIV JERUSALEM (Israel) DEPT OF MATHEMATICS, 1967. URL http://books.google.com/books?id=sW6WNwAACAAJ.

J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981. ISSN 1097-0037. doi: 10.1002/net.3230110211.

Stefano Leonardi and Tim Roughgarden. Prior-free auctions with ordered bidders. In *Proceedings of the 44th symposium on Theory of Computing*, STOC '12, pages 427–434, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1245-5. doi: 10.1145/2213977.2214018. URL http://doi.acm.org/10.1145/2213977.2214018.

Quan Lu and Maged Dessouky. An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science*, 38:503–514, November 2004. ISSN 1526-5447.

Quan Lu and Maged M. Dessouky. A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. *European Journal of Operational Research*, 175(2):672 – 687, 2006. ISSN 0377-2217. doi: DOI:

10.1016/j.ejor.2005.05.012. URL http://www.sciencedirect.com/science/article/B6VCT-4GSJXHP-4/2/6e01c6fdd1898b873affe8f44c47152a.

Wei Lu, Lu Lu, and L. Quadrifoglio. Scheduling multiple vehicle mobility allowance shuttle transit (m-mast) services. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 125–132, Oct 2011. doi: 10.1109/ITSC.2011.6083083.

M. Maschler, B. Peleg, and L. S. Shapley. Geometric properties of the kernel, nucleolus, and related solution concepts. *Mathematics of Operations Research*, 4 (4):303–338, 1979. doi: 10.1287/moor.4.4.303. URL http://dx.doi.org/10.1287/moor.4.4.303.

C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329, October 1960. ISSN 0004-5411. doi: 10.1145/321043.321046. URL http://doi.acm.org/10.1145/321043.321046.

Catherine Morency. The ambivalence of ridesharing. *Transportation*, 34(2):239–253, 2007. ISSN 0049-4488. doi: 10.1007/s11116-006-9101-9. URL http://dx.doi.org/10.1007/s11116-006-9101-9.

Roger B. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981. doi: 10.1287/moor.6.1.58. URL http://mor.journal.informs.org/content/6/1/58.abstract.

Roger B. Myerson and Mark A. Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 29(2):265–281, April 1983. URL http://ideas.repec.org/a/eee/jetheo/v29y1983i2p265-281.html.

Noam Nisan and Amir Ronen. Algorithmic mechanism design (extended abstract). In

*Proceedings of the thirty-first annual ACM symposium on Theory of computing*, STOC '99, pages 129–140, New York, NY, USA, 1999. ACM. ISBN 1-58113-067-8. doi: 10.1145/301250.301287. URL http://doi.acm.org/10.1145/301250.301287.

Noam Nisan and Amir Ronen. Computationally feasible vcg mechanisms. In *Proceedings of the 2nd ACM conference on Electronic commerce*, EC '00, pages 242–252, New York, NY, USA, 2000. ACM. ISBN 1-58113-272-7. doi: 10.1145/352871.352898. URL http://doi.acm.org/10.1145/352871.352898.

Christos Papadimitriou, Michael Schapira, and Yaron Singer. On the hardness of being truthful. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '08, pages 250–259, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3436-7. doi: 10.1109/FOCS.2008.54. URL http://dx.doi.org/10.1109/FOCS.2008.54.

Harilaos N. Psaraftis. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14(2): 130–154, 1980. doi: 10.1287/trsc.14.2.130.

Harilaos N. Psaraftis. An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation Science*, 17(3):351–357, 1983. doi: 10.1287/trsc.17.3.351.

Luca Quadrifoglio, Maged Dessouky, and Kurt Palmer. An insertion heuristic for scheduling mobility allowance shuttle transit (mast) services. *Journal of Scheduling*, 10:25–40, 2007. ISSN 1094-6136.

Luca Quadrifoglio, Maged M. Dessouky, and Fernando Ord-ez. Mobility allowance shuttle transit (mast) services: Mip formulation and strengthening with logic

constraints. *European Journal of Operational Research*, 185(2):481 – 494, 2008.

Daniel J. Rosenkrantz, Richard E. Stearns, Philip M. Lewis, and II. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563–581, 1977. doi: 10.1137/0206041.

A. Santos, N. McGuckin, H.Y. Nakamoto, D. Gray, and S. Liss. Summary of travel trends: 2009 national household travel survey. Technical report, U.S. Department of Transportation Federal Highway Administration, 2011.

M. W. P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):17–29, 1995. doi: 10.1287/trsc.29.1.17.

David Schmeidler. The nucleolus of a characteristic function game. *SIAM Journal on Applied Mathematics*, 17(6):pp. 1163–1170, 1969. ISSN 00361399. URL `http://www.jstor.org/stable/2099196`.

D. Schrank and T. Lomax. 2007 urban mobility report. Technical report, Texas Transportation Institute, 2007.

David Schrank, B. Eisele, and T. Lomax. Tti's 2012 urban mobility report. Technical report, Texas A&M Transportation Instutite, 2012.

Thomas R. Sexton and Lawrence D. Bodin. Optimizing single vehicle many-to-many operations with desired delivery times: I. scheduling. *Transportation Science*, 19(4):378–410, 1985. doi: 10.1287/trsc.19.4.378.

Lloyd S. Shapley. On balanced sets and cores. *Naval Research Logistics Quarterly*, 14(4):453–460, 1967. ISSN 1931-9193. doi: 10.1002/nav.3800140404. URL `http://dx.doi.org/10.1002/nav.3800140404`.

Dusan Teodorovic and Gordana Radivojevic. A fuzzy logic approach to dynamic dial-

a-ride problem. *Fuzzy Sets and Systems*, 116(1):23 – 33, 2000. ISSN 0165-0114. doi: DOI:10.1016/S0165-0114(99)00035-4.

Hal R. Varian. Position auctions. *International Journal of Industrial Organization*, 25(6):1163 – 1178, 2007. ISSN 0167-7187. doi: 10.1016/j.ijindorg.2006.10.002.

William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961. ISSN 1540-6261. URL `http://dx.doi.org/10.1111/j.1540-6261.1961.tb02789.x`.

J. von Neumannn and O. Morgenstern. *Theory of games and economic behavior*. Princeton University Press, Princeton, New Jersey, 1944.

Xing Wang. *Optimizing ride matches for dynamic ride-sharing systems*. PhD thesis, Georgia Institute of Technology, May 2013.