Car2Go: On-Demand Mobility in a Digitalized Environment

Group 4: Car determines optimal place to pick up customer from

Seminar paper

Students:

Paul Braitenberg 01515188

Leon Rhein 12240073

Faisal Schwab 11938102

Alexander Klugsberger 11702381

Kamil Andrzejewski 12243212

universität wien

Abstract

This paper explores the enhancement of car-sharing services, exemplified by Car2Go, now operating under ShareNow. The key challenge addressed is the optimization of customer pick-up locations, a vital aspect of modern urban mobility in the context of digitalization. Our project develops a system that integrates real-time urban data to determine the most efficient pick-up spots, factoring in variables like traffic, weather, and local events. The methodology involves a simulated city map with 54 interconnected nodes, serving as the basis for algorithmic testing and decision-making processes. The system utilizes Python for backend operations, Dijkstra's algorithm for effective pathfinding, and a rule-based decision-making process implemented in Prolog. This combination allows for a comprehensive analysis and response to varying urban scenarios.

Our system's architecture, comprising a web service interface, a pathfinding module, and a decision-making module, ensures rapid adaptation to changing city dynamics. The integration of Python and Prolog strengthens the system's capability to not only identify the nearest vehicle but also select the most contextually appropriate pick-up location.

This study contributes to the evolving field of on-demand mobility, offering insights into the practical application of digital solutions in optimizing urban transportation services. The paper provides an in-depth discussion of the problem, the employed methodologies, our thoughts behind the chosen technologies, and potential approaches for future enhancements, including the application of machine learning techniques for predictive urban mobility analysis.

Table of contents

1. Introduction..... 3
2. Problem statement..... 4
3. Project Description..... 5
4. Methodology..... 5
5. Relevant Concepts and Parameters..... 7
6. Technology and Libraries Used..... 8
7. Discussion on Adequateness of Technology..... 9
8. Conclusion..... 9

1. Introduction

The rise of digital technology has greatly changed many industries. Since a long time it has reached a point where it is also affecting our everyday life, for example in the way we move around in cities. Nowadays, services that offer rides on demand have become very important in city transportation. They are flexible, quick, and give a personalized way of traveling. This paper looks at a project that is right at the heart of these on-demand ride services and digital advances, focusing specifically on a service similar to the Car2Go service.

Car2Go is a well-known company in the car-sharing field, offering many vehicles that people can rent for short periods. Back in 2019 the company merged with DriveNow to form ShareNow, which is nowadays the most known company in the car-sharing field in Vienna.[1] This service works through an online platform where customers can find, reserve, and use cars. However, one big challenge in this area could be figuring out the best places to pick up customers.
On-demand services are all about giving users fast, easy, and most important, efficient options. With the known technologies, users can usually pick one of the available cars on the city map. They just have to walk to the place where the car is located. In other scenarios, for example with regard to future technologies such as self-driving cars, it would be conceivable for the car to also drive to the customer, or for another location to be agreed where the customer and vehicle can meet.

Our project tackles this challenge. We're creating a system that figures out the best pick-up spots by considering many changing factors like traffic, weather, roadwork, and local events. This system uses a fictional city map and different technologies, like Python for the main programming tasks, Dijkstra’s algorithm to find the shortest routes, and a rule-based system in Prolog for making decisions.

The goal of this paper is to give a full overview of the project. We want to explain the problem, describe the methods we used, talk about why certain factors matter, and take a close look at how well the technologies we used worked. By doing this, the paper will hopefully help to understand more about on-demand ride services in today's digital world,

[1] Leadersnet: *Neue App ShareNow bündelt Dienste von Car2Go und DriveNow*. Online under: https://www.leadersnet.at/news/40219,carsharing-neue-app-share-now-buendelt-dienste-von-car2go-und.html

especially how they can work better and improve the experience for users like those of Car2Go or ShareNow.

2. Problem statement

In the context of car sharing company's operations, a primary issue that emerges is the determination of the most efficient and convenient pick-up locations for customers. This challenge involves various dynamic and static environmental factors, which makes solving the problem very complex. The ShareNow service model, which depends on providing instant mobility solutions, is heavily reliant on its ability to efficiently connect customers with the nearest available vehicle. However, the 'nearest' vehicle is not always the 'optimal' one. The optimal pick-up location is a multifaceted concept influenced by several parameters beyond physical proximity.

The importance of determining the optimal pick-up location lies in multiple facets of customer experience and operational efficiency. From a customer's perspective, the ideal pick-up point should not only be easily accessible but also consider factors such as traffic conditions, weather, and road safety, and could also be dependent on the destination where the customer wants to go. For instance, a closer vehicle located in a traffic-congested area might not be as desirable as a slightly farther one in a less congested area. Similarly, during adverse weather conditions, a sheltered pick-up point, even if slightly distant, could significantly enhance customer convenience and safety.

From an operational standpoint, strategically optimized pick-up locations can lead to better fleet management, reduced waiting times, and lower operational costs. Efficiently managing the vehicle fleet not only improves the turnaround time for each vehicle but also minimizes the deterioration resulting from unnecessary travel. Additionally, it can contribute to a balanced distribution of vehicles across the service area, preventing clustering in certain regions while others face scarcity.

Therefore, this project is not just about finding the nearest vehicle for a customer, but determining the optimal pick-up location taking into account a variety of variables. This requires careful analysis and integration of real-time data on traffic, weather, local events and road conditions, among other things.

3. Project Description

The project is set against the framework of an increasingly digitized world, where on-demand mobility services are becoming an integral part of urban transportation. In such an environment, the ability to rapidly and efficiently connect customers with transportation services is very important. The project leverages digital technologies to address these needs, focusing on Car2Go like services that represent the modern demand for instant, accessible, and flexible transportation solutions.

The core objective of this project for our group is to enhance the Car2Go service by developing an intelligent system that determines the most optimal pick-up locations for customers in a fictional city map, based on not only the distance between several available cars and the customer, but on various parameters and conditions, using Prolog and First Order Logic rules. This system is designed to consider a variety of dynamic environmental and situational factors, thus ensuring that the pick-up locations are not only convenient and accessible but also contextually appropriate.

4. Methodology

Our project's system is built to be both strong and adaptable, capable of processing information in real time and providing quick responses. It integrates three key functionalities: a web-service interface, pathfinding, and rule-based decision-making.
We've developed the web service interface using Flask, a Python framework known for its simplicity and effectiveness. This interface is the main point of interaction for users. It manages all the incoming and outgoing requests, simplifying the complex operations that occur behind the scenes.

For the pathfinding aspect, we utilize Dijkstra's algorithm. This well-established method is essential for identifying the shortest route between two points, which is particularly important for pinpointing the nearest available vehicle to a customer. By calculating the minimum distance between the customer's location and the available vehicles, we can efficiently determine the most potential pick-up points.

The decision-making part of the system is where we really get into the details of determining the best pick-up location. Developed in SWI-Prolog, this module employs a rule-based system that considers various factors such as traffic conditions, weather, local events or road

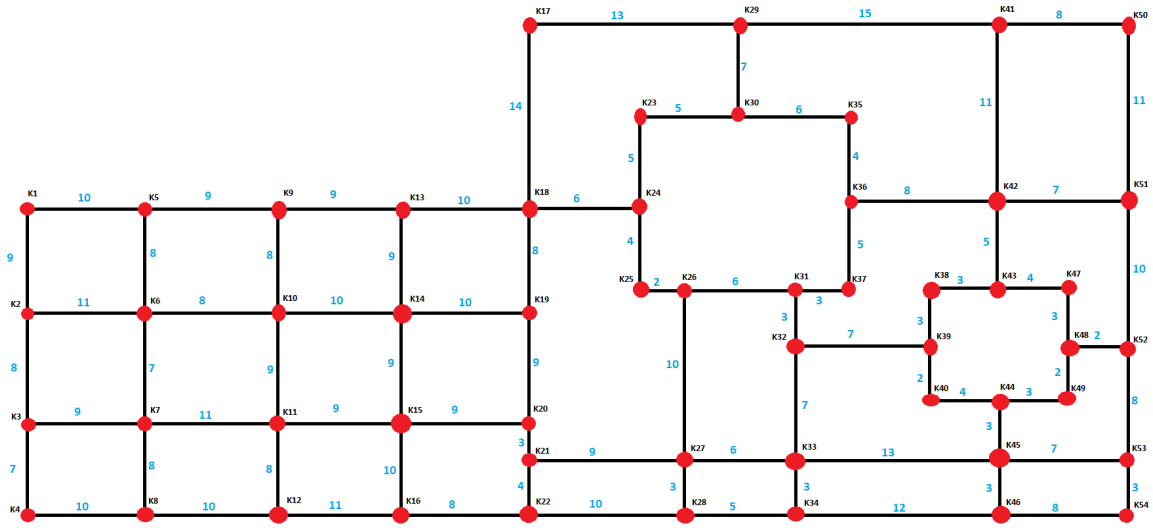
construction works. These rules are crafted using first-order logic, enabling our system to make smart decisions about where to pick up customers.

A critical aspect of our system's architecture is the integration of Python and Prolog. While Python is responsible for the web-service interface and handling the pathfinding algorithm, Prolog takes charge of the rule-based decision-making process. This harmonious integration ensures that our system is not only efficient in its operations but also adept at making complex decisions, leading to an enhanced user experience and optimized service performance.

In addition to the system's core functionalities, an essential part of our project is the creation of a fictional city map, which forms the basis for our simulations and decision-making processes. This map comprises 54 nodes, representing intersections or key points within the city, and the weighted edges that connect these nodes, symbolizing the streets and their length.
This fictional city layout is integral to our project as it provides a controlled environment where we can test and refine our algorithms and rules. By designing this map, we have a comprehensive framework that is not too simple but also not too complex. It allows us to simulate various scenarios and conditions that a real city would experience. It helps us in understanding how different factors like traffic flow, road closures, and event locations can affect vehicle movement and pick-up locations.

All assumptions and rules within our system are based on this city map. When applying Dijkstra's algorithm for pathfinding, it uses the connections between these nodes to calculate the shortest paths. Similarly, the rule-based decision-making process in Prolog takes into account the layout and characteristics of this map to determine the optimal pick-up locations. For instance, if there's a large event happening near a particular node, our system can predict and adjust for the increased traffic in that area, suggesting alternative pick-up locations that avoid congestion.

By using this fictional city map, we can thoroughly test our system in a variety of simulated conditions, ensuring that it's not only theoretically sound but also practically viable in a real-world setting. This approach allows us to refine our system in a controlled environment, enhancing its reliability and effectiveness before any real-world application.



2

5. Relevant Concepts and Parameters

Several key parameters are so far taken into consideration in the decision-making process:

Weather Conditions: Weather plays a significant role in determining the suitability of a pick-up location. For instance, in bad weather, a covered or indoor pick-up point might be preferred. Some of the 54 nodes could be defined as indoor pick-up points, or certain streets could be defined as suboptimal in bad weather conditions.

Traffic Conditions: Real-time traffic data helps in assessing the accessibility of a location. High traffic areas might be avoided to reduce waiting time and improve customer experience. We could define, for instance, certain times where certain nodes should be avoided because of high traffic, using FOL rules.

Local Events: Events can significantly alter the usual traffic patterns and accessibility of areas. The system takes into account any ongoing events that might impact pick-up locations. Our city map, at this point, is only a network consisting of nodes and edges. In a later stage of the implementation we will define certain city areas within that city map that should be avoided when certain local events take place. For example, we could define a central park area, where regular demonstrations take place. If the customer wants to order a car during such a local event, certain nodes should be avoided and not considered when figuring out an optimal route from the car to the customer.

[2] Our fictional city map consists of 54 nodes with connecting edges.

Road Conditions: Roadworks or closures can affect the viability of certain routes in a similar way. The system dynamically adjusts its recommendations based on these conditions.

By integrating and analyzing these parameters, the system can make informed decisions, ensuring that the selected pick-up location is not only the nearest but also the most convenient and practical under the current conditions.

6. Technology and Libraries Used

Python (Flask, Redis): Python serves as the core programming language for the project, known for its simplicity and robustness. Flask, a lightweight Python web framework, is used for creating the web-service interface. It is chosen for its ease of use and flexibility in handling web requests and responses. Redis, an in-memory data structure store, is employed as a caching and message broker tool, enhancing the performance of the system.

SWI-Prolog: Prolog, specifically SWI-Prolog, is utilized for the rule-based decision-making component. Its powerful pattern-matching and logical inference capabilities make it ideal for processing complex rule sets and determining optimal pick-up locations.

Docker: Docker is used for containerizing the application, ensuring consistent environments for development, testing, and deployment. This addresses the challenge of "it works on my machine" and facilitates smooth deployment on any system.

The choice of these technologies is driven by the need for a reliable, scalable, and maintainable system. Python's widespread adoption and extensive library support make it an excellent choice for backend development. Flask's minimalistic but extensible nature allows for quick development of web services. Redis enhances the system's responsiveness, crucial for real-time applications. SWI-Prolog is specifically chosen for its efficiency in handling logical rules and queries, which are central to the decision-making process. Docker's containerization ensures that the system remains platform-independent and alleviates deployment challenges.

7. Discussion on Adequateness of Technology

In our project, we have chosen a combination of Python, Prolog, and Docker for our technology stack, and each has played a crucial role. Python is a user-friendly programming language that, together with Flask, a simple yet powerful web development framework, forms the backbone of our web service. It's known for its ease of use and flexibility, making it a popular choice for developers. Prolog, on the other hand, excels in logical reasoning and decision-making. It's particularly useful in scenarios where complex rules and conditions need to be evaluated. Docker complements these technologies by ensuring that our application can be deployed consistently across different environments. It also helps us scale our service as needed, accommodating more users or handling more data without a drop in performance.

Despite their strengths, these technologies are not without their limitations. Prolog, while excellent for logical operations, is not as well-suited for tasks that require heavy data processing. This can be a drawback in situations where the application needs to handle large amounts of data quickly and efficiently. Python, although versatile and widely used, can also face challenges with performance, especially in large-scale data processing scenarios. This might lead to slower response times and affect the overall user experience.
To improve our project in the future, we could explore integrating additional technologies that are optimized for handling big data. Another area for improvement could be in the realm of decision-making logic. Investigating alternative rule-processing engines that might offer more efficiency or better integration with Python could refine our system's ability to make complex decisions more effectively.

8. Conclusion

Our project showcases a successful implementation of a technology-driven system for a car sharing service, specifically designed to optimize pick-up locations for Car2Go customers. We've achieved this by implementing and using the strengths of Python for flexibility, Prolog for advanced decision-making, and Docker for consistent deployment and scalability. This blend not only caters to current needs but also sets a foundation for future enhancements.

As we look ahead, integrating machine learning algorithms emerges as a promising step. This advancement could enable predictive analytics, making the system not only responsive but also anticipatory in meeting customer needs. It would also improve the system's ability to process large data volumes more efficiently. Exploring sophisticated rule-based systems is

another pathway to enhance our decision-making processes, particularly in complex scenarios involving numerous variables.

A significant upcoming challenge will be to define a sufficient number of rules for the system to realistically model complex scenarios without becoming overly complicated. This balance is crucial; too few rules might oversimplify real-world situations, while too many could make the system unwieldy and hard to maintain. Achieving this balance will require careful consideration and possibly the development of new methodologies or algorithms to manage the complexity effectively.

Additionally, the potential of this technology extends beyond car sharing services. Its adaptability to other on-demand services underscores its versatility and scalability, making it a valuable tool for diverse applications. While the current system is efficient and effective, the road ahead is filled with opportunities for growth and refinement. Future enhancements, especially in handling the complexity of rule definition and integration of new technologies, will likely open new horizons for its application across various sectors.