

Laurel Maisha Raven

---

**Optimizing Intersection Layouts to  
Improve Emergency Vehicle Travel  
Times using a Genetic Algorithm**

---





FAKULTÄT FÜR  
INFORMATIK

Intelligent Cooperative Systems  
Computational Intelligence

# Optimizing Intersection Layouts to Improve Emergency Vehicle Travel Times using a Genetic Algorithm

Bachelor Thesis

Laurel Maisha Raven

August 25, 2025

Supervisor: Prof. Dr.-Ing. habil. Sanaz Mostaghim

Advisor: Carlo Nübel, M.Sc.

**Laurel Maisha Raven:** *Optimizing Intersection Layouts to Improve Emergency Vehicle Travel Times using a Genetic Algorithm*  
Otto-von-Guericke Universität  
Intelligent Cooperative Systems  
Computational Intelligence  
Atlanta, Georgia, 2025.

---

# Abstract

Prompt arrival times for emergency vehicles is of the utmost importance when providing urgent medical care or responding to a fire. Although existing research shows improved arrival times for emergency vehicles (EVs) with the help of real-time approaches such as adaptive traffic light controls and dynamic path planning, solutions to improve travel times using offline-optimized intersection configurations are unexplored. In this research we show the effectiveness of using a genetic algorithm (GA) in determining optimal configuration of intersections with the goal of improving EV travel times. Specifically, a one-point and a uniform crossover are tested and assessed based on fitness over time and fitness of the final generation. These tests were performed on three different maps to test the effectiveness of the solution with different traffic amounts. The generated solutions are shown to decrease the travel times compared to the solutions emulating the real-world configurations. The robustness of the solutions was also tested with different traffic volumes. The impact on travel times for non-emergency vehicles (NEVs) was also assessed, although it was not a performance criteria.

---

# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction</b>                              | <b>2</b>  |
| 1.1      | Motivation . . . . .                             | 2         |
| 1.2      | Goals . . . . .                                  | 3         |
| 1.3      | Thesis Structure . . . . .                       | 4         |
| 1.4      | Use of AI . . . . .                              | 4         |
| <b>2</b> | <b>Theoretical Foundation</b>                    | <b>6</b>  |
| 2.1      | Evolutionary Algorithms . . . . .                | 6         |
| 2.2      | Genetic Algorithms . . . . .                     | 6         |
| 2.3      | Exploration vs. Exploitation . . . . .           | 11        |
| <b>3</b> | <b>Related Work</b>                              | <b>12</b> |
| 3.1      | City Planning . . . . .                          | 12        |
| 3.2      | Reducing EV Travel Times . . . . .               | 12        |
| 3.2.1    | Dynamic Path Finding . . . . .                   | 13        |
| 3.2.2    | Dynamic Traffic Signal Control . . . . .         | 14        |
| 3.2.3    | Research Gap . . . . .                           | 15        |
| <b>4</b> | <b>Methodology</b>                               | <b>16</b> |
| 4.1      | Overview of Genetic Algorithm Approach . . . . . | 16        |
| 4.2      | Software Used . . . . .                          | 16        |
| 4.3      | Data Sources and Preparation . . . . .           | 17        |
| 4.4      | File Editing During Runtime . . . . .            | 21        |
| 4.5      | Evolutionary Algorithm Design . . . . .          | 23        |
| 4.5.1    | Sampling and Individual Encoding . . . . .       | 23        |
| 4.5.2    | Parameter Selection . . . . .                    | 24        |
| 4.5.3    | Crossover Functions . . . . .                    | 24        |
| 4.5.4    | Mutation . . . . .                               | 25        |
| 4.5.5    | Fitness, Survival, and Selection . . . . .       | 25        |
| <b>5</b> | <b>Experiments and Results</b>                   | <b>32</b> |
| 5.1      | Experiments . . . . .                            | 32        |

|          |   |           |
|----------|---|-----------|
| 5.2      | Results . . . . .                                       | 32        |
| 5.2.1    | Comparison of EA Solution Fitness to Real-world Fitness | 33        |
| 5.2.2    | Evaluation of Different Crossovers . . . . .            | 34        |
| 5.2.3    | Influence of Traffic Volume . . . . .                   | 35        |
| 5.2.4    | Impact on Non-Emergency Vehicles . . . . .              | 37        |
| <b>6</b> | <b>Conclusions and Future Work</b>                      | <b>46</b> |
|          | <b>Bibliography</b>                                     | <b>48</b> |

---

# 1 Introduction

## 1.1 Motivation

When it comes to medical emergencies, fires, or similar incidents, every second counts. Extra time spent traveling to the scene of an accident or transporting injured people to a hospital can result in more lasting damage or even loss of life. According to the WHO, the ideal emergency response time (the interval between when an incident is reported and when an ambulance arrives) is under eight minutes (1). In some emergencies, the time window is even smaller, such as in cases of cardiac arrest, where CPR must be started in as little as four to six minutes before brain damage is irreversible (2).

In the case of building fires, a timely response is also crucial. In the US, the response time is required to be four minutes or less for at least 90% of cases (3). As response times increase from four to eleven minutes, fires are more frequently reported as involving the entire building rather than being contained to a single room. Longer response times are also associated with an increased likelihood of one-story residential fires being reported as causing “extreme” rather than “minor” damage, particularly within the four to fourteen minute response window (4). Quick response times to fires are therefore key to reduce harm to individuals and damage to infrastructure.

The chance of a fire starting, spreading beyond the room or structure of origin, and/or causing death and injury are especially high in certain areas. According to the Fire-Community Assessment Response Evaluation System (FireCARES), the city of Chicago, one of the areas evaluated in this thesis, is classified as high risk for fires, fire spread, and death and injury when compared to communities of similar size, with a total of 1,506 structural fires in 2023 (5). FireCARES labeled the city of South Bend, Indiana, another one of the areas examined in this thesis, as high risk of fire spread compared to communities classified as the same size (6).

As congestion in urban areas continue to rise, so does the risk of delays to EVs (3). Therefore it is imperative to design cities and infrastructure in a way that not only ensure timely arrivals for regular traffic, but especially for emergency vehicles.

In this thesis, we examine the possibility of solving this issue using a GA. GAs are generally suitable for a wide range of difficult to solve optimization problems (7). For problems such as the one addressed in this thesis, the search space was large enough that it was not realistic to simply iterate through all possible solutions and a more intelligent strategy was needed. Therefore an approach using a GA was chosen since it is better at finding an optimal solution within a large pool of possible solutions (8). At the time of writing, the application of GAs for intersection planning to improve EV travel times had not yet been explored, therefore this thesis aims to address this research gap.

## 1.2 Goals

The goal of this thesis is to explore and present the findings for the potential reduction of travel times for emergency vehicles with the implementation of an genetic algorithm, which optimizes an arrangement of intersections on a given map. Specifically, we will explore how much EV travel times can be improved and the impact of the improvements on regular traffic. For this, two different crossover methods will be tested and assessed: a one-point crossover and a uniform crossover. The one-point crossover allows the preservation larger groups of compatible intersections, while the uniform crossover can separate smaller groups of compatible intersections. These experiments will be performed on three maps with different layouts amounts of traffic to compare results based on map type and traffic volume. We also test the robustness of the best solutions to determine how well they perform with varying amounts of traffic.

The following research questions and hypotheses will be addressed in this thesis:

- RQ1: How can the evolutionary approach minimize travel times by optimizing intersection type layouts?
  - H01: The EA will find solutions that offer significant improvements in travel times compared to the baseline map configurations.
  - H11: The EA is not suitable for this problem and will find solutions with no significant advantage over the baseline map configurations.
- RQ2: How do different crossover operators influence the optimization?

- H02: There are significant differences in the travel time reduction and convergence time between the one point and the uniform crossover.
- H12: There are no significant differences in the travel time reduction and convergence time between the one-point and the uniform crossover.
- RQ3: How well do the solutions perform under different traffic amounts?
  - H03: The improvements offered by the EA will perform well for varying traffic amounts.
  - H13: The solutions will be overfit to the amount of traffic used to find solutions with.
- RQ4: How are the travel times for non-emergency vehicle (NEV) traffic impacted?
  - H04: NEVs will also experience a significant reduction in travel times.
  - H14: The new configurations will only offer an improvement for EVs and will be disadvantageous to NEV traffic.

## 1.3 Thesis Structure

First we will present the theoretical foundation, including information about evolutionary algorithms and specifically genetic algorithms needed to understand this thesis. Then we will explore related work in the field of city planning and reducing EV travel times. Next, we will go into detail about the methods and software used to create the proposed solution. Afterwards, we will explain the experiments that were conducted in more detail and evaluate the results. We conclude with a summary of findings and propose ideas for future work.

## 1.4 Use of AI

The large language model ChatGPT was used to generate the code used for this thesis. Specifically, GPT-4 Series, including 4.1 and 4.5, were used. The

author gave detailed specifications, requirements, and parameters as input and GPT performed the realization of these requirements into code. This code was primarily used to parse and edit the xml files for the simulation, as well as to implement the mutation and crossover functions, and to process simulation results and create graphs.

This was done for the sake of efficiency and simplification during the implementation phase and it allowed more time to focus on other tasks, such as research and evaluation. The use of ChatGPT for code generation still required extensive knowledge from the author about the tasks to be performed and the functionality and operation of the software and framework used. The generated code was tested, adapted, and verified by the author for correctness.

ChatGPT was also used for the Results section 5.2 to suggest approaches for comparing data based on the research questions and to generate the code for parsing and graphing the results. Based on the observed results, ChatGPT suggested explanations for the patterns found in the data. All suggested approaches and interpretations were critically reviewed by the author and only those that were consistent with the data and research objectives were included in the thesis.

ChatGPT was also used for the creation of graphics. The images 2.3 and 2.2 were created entirely using prompts given to ChatGPT. It was also used to make minor changes to the flowcharts and graphics, such as adding nodes and adjusting positioning, and to generate the layout for the tables based on the author's formatting instructions.

Additionally, ChatGPT was used to verify the proper citation of sources in cases where the source information was incomplete.

---

## 2 Theoretical Foundation

This chapter will explain the concept and functionality of evolutionary algorithms and any other theoretical information required to understand this thesis.

### 2.1 Evolutionary Algorithms

The term evolutionary algorithm (EA) is named such because it is based on Darwin's theory of evolution (9). An EA is a search algorithm that mimics biological evolution to find a solution to an optimization problem (10) (11). It is formally defined as a "collective term for all variants of (probabilistic) optimization and approximation algorithms that are inspired by Darwinian evolution. Optimal states are approximated by successive improvements based on the variation-selection-paradigm. Thereby, the variation operators produce genetic diversity and the selection directs the evolutionary search." (12) The term evolutionary algorithm refers to a family of techniques that match this description (10) and this paper will focus specifically on a type known as a genetic algorithm (GA).

### 2.2 Genetic Algorithms

Here is some common vocabulary required to understand a genetic algorithm (8) (10) (13). The biology analogy comes from (13):

- Individual: A single solution candidate to the problem. The terms individual and candidate are used interchangeability in this thesis.
- Chromosome: The encoding of a solution candidate. A chromosome is a sequence of genes.
- Gene: Part of the chromosome which encodes a particular trait. In biology, a gene would be responsible for a trait such as eye color.
- Allele: Possible values for a trait that a particular gene can have. In biology, the possible alleles for eye color would be blue, brown, hazel, etc.

- Population: A group of individuals.
- Starting population: The population of the first generation. This is normally chosen randomly.
- Generation/Iteration: The population at a point in time.
- Fitness: How good or "fit" a solution is. Fitness can describe a single individual or an entire population. Fitness can be based on a single output (single-objective) or multiple outputs (multi-objective).
- Termination criteria: The point at which the genetic algorithm stops. This can be due to a predefined number of iterations being reached or a solution being found which meets a minimum fitness.

Now that the basic vocabulary has been defined, we give a detailed description of each step involved in a GA. Figure 2.1 is re-created based on pymoo's GA documentation with slight wording adjustments for clarity and illustrates the basic GA cycle (14). It is worth noting that some literature simplifies the survival selection and mating selection steps into a single step, where the selection process involves directly selecting individuals for reproduction (8). This thesis uses the more elaborate definition in which survival selection and mating selection are separate steps, since this is how the process is described in the pymoo documentation (15).

**Initialize Population** The first step in an evolutionary algorithm is to choose a starting population, which is the very first generation of individuals and the starting point for all future generations. This is usually done by randomly assigning genes to each individual to ensure a diverse selection (10) (16).

**Evaluation** Evaluation refers to assessing the fitness of a solution using the criteria of the problem to be solved, in other words, assessing how well an individual solves the problem. In this step we evaluate the quality of the individuals within the current population and assign a fitness value to each individual based on how good it is (10).

**Survival Selection** Survival selection refers to how individuals are chosen for the next generation. A simple approach is to sort individuals by their fitness and to choose the individuals with the highest ranking (16). This subset survives to be taken into the next generation and the others will be discarded. This method is called survival of the fittest because it

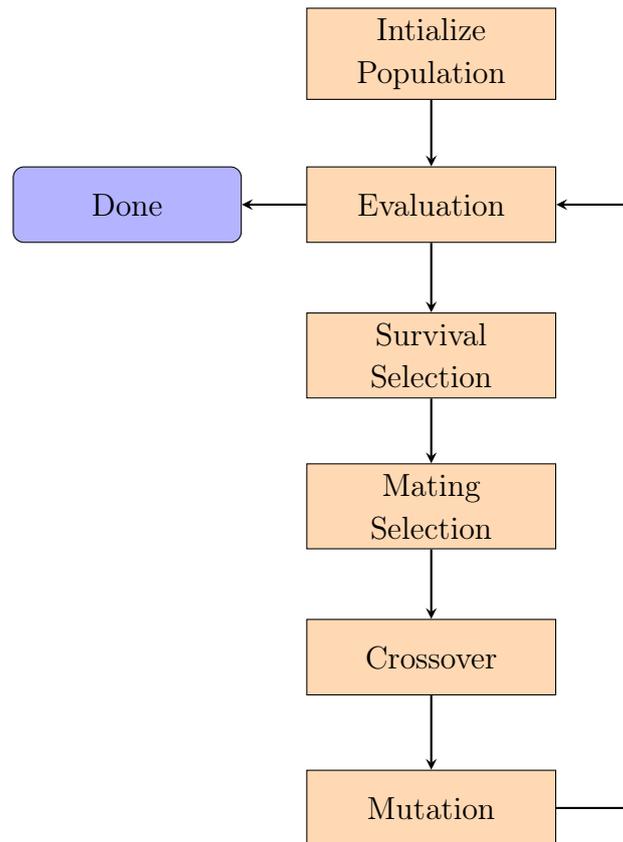


Figure 2.1: The cycle of a genetic algorithm, adapted from (14). The flow charts used in this thesis were edited with the help of AI, as disclosed in section 1.4.

mimics this concept as seen in nature. While survival of the fittest is a common method, the strategies described in *mating selection* can also be applied to *survival selection* (7).

**Mating Selection** Once the surviving individuals have been chosen, a subset of those will be selected to breed, which is called *mating selection* (8). There are a number of mating selection approaches, and the ideal approach depends on the problem to be solved. Some methods include:

- **Roulette Wheel Selection:** Roulette wheel selection is named such because it can be compared to spinning a roulette wheel, where each individual is assigned a different sized section of the wheel based on their fitness. With roulette wheel selection, the probability of an individual being selected is proportional to the fitness of the individual, and more

fit individuals have a larger slice of the wheel. An issue can occur with roulette wheel selection when there the fitness among individuals is disproportionate. This makes it very difficult for individuals that are not the very best to be chosen (10).

- **Rank Selection:** In rank selection, each individual is assigned a rank based on fitness and each rank is then assigned a probability. These probabilities are then used instead of the in roulette wheel selection as an alternative that addresses the issue of disproportionate fitness (13) (10).
- **Tournament Selection:** In tournament selection, a number of individuals compete against each other and the individual(s) with the highest fitness are added to the mating pool. This offers the advantage that individuals with lower fitness still have a chance of being selected to breed, since fitness only dictates the winner of the tournament and all individuals have an equal change of competing (10) (7).

**Crossover** After the individuals have been selected with mating selection, they are used to create new offspring. One of the methods used to create new offspring is called the crossover function. This takes two or more parents from the *mating selection* step and recombines them to create children which are composed of parts from the parents. The goal is to create children with optimal features from both parents. Here are a few common crossover methods. There are a variety of crossover variants, but the one-point (single point) and uniform crossover were the ones used in the experiments. (10).

- **One-point Crossover:** In one-point crossover, a random point between two genes is chosen as the crossover point. This point divides the chromosome into two parts. The first part of one parent is then combined with the second part of the second parent, and vice versa (10). An example of a simple one-point crossover is shown in figure 2.2 .
- **$n$ -point Crossover:** The  $n$ -point crossover functions similarly to one-point crossover, except that  $n$  crossover points are chosen instead of one (10).
- **Uniform Crossover:** A uniform crossover functions by randomly choosing genes from each parent with a certain probability (10).

**Mutation** The final step in diversifying the next generation is the mutation function. The mutation function works by creating small changes within

an individual. A standard mutation is used for the experiments in this thesis. Some mutation functions include:

- **Standard Mutation:** Standard mutation is used to replace the allele of a single gene with a new one. This is especially useful when a particular allele is missing from the starting population, as it is the only way to introduce that gene to the population later on (10). An example of a standard mutation is shown in figure 2.3.
- **Shift:** The shift mutation moves a subsequence of genes to a new location by shifting them (10).
- **Inversion:** An inversion works by reversing the order of alleles within a subsequence (10).
- **Permutation:** A permutation randomly changes the order of alleles within a subsequence (10).

The shift, inversion, and permutation mutations are especially useful in cases where it is necessary to keep the same set of alleles in each chromosome (10).

**Termination** This cycle is repeated from the evaluation step until the termination criteria is met. There are a number of possible termination conditions including: an individual with an acceptable fitness being found, the fitness of the best individual not increasing significantly after a certain number of generations, or a pre-defined number of iterations being reached (7) (10).

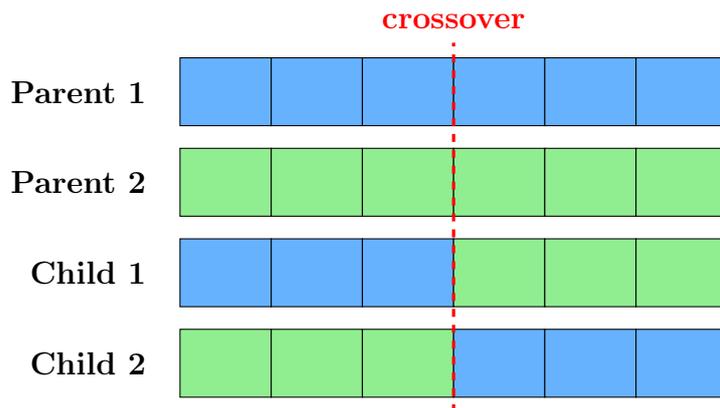


Figure 2.2: Example of a one-point crossover. Image adapted from (7) and created with ChatGPT as disclosed in 1.4.

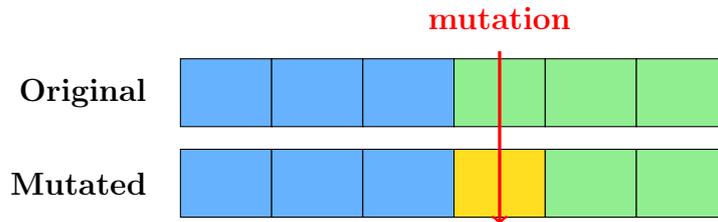


Figure 2.3: Example of a standard mutation. Image adapted from (10) and created with ChatGPT as disclosed in 1.4.

## 2.3 Exploration vs. Exploitation

The success of an evolutionary algorithm depends greatly on the initial selection of parameters. The selection of optimal values (number of iterations, population size, crossover rate, etc.) is a science within itself and goes beyond the extent of this paper. The ideal parameters depend on the problem to be solved so there is no "one size fits all" approach to this. However, there are still general principles that can be followed to take some of the randomness and guessing out of the procedure. A good approach is to balance exploration and exploitation. Exploration describes testing the search space for previously unseen solutions. Exploitation describes using and refining solutions that have already been found to be good. Both are key in the success of the algorithm but a balance between the two needs to be found (10) (13).

---

## 3 Related Work

This chapter will cover related work in the areas of reducing EV times and city planning. The work in reducing travel times can be divided based on the methods applied, namely dynamic path-finding and dynamic traffic signals. At the end of the section we will discuss the current research gap and how this thesis offers a new approach.

### 3.1 City Planning

While there is existing work exploring the use of EAs in city planning, we were unable to find research examining the use of an EA to change or plan cities at the intersection level. In the paper (17), a genetic algorithm was used for the planning of city layouts for new cities. However this was not concerned with the specifics of intersections layouts or traffic flow, but rather the general layout of a city and the optimal placement of different district types (residential, commerce, industrial, etc.) on a grid-map. The author of (18) also used in the planning of urban layouts, but instead used an evolutionary deep learning model. The model was used to plan smart cities with a focus on reducing energy consumption and pollution. This was also used to plan changes to infrastructure of already existing smart cities. In the paper (19), the authors explore the use of a GA in an environment with Connected and Autonomous Vehicles (CAVs) to optimize the layout of an intersection. This proved to be better than traditional solutions however was only tested for an isolated intersection and not a map.

### 3.2 Reducing EV Travel Times

Many of the following related works are made possible by using technologies that are part of an Intelligent Transportation Systems (ITS). The term ITS refers to a variety of methods used to enable connectivity and communication in the transportation sector. This includes technologies such as vehicle-to-vehicle (V2V), vehicle to infrastructure (V2I), infrastructure-to-vehicle (I2V), and infrastructure-to-infrastructure (I2I) communications. Goals of these commu-

nications include increasing efficiency, increasing safety, reducing transportation delays, and being more environmentally friendly (20)(21). Also belonging to ITS is the so-called Advanced Driving Assistance System (ADAS), which is "a group of vehicle technologies that warn the drivers timely regarding the risky or hazardous situations to avoid crashes" (22).

### **3.2.1 Dynamic Path Finding**

The following related work employed various strategies in the area of dynamic path planning with the goal of reducing EV travel times. These studies all showed an average improvement in travel times for EVs and often for non-emergency vehicles (NEVs) as well.

The authors of (23) applied 4 different strategies including Dijkstra, Density-Based Dijkstra, Evolution Strategy, and Density-Based Evolution Strategy. The Dijkstra strategies used number of lanes to determine priority while the Evolution Strategies used an evolutionary algorithm to apply priorities. Both the Density-Based strategies also included traffic density when calculating priorities. The density statistics were transmitted via V2I technology. The experiments showed the most significant improvements using the Density-Based Evolution Strategy which reduced the arrival times for EVs by at least 47.9%. These experiments were conducted on 3 different maps of different urban areas, however the selection of maps were all large urban areas and did not include smaller urban areas or country roads. Additionally, the number of vehicles does not appear to stem from historical traffic data and may not reflect real-world traffic volumes.

The work of (24) also relied on V2I communications to improve travel times, but it additionally used V2V communications. The authors created an extension to the transportation simulation program MATSim which added support for V2I and V2V communication. This allowed any vehicles equipped with an ADAS to utilize V2V and V2I communication and change their routes dynamically in the event of an emergency. The experiments showed the most improvement of emergency response times when over 50% of vehicles had an ADAS, with an improvement of 36.7% when all vehicles supported it.

The research conducted in (3) was also able to improve travel times using dynamic routing, but the approach also combined this with other solutions such as traffic light changes, lane clearance, and changing speed limits. These

were all possible actions belonging to an adaptive Traffic Management System using fuzzy logic. Information such as the Congestion Level and Emergency Level (severity of the emergency) were combined to determine an Emergency Response plan which dictated which of the aforementioned actions to take. This resulted in significant reduction in arrival time for EVs, the highest improvement being 76%, while only causing minor disruption to the travel times for NEVs.

The paper (25) also used dynamic traffic light controls and vehicles which are able to communicate with a traffic control mechanism, but this time the traffic light phases were controlled using an algorithm called Priority Level Mutualism for Emergency Vehicle, which works by assigning a different level to each vehicle on a road and calculating which lanes of traffic should be granted a green phase based on the sum of the vehicles ranks. Emergency vehicles have a higher rank and can even be individually ranked based on the severity of the emergency. This solves any conflict between multiple EVs approaching the same intersection by allowing EVs on route to more urgent accidents to take priority. While the results were promising with an average reduction to the delay of an EV being as much as 49%, the tests were only performed on a single intersection with one lane of traffic. Experiments with larger networks and more complex intersections were not performed.

### 3.2.2 Dynamic Traffic Signal Control

A variety of research has also been done to decrease EV travel times using dynamic traffic signal controls. The work conducted in (26) concentrated on creating a green wave to enable EVs to drive through intersections as quickly as possible. This was done by using a combination of intrusive and non-intrusive pre-preemption. Intrusive preemption involved changing the light to green when an EV approached, while non-intrusive preemption involved extending the green phase so that the light was green when the EV entered the intersection. The non-intrusive method was used whenever possible and the intrusive method was only used when the non-intrusive method was non possible. After the EV passed through, the recover phase begins. This was able to quickly and effectively restore the flow of traffic and the authors claim that the methods applied had essentially no effect on the road network thanks to the restoration.

(27) also proposes a solution that enables the creation of a green wave for EVs. Specifically, they implement a solution that is able to restore coordination of the adjusted intersection signals with other signals. This is able to reduce average travel time for EVs by 26.9% and by 19.6% for NEVs. However, this approach only works in scenarios with centralized control of traffic signals and it is still possible for emergency vehicles to have red lights if they are unable to keep up with the intended green wave. (28) presents a solution adapted to manage priorities in situations where trams and EVs are present in an intersection, since the presence of a tram can block the path of an EV. However this experiment was only performed on a single intersection. The work presented in (29) provides a solution to the Emergency Deadlock problem, which is when green light priorities must be given in an intersection where EVs are approaching from all directions.

### **3.2.3 Research Gap**

In contrast to related studies that use EAs for city planning, such as the papers (18) and (17), where the focus lies on city planning as a whole, this thesis focuses specifically on planning intersections and does so in the context of reducing EV travel times. While (19) uses an EA for the creation of optimized intersection configurations, this was tested for NEV traffic and was performed on single intersections, not on a map.

While many dynamic traffic control and dynamic path finding methods found also had the goal of improving EV travel times, this thesis does not take a dynamic approach. We focus instead on creating fixed configurations to improve the travel times of EAs, which can be applied to areas that do not use ITS technologies, as it does not require V2I or V2V communication to function.

---

## 4 Methodology

This chapter goes into detail about the proposed solution including the parameters chosen, the software and data sources used, the scenarios tested, and how the code functions. The optimization and coordination of traffic light timings and their coordination is beyond the extent of this paper. The intersection configurations selected by the GA are from a set of configurations that were determined before the simulation run. The traffic light timings are static and therefore there may be more optimal timings which were not part of the selection set.

### 4.1 Overview of Genetic Algorithm Approach

While the previous chapter showed existing research on how to improve travel times for EVs dynamically, less research has been conducted using static or offline-optimized methods of city planning. Although dynamic methods can significantly reduce travel times, it is also important that roads and intersections are configured optimally to assist in this. For that reason, the idea presented in this thesis is to use a genetic algorithm which treats each intersection as a gene within an individual, where the alleles of these genes can be modified so that the intersections are set in an optimal configuration.

### 4.2 Software Used

The traffic experiments were conducted using the traffic simulation software *Simulation of Urban MObility (SUMO)* (30). *SUMO* was chosen because it is open source, free to use, and well documented. *SUMO* has been used in a variety of academic contexts, including these works aiming to reduce EV travel times: (23) (3) (31) (26). In order to run, *SUMO* requires at least two input files: a network file, which contains details about the edges, nodes and connections of a map, and a route file, which contains information about each trip, including the vehicle type, departure time, and the path it will take. Additional data such as traffic light timings may also be provided, but in this case we opted to use the traffic light configurations that are automatically generated

by *SUMO* when no configurations are explicitly provided. *SUMO* also allows the creation of vehicles with different classes, where the class determines the vehicle behavior. Among these classes is the class "emergency", which allows vehicles to perform actions such as ignoring red lights, overtaking on the right, and driving in a lane in the wrong direction (32).

One of the downsides to using *SUMO* was the simulation time. *SUMO* is not yet optimized to run on more than a single core (33). While the runtime on a single core is acceptable for conducting a few experiments, an EA requires many runs of the simulation and the sub-optimal runtimes greatly reduced the number of iterations that could be completed. A complete run of an entire simulation scenario using an Intel i7-8550U processor with 16 GB RAM took approximately seven hours. Although the simulation itself does not run on multiple cores, *SUMO* allows calculation of routes using multiple cores, and this option was used (33).

The evolutionary algorithm was run using *pymoo*, which is a python framework for multi-objective optimization (15). The experiments were conducted using *pymoo*'s genetic algorithm using the built in functions for survival and selection, which are fitness survival and tournament selection, respectively (34) (35). The fitness survival function ranks individuals by fitness and returns those with the best ranking (34). According to the source code, the default tournament type is a binary tournament selection (35). The sampling, fitness evaluations, crossovers, and mutations were done using custom functions which will be explained in greater detail later.

## 4.3 Data Sources and Preparation

A total of three maps were chosen to carry out the experiments to compare how well the algorithm performs with different traffic amounts. All maps are of areas in the United States, since city layouts there tend to be uniform and grid-like, which makes the editing of maps easier to perform. The maps include part of downtown Chicago (Figure 4.1), which has a high amount of traffic, downtown South Bend, Indiana (DTSB) (Figure 4.2) which is medium-sized city with less traffic, and an area in the southern part of St. Joseph County, Indiana (SJCS) (Figure 4.3) with little traffic and long, country roads. Though the maps vary in size, each contains 27-34 intersections that were adapted during the experiments.

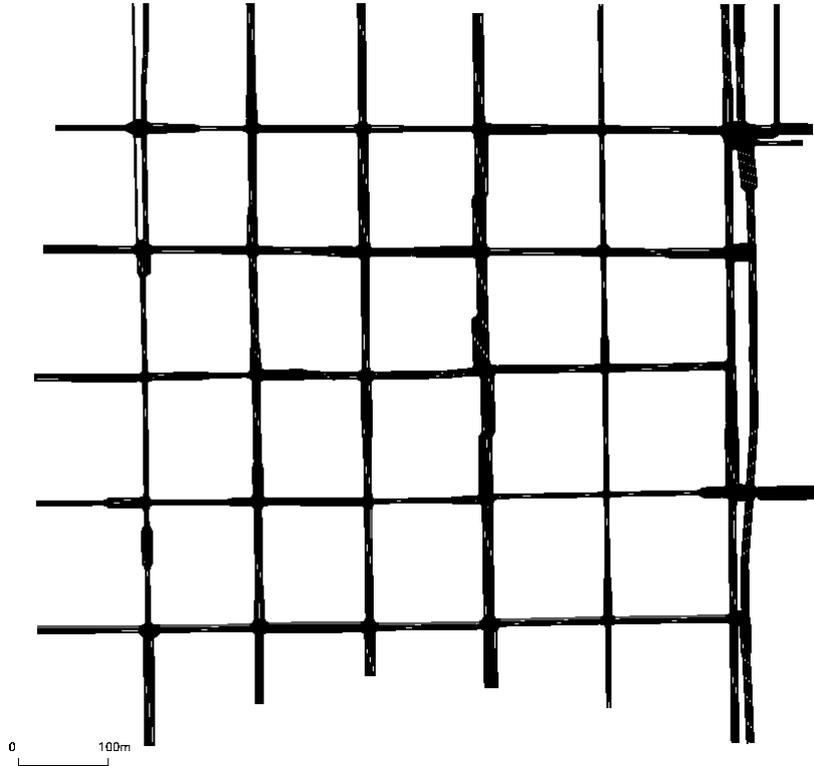


Figure 4.1: Simulation map of downtown Chicago. Map data © OpenStreetMap contributors, available under the Open Database License: <https://www.openstreetmap.org/copyright>.

The data for the maps was gathered from *Open Street Map (OSM)* (36). The exported data from *OSM* contained many details including among others sidewalks, bicycle lanes, and residential roads, and alleyways. Since the experiments are focused on EV traffic on main roads, these extra road types were removed. Bicycle and pedestrian traffic were not simulated in the interest of simplicity in setting up the simulation as well as to reduce computational complexity and simulation time. Additionally, for the map SJCS, roads within the small town of Lakeville were removed to keep the number of optimized intersections consistent with the other maps and to focus on the more major roads in the area.

The simulation uses real-world traffic data to be as realistic as possible. The traffic data for Chicago was sourced from (37) and the data for DTSB and SJSC came from (38). The data from both sources was given as a vehicle count over

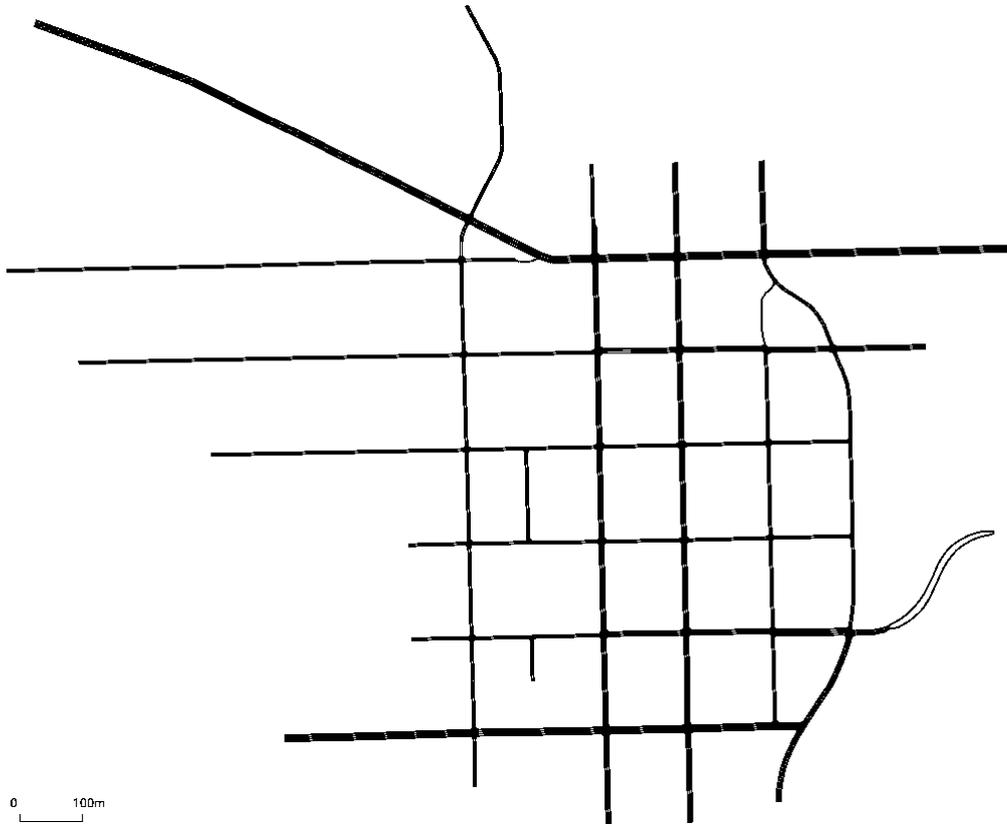


Figure 4.2: Simulation map of downtown South Bend (DTSB). Map data © OpenStreetMap contributors, available under the Open Database License: <https://www.openstreetmap.org/copyright>.

the span of 24 hours for each road. The data for Chicago was complete for all roads but somewhat generalized, as some vehicle counts were the same for long stretches of road extending several blocks and were not given specifically for each road segment. The data for DTSB and SJCS was incomplete, as some road segments had no statistics. Traffic data from neighboring segments of the same road was used in place of the missing data. The 24 hour data was later split into hourly data based on traffic volume patterns from (39), to ensure a more realistic distribution of traffic for each hour.

Due to the random nature of the starting populations, many initial map configurations were so sub-optimal that the rush-hour traffic on the Chicago and DTSB maps would bring the simulation to a standstill. Thus, the traffic had to be reduced by 20% to allow the simulation to run. The traffic volumes for SJCS remained true to the data gathered and the distributions found.

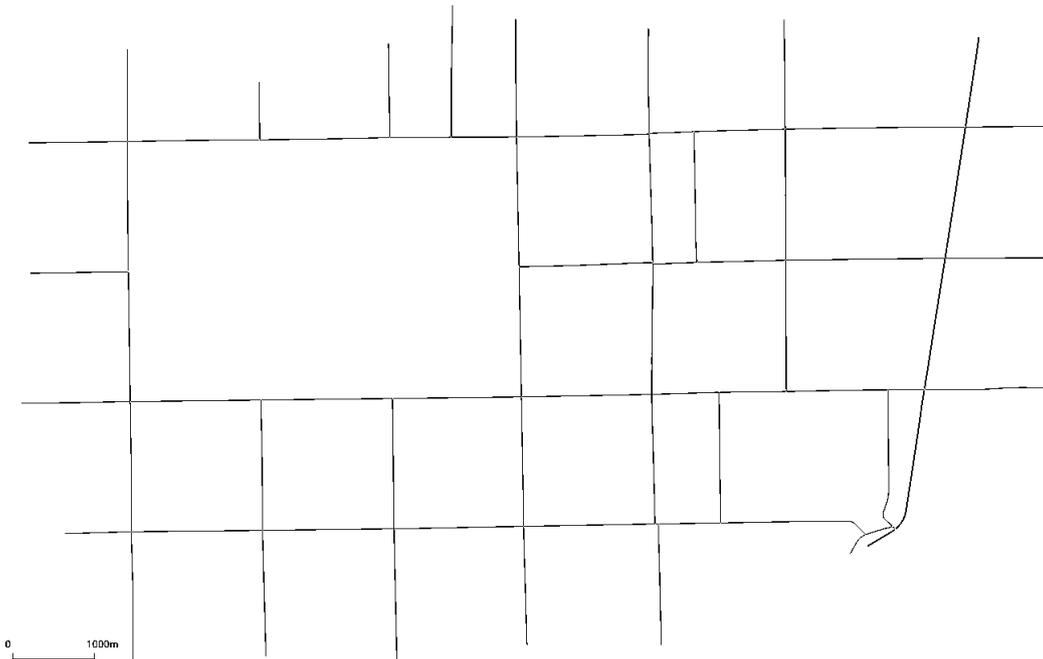


Figure 4.3: Simulation map of south St. Joseph County (SJCS). Map data © OpenStreetMap contributors, available under the Open Database License: <https://www.openstreetmap.org/copyright>.

By default, the vehicles generated were standard passenger cars. From these vehicles, 3% were randomly selected and converted to vehicle class "emergency", giving them special behavior similar to real-life EVs. While this exceeds a realistic amount of emergency vehicle traffic, it was necessary to ensure that the intersections were covered thoroughly and enough travel time data would be gathered to search for an optimal solution.

Each intersection can use one of two traffic light configurations using the timings automatically generated by *SUMO*. They are of type "traffic light" and "traffic light right on red". The "traffic light right on red" option is common in the United States and allows vehicles to take a right turn at a red light after coming to complete stop, provided that there is no oncoming traffic. The normal "traffic light" configuration requires vehicles to wait for a green light before turning right. The traffic light timings include the phases for each traffic lane and the duration of each phase. It was assumed that the automatically generated traffic light timings would be valid and suitable to the traffic demand. It is worth noting that the traffic light timings were fixed and did not

adjust to traffic volume. *SUMO* allows the use of actuated traffic lights, which are lights that adjust their green phase based on traffic demand (40), however this was not used since running the simulations with this option caused the EA to crash.

## 4.4 File Editing During Runtime

The code described in this section was developed with the help of AI, as disclosed in section 1.4. Figure 4.4 shows the steps taken to edit the XML files needed to run the simulation. These actions are performed on each individual and transform its encoding into a corresponding map for the simulation to run on.

**Edit Intersections** The function takes an individual and an edge, node, and connections XML file as input. These XML files already contain the information required to generate the network, but we modify them to contain the intersection types encoded in the individual. For each intersection in the individual, the appropriate function to edit the files will be called based on the intersection type to change to. In a simple case, for example an intersection that needs to be modified to have a traffic light, this requires changing the parameter "type" for that intersection in the node file to "traffic\_light". Adding a roundabout is more complex and requires creating new nodes and edges within the roundabout and connecting all the edges to each other.

**Create Net File** The edited node, edge, and connections files resulting from the previous step are used by the *SUMO* function *netconvert*, which combines them all into a single network file. This is one of the two files used to run the actual simulation.

**Create Routes File** The routes file contains the actual paths taken by each vehicle. It is created from the network file, which now contains the modified intersections, and the trips file, which contains start and end locations for a vehicles journey but does not specify the path between. The trips file was created once for each map prior to the begin of the simulation using *randomTrips.py*, which is included in *SUMO*. The following explains how *randomTrips.py* was used. To avoid short journeys, the parameter **-min-distance** <FLOAT> was used. This specifies the

minimum distance between the start and end location. A large enough value was chosen so that a vehicle would need to travel through four to five intersections. The parameter `–fringe-factor` with the value 10 was also used, which means that "edges that have no successor or no predecessor will be 10 times more likely to be chosen as start- or endpoint of a trip" (41). Since the maps of the downtown areas do not include the entire downtown, this was done to simulate the flow of vehicles that start outside map and travel to downtown, or just pass through downtown. Additionally the parameter `–validate` was used to ensure all trips have a valid route connecting them.

The routes file is created by *duarouter*, which is built into *SUMO*. *duarouter* calculates a path between the start and end location using the shortest path (42).

**Adjust Traffic Volume** The routes file from the previous function is a list of possible journeys that can be taken, but it does not yet reflect the frequency of use for each edge. Therefore we run *routeSampler.py*, another tool built into *SUMO*, to adjust the routes file based on the traffic amount of each edge, and to add more vehicles to each route as needed. The function takes the existing routes from the routes file and the hourly traffic amount for each edge from the traffic data file.

**Add EV Traffic** The routes in the adjusted route file contain the vehicle class, route, and departure time. By default, a standard passenger vehicle class will be used. This function changes a percentage of the vehicle traffic, in this case 3%, to class "emergency". The vehicles to be converted are chosen randomly.

**Run Simulation** Now that the map has been edited and the appropriate traffic is generated, the simulation is run. The simulation is run without graphical output to save on resources, and each simulation finishes once there are no more vehicles to simulate.

**Process Output Logs** Several output logs are generated once the simulation finishes. These outlook logs are then parsed to gather the travel time for each vehicle of class "emergency" and the average travel time in calculated. This value is then used as the fitness for the individual.

## 4.5 Evolutionary Algorithm Design

The mutation and crossover functions described in this section were created with the help of AI, as disclosed in section 1.4. This section goes over details of the setup of the evolutionary algorithm including the encoding of individuals, mutation and crossover functions, and the fitness function. Due to the long simulation time, the population size and number of iterations had to be reduced. However, since the goal of this thesis is to show the potential for improvement using an evolutionary algorithm and not necessarily to find the optimal solution, this was deemed acceptable.

### 4.5.1 Sampling and Individual Encoding

Sampling is performed at the start of each run of the algorithm to build the individuals of the starting population. An individual is encoded as a list of numbers, where each index in the list corresponds to a different intersection and each number is an allele representing the intersection type. The allele for an intersection represents the amount of roads it connects: four roads, three roads, two roads (a connecting node for when the amount of lanes change), and dead-end (entry/exit nodes on the map). It also represents the intersection type: roundabout, traffic light, traffic light with right turn on red, all-way stop, and priority stop (two-way stop). These various intersection types were chosen to give a variety of possible encodings for each gene. It is worth noting that the roundabout intersection type results in changes to the physical layout of the map, while the other configurations only modify how traffic is controlled. The roundabouts are comprised of three or four nodes within the roundabout and are connected to the pre-existing roads. While the edges within the roundabout are not curved, the roundabouts will still function properly so long as the correct priorities are assigned to the edges (43). Real-world physical constraints were not taken into account when placing roundabouts. Figure 4.5 shows a mapping of the encodings and a simple map with an example configuration. It is worth noting that the entries in the nodes files are not ordered according to their location on the map, i.e. two consecutive entries in the nodes file are not necessarily neighboring nodes on the map.

### 4.5.2 Parameter Selection

There was some difficulty in determining what crossover probability would be appropriate for the experiments. Due to the long duration of each simulation run, the population size and number of iterations had to be reduced so that the simulations could be performed in time to gather and analyze results. The parameters chosen were based on research on EAs conducted in (44). It is worth noting that these experiments were conducted with significantly more iterations. Nevertheless, it seemed reasonable to use these values as a point of reference and the values found were further adapted to fit the limited number of iterations.

A static crossover rate of 0.9 was used for both crossover methods. The mutation rate was set dynamically based on the number of intersections, which ranges from 27-34. The mutation rate starts at  $5.0/l$  (where  $l$  is the length of an individual) and decreases gradually with each generation to end at  $1.0/l$ . For example, if there are 30 intersections, the starting mutation probability is  $5.0/30 = 0.167\%$  and the final mutation rate is  $1.0/30 = 0.033\%$ . The large starting rate is due to the low number of iterations. We introduce more diversity in the starting generations and then refine this selection of individuals over time. This prioritizes exploration in the early generations and exploitation in the late generations. A population size of 20 was used and each experiment was conducted over 35 generations, for a total of 700 evaluations per experiment.

### 4.5.3 Crossover Functions

A one-point crossover and a uniform crossover were tested. These two crossover types were chosen to see if there is any difference in conversion, since the former preserves larger sections of the map while the latter creates more diverse offspring. To determine if a crossover will take place, a seed is given at the start of the simulation which is used to generate a number between 0.0 and 1.0. If the generated number is smaller than 0.9, a crossover will occur.

#### One-Point Crossover

The one-point crossover works by using the seed to generate a random index within an individual as the crossover point. It then creates two new individuals

that each contain the first half of one parent and the second half of the other parent. This is demonstrated in figure 4.6.

### Uniform Crossover

For the uniform crossover, we use the seed to create a mask that determines which genes in the children come from each parent. We generate a number between 0.0 and 1.0 for each position in the individual. For the positions where the number is smaller than 0.5, the mask is filled with a one; otherwise it is filled with a zero. The first child contains genes from the first parent wherever the mask contains a one, and contains genes from the second parent wherever the mask contains a zero. The second child is the opposite; it contains a gene from the second parent wherever the mask contains a one and a gene from the first parent wherever the mask contains a zero. The offspring are guaranteed to have a valid encoding after the crossover, since the genes taken from parents still have the same index corresponding to a certain intersection. Figure 4.6 shows an example of a uniform crossover.

#### 4.5.4 Mutation

The mutations performed are standard mutations and are performed by selecting a single gene of an individual, which in this case encodes a single intersection, and changing the allele. Once the gene is selected, the allele for that gene is looked up in a dictionary containing a list of compatible mutations and a replacement allele is selected randomly from the list. The selected allele will then be applied as the new intersection type. Other mutation types such as the shift mutation would not be applicable in this case, since not all allele are interchangeable and a shift would result in intersections being encoded with values that do not match the intersection size.

#### 4.5.5 Fitness, Survival, and Selection

The fitness of an individual is the mean travel time of all EVs during a single simulation run. The travel times for NEVs are not taken into account when calculating fitness. Since the goal is to minimize EV travel times, candidates with a lower fitness are deemed as better. The fitness of all map layouts in a generation are then compared to determine the surviving individuals for the

following generation. Those with the lowest fitness survive (34). The selection method used is binary tournament selection, which is pymoo's default for a genetic algorithm (35).

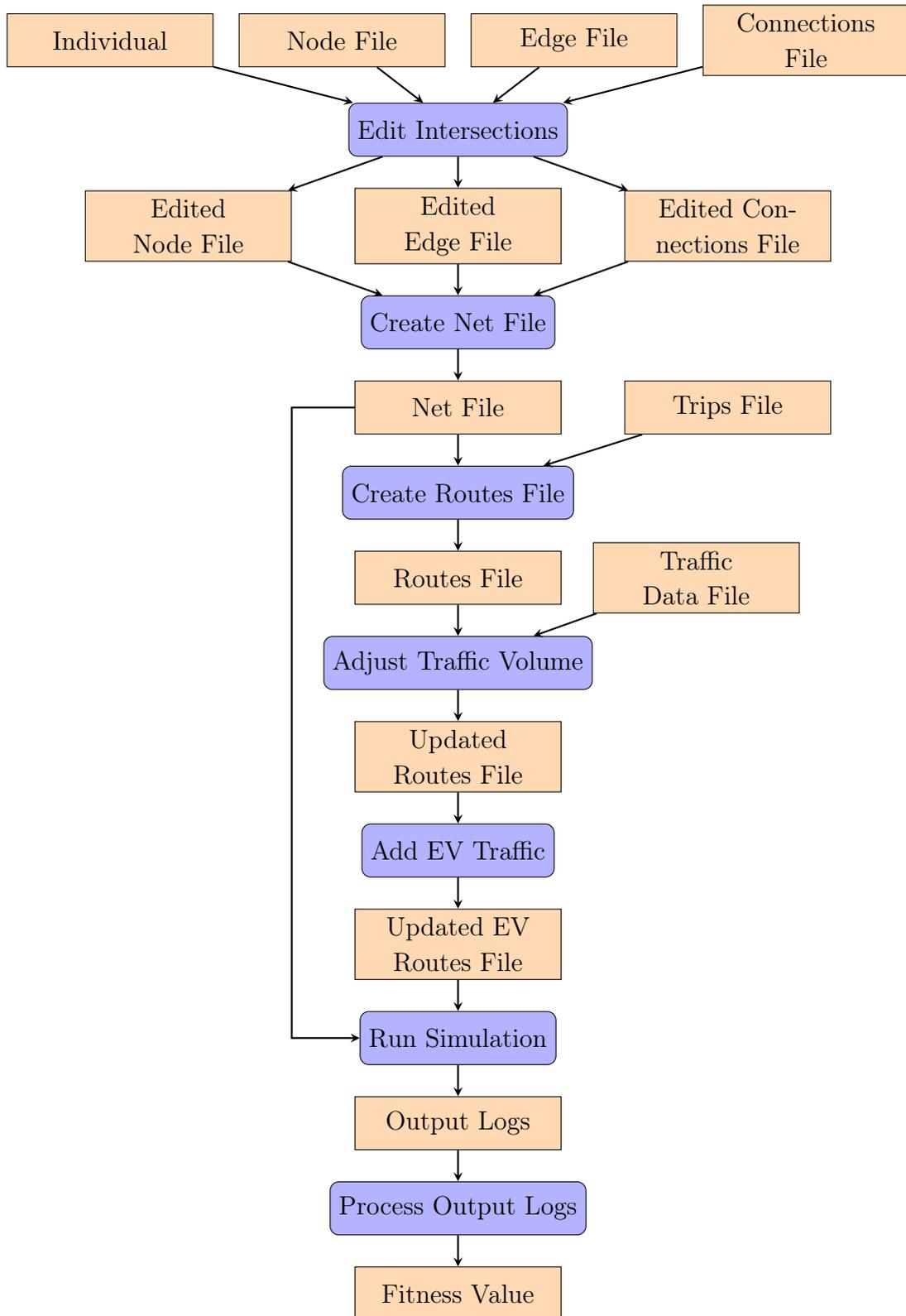
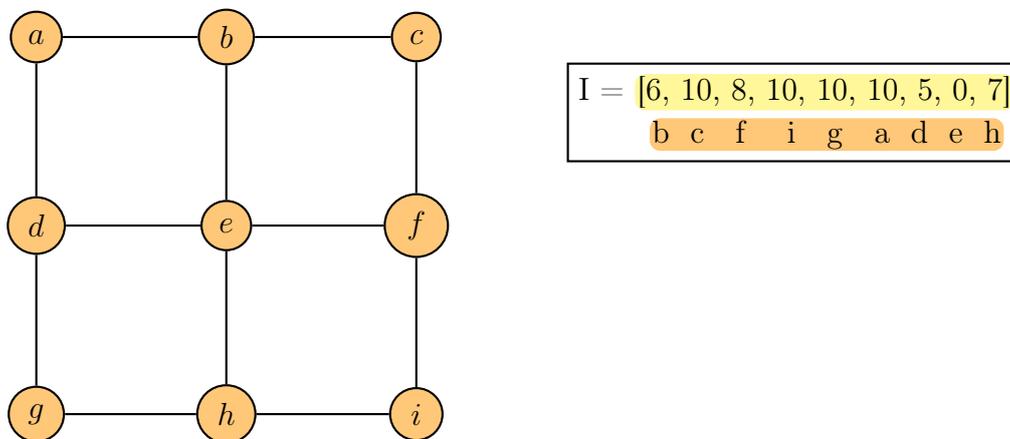


Figure 4.4: Procedure of XML file handling.

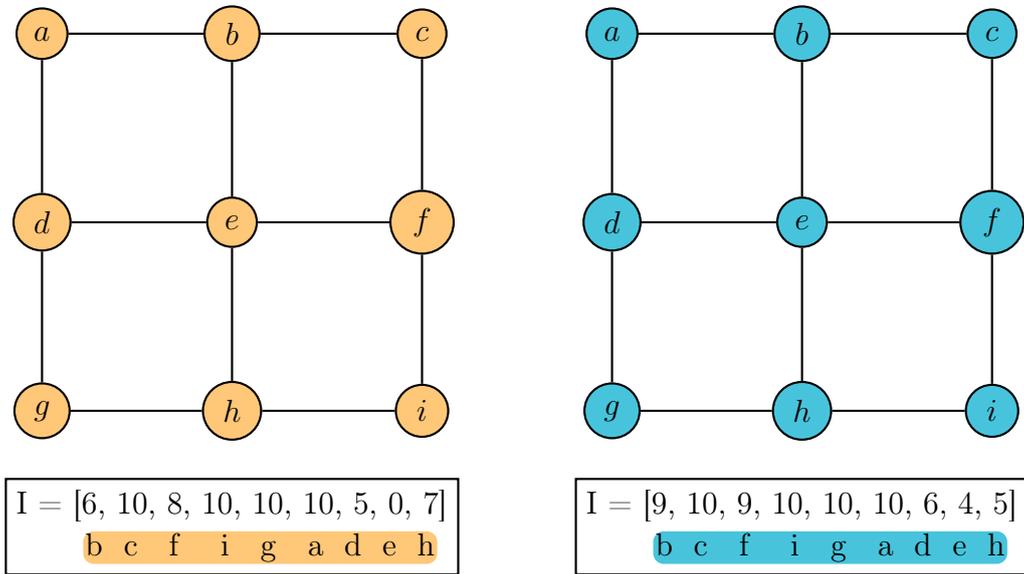
| Intersection Encoding | Intersection Type                    |
|-----------------------|--------------------------------------|
| 0                     | four-way roundabout                  |
| 1                     | four-way traffic light               |
| 2                     | four-way traffic light right-on-red  |
| 3                     | four-way all-way stop                |
| 4                     | four-way priority stop               |
| 5                     | three-way roundabout                 |
| 6                     | three-way traffic light              |
| 7                     | three-way traffic light right-on-red |
| 8                     | three-way all-way stop               |
| 9                     | three-way priority stop              |
| 10                    | two lanes connected                  |
| 11                    | dead-end                             |

(a) Encoding legend

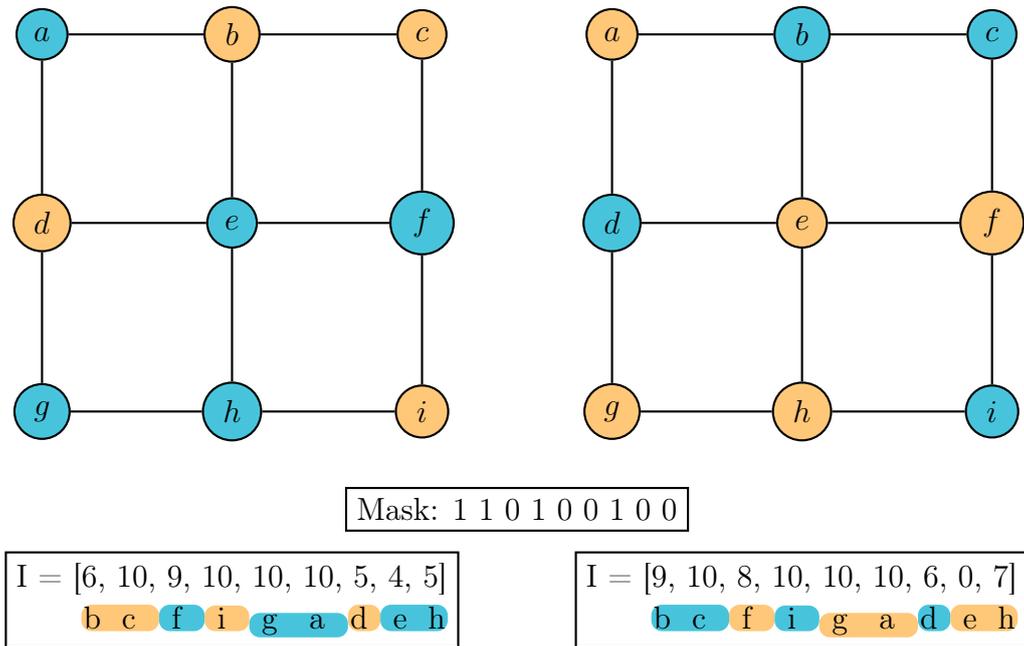


(b) Example map and encoding of configuration

Figure 4.5: Mapping of encodings and an example of a possible configuration. The tables were created and the graphics were edited with help of AI, as disclosed in 1.4.

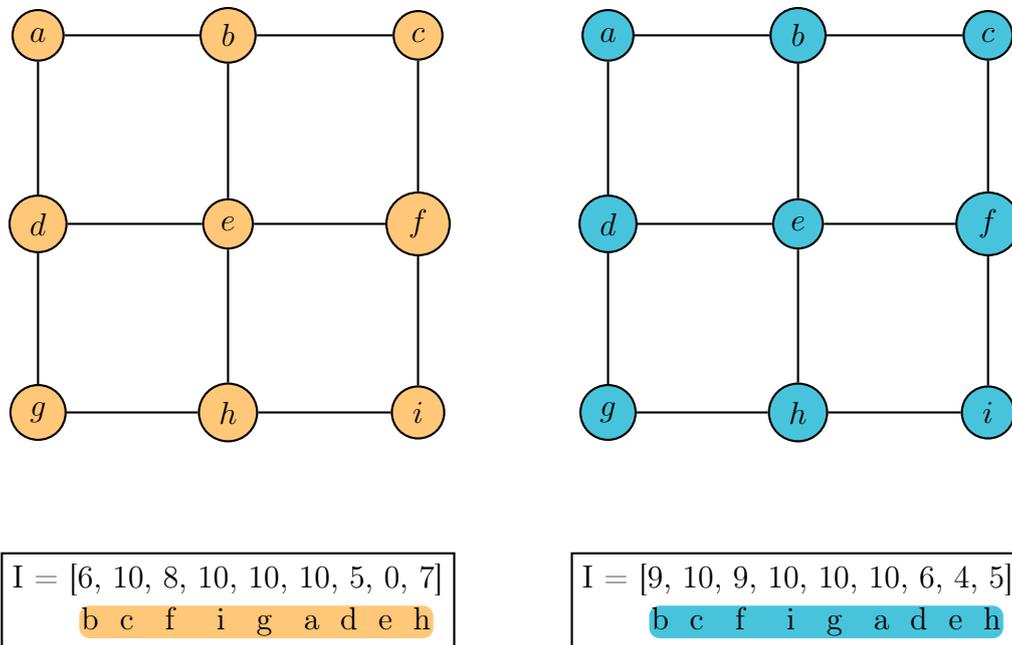


(a) Parents in a uniform crossover

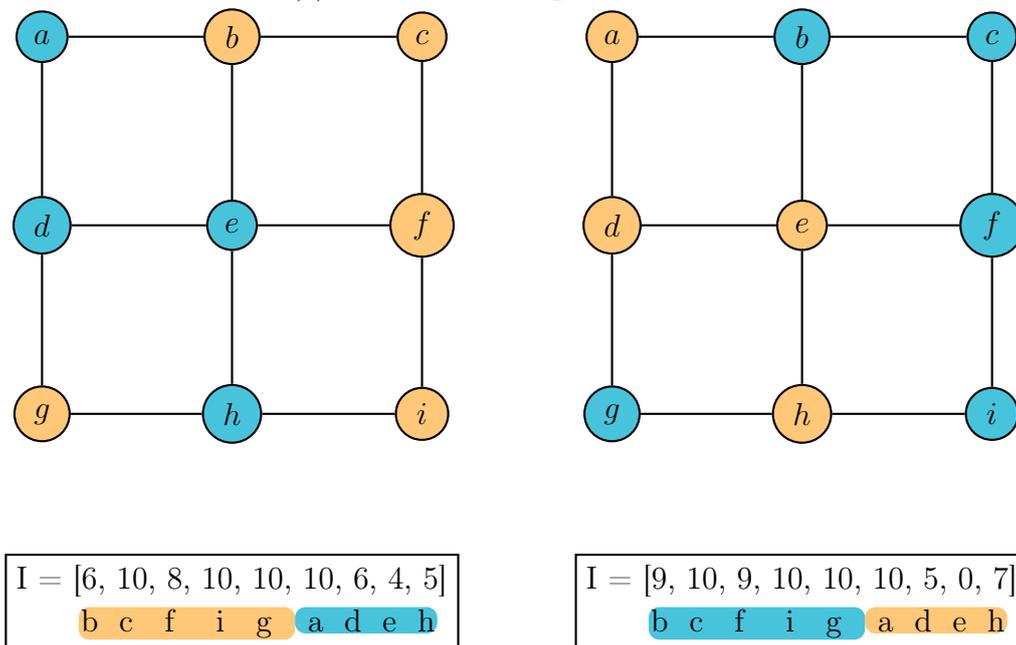


(b) Offspring of uniform crossover

Figure 4.6: Example of a uniform crossover.



(a) Parents in a one-point crossover



(b) Offspring of one-point crossover

Figure 4.7: Example of a one-point a crossover.

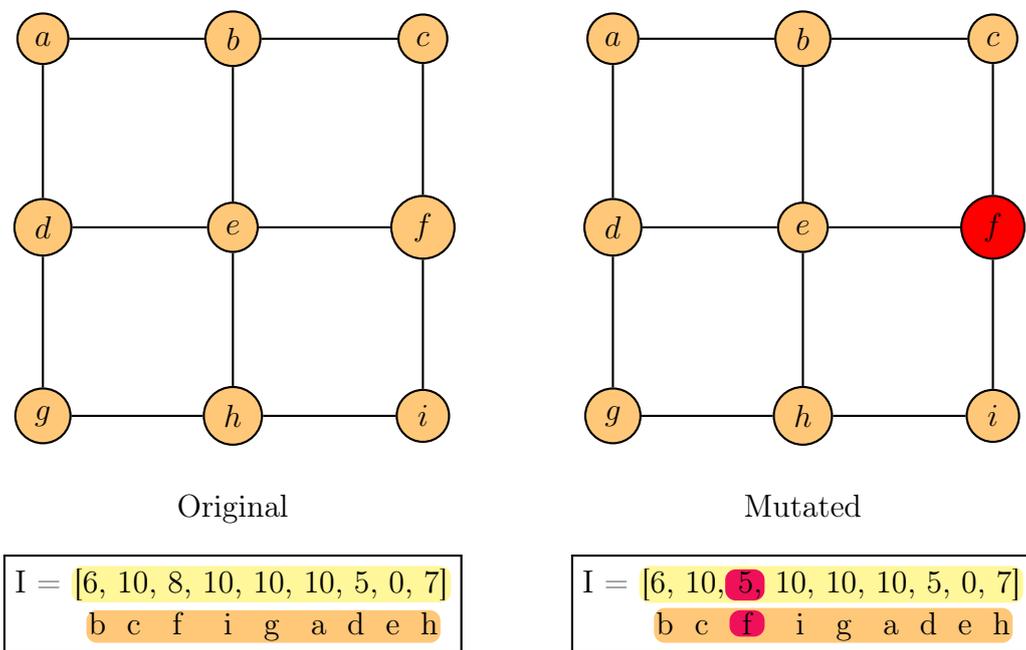


Figure 4.8: Example of a mutation.

---

# 5 Experiments and Results

This chapter will first go into detail about the simulation scenarios and the setup of each experiment. Afterwards, the results of the experiments will be evaluated.

## 5.1 Experiments

The experiments were performed on three maps containing 27-34 intersections, each with a different layout and amount of traffic. A summary of the different experiments can be found in table 5.1. Simulations were ran for enough simulation hours to generate the needed EV traffic for the evaluations. For the Chicago map, the simulations were ran for one hour of rush-hour traffic, from 17:00 to 18:00. For the DTSB map, the simulations were ran for four hours of evening rush hour traffic from 14:00 to 18:00. The SJCS simulations were run for a full 24 hours, since there is significantly less traffic per hour on this map. The experiments were run on each map using two crossover types: uniform and one-point. These experiments were performed using three different seeds due to computational limitations, for a total of six experiments per map. The seeds were used to generate the starting population as well as to generate the random numbers needed for the probabilities for the crossover and mutation functions. Additionally, a baseline experiment was run on each map using a configuration that mirrors the real-world intersection configuration as accurately as possible. Image data from Google Maps was used to verify which intersection types were in place as of the time of writing.

## 5.2 Results

The analysis and visualization code used in this section were developed with the help of AI, as disclosed in section 1.4. In the following section we will address the research questions posed in section 1.2. We examine the results of the simulation and the impact on the travel times for EVs and NEVs. We compare the convergence of the two crossover methods, as well as perform robustness tests to see how well the best solutions perform with different traffic volumes.

### 5.2.1 Comparison of EA Solution Fitness to Real-world Fitness

First, we compare the fitness of the generated solutions to the fitness of the real-world baseline solutions to see how much of an improvement was achieved by the algorithm. Table 5.2 shows the fitness of the real-world experiments compared to the fitness of the final iteration for each scenario and the improvement achieved. Details about each scenario are given in section 5.1 and in table 5.1. Both crossover methods showed improvements over the real-world scenarios for all maps.

The most significant improvements were on the Chicago map, where average EV travel times are improved by 127.51 seconds in the best case. An analysis of the intersection frequencies in the final individuals of the Chicago map was performed to gain insights as to why there was such a large improvement. Table 5.3 shows that the most common intersection types are all-way and priority stops and that the least common intersection types are traffic lights. This seems counterintuitive for a large city like Chicago, since in reality the intersections in this area are all traffic light controlled. However, the simulated traffic lights were neither configured to adjust dynamically to traffic volume nor were they synced to each other to create "green waves". The baseline configuration for Chicago is all traffic lights, so it could be that this configuration is simply inefficient without the lights being coordinated and set for dynamic adjustment. The traffic lights also have the issue that they create sudden influxes of traffic, which can cause the roundabouts to clog up if there is too much traffic at once. This could explain why roundabouts are also not a commonly occurring intersection type. In the case of these simulations, all-way stops and priority stops are able to create a slower but more consistent flow of traffic.

The least significant improvements were on the SJCS map, with an improvement of 10.39% in the best case and as little as 2.99% improvement in the worst case. This is likely because the majority of the roads in the baseline configuration are priority roads with little traffic, so an EV can proceed quickly through most intersections no matter how they are configured. The EV arrival times for this map are more dependent on the distance traveled than the intersection configurations, so there was less improvement to be gained in this scenario.

### 5.2.2 Evaluation of Different Crossovers

Since both crossover methods outperformed the baseline configuration, we next compare the average fitness between the crossover methods to see which had better performance over time. This is compared for each map and all seeds per map. Figures 5.1, 5.2, and 5.3 show that both crossover methods have a similar convergence pattern, with the largest improvements occurring during the first few iterations and minor improvements in the final iterations.

Table 5.2 shows the best-performing crossover for each seed in bold. Since fitness is calculated as the average travel times of all EVs in a simulation run, a lower fitness value is better. The uniform crossover produced the best individuals for all seeds of the Chicago map, while the results for the other maps are mixed, with one-point crossover producing better individuals for the majority of scenarios on the other two maps. Each crossover method was tested for nine scenarios, with the uniform crossover each outperforming the one-point crossover for in five of the nine scenarios. These results suggest that the uniform crossover could be slightly better but there is no obvious advantage of one method over the other, since the sample size is limited.

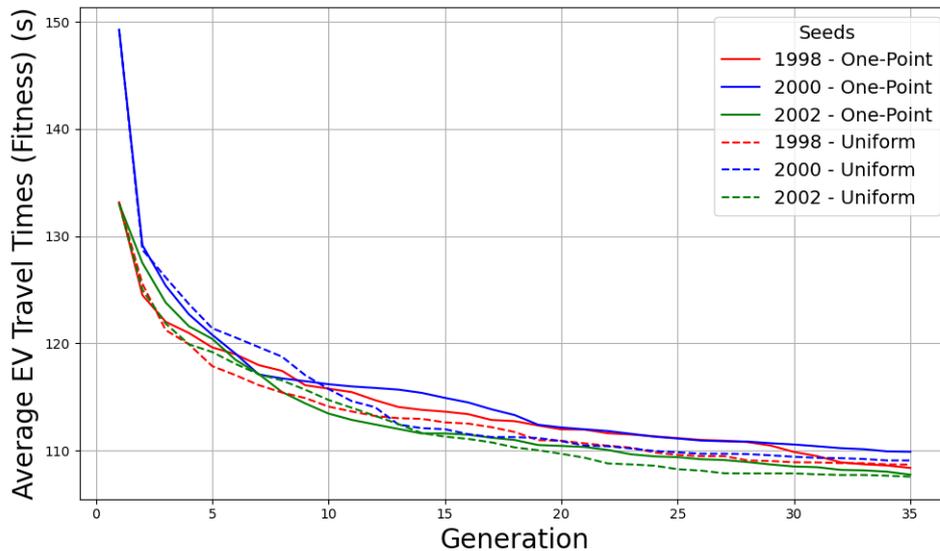


Figure 5.1: Crossover comparison - average fitness per generation on DTSB map

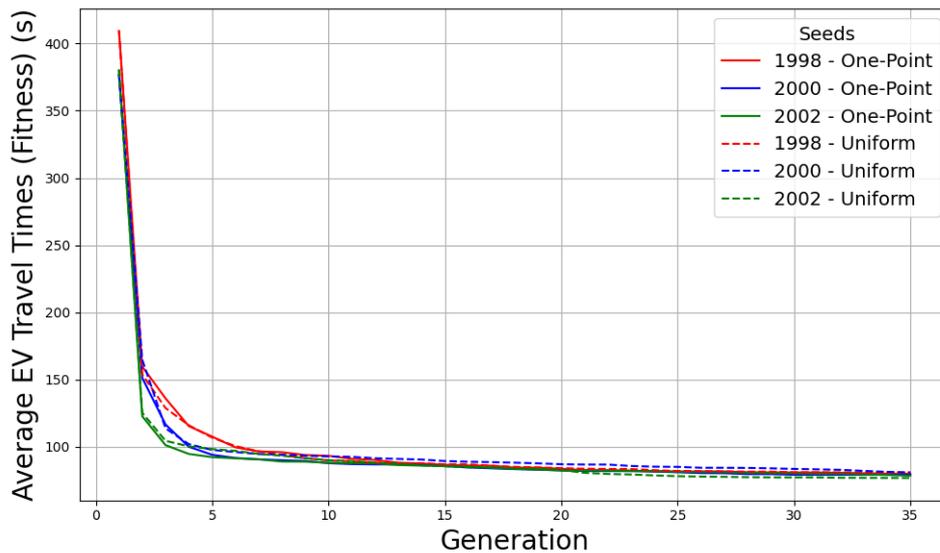


Figure 5.2: Crossover comparison - average fitness per generation on Chicago map

### 5.2.3 Influence of Traffic Volume

We evaluated the performance of the individuals post-EA with different traffic amounts to examine the robustness of the generated solutions with different traffic volumes. The individuals with the highest relative improvement for each map (shown in figure 5.2) were compared to the individuals of the real-world configurations. These individuals with the highest relative improvement will be referred to as champions. The following tables shows the results of simulations ran with 75%, 100%, 125% and 150% traffic volume, where 100% traffic volume is consistent with the reduced traffic volume used during simulations, not the original traffic volumes gathered. Each traffic volume experiment was performed with 10 seeds. Also shown are the ranges of highest and lowest EV travel times across all seeds, which are drawn within the shaded areas.

Figure 5.4 shows the performance comparison on the Chicago map for all traffic volumes. Here the average travel times increase greatly for both solutions with 125% and 150% traffic, with the champion solution performing especially poorly with 150% volume. A closer look at the EV travel times over the course of the simulations with 125% and 150% volumes shows acceptable times in the beginning, since the simulation starts out without any vehicles loaded. There is a large spike in travel duration towards the end as the simulation

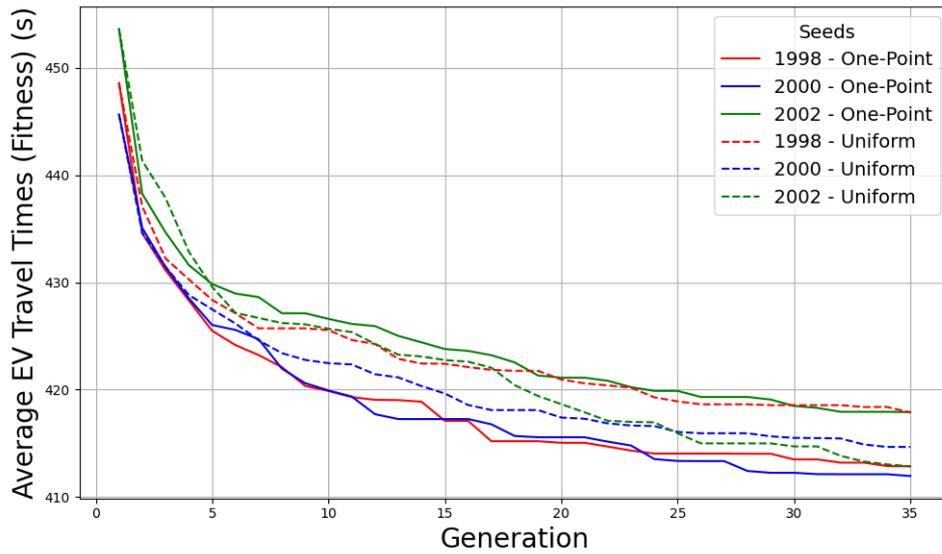


Figure 5.3: Crossover comparison - average fitness per generation on SJCS map

becomes overcrowded with vehicles. This also occurs for a few seeds with 100% traffic volume, but the amount of vehicles experiencing this is low enough that the travel times are misleading, since they appear only slightly higher when graphed as an average. The travel times remain consistent for the champion solution with 75% travel volume without an increase in duration at the end. Figure 5.5 provides a closer look at performance for 75% and 100% traffic volume and shows that the champion configuration outperforms the real-world configuration with 75% and 100% traffic volume.

The performance on the DTSB map is shown in figures 5.6 and 5.7. Here we see that both solutions performed poorly for 150% traffic volume. A closer examination shows that for most of the seeds tested with the champion solution a spike in travel times occurs towards the end of the simulation. The champion solution performs better than the real-world solution for the other traffic volumes and does not show the large spike in travel duration.

Unlike on the other two maps, the champion for the SJCS map performs worse for almost all traffic amounts except 75%. A closer look at the travel times for each seed shows mixed results. The champion solution performs better than the real-world solution for some seeds but produces slower travel times as a whole. This suggests that the slower times could be a result of the seeds selected but testing with more seeds would be needed to confirm this.

Surprisingly, the champion solution performs slightly better for 125% traffic volume than it does for 100% traffic volume. This could be due to random differences in the EVs routes and traffic patterns with this traffic volume, since the EVs are a randomly selected 3% of the total vehicles.

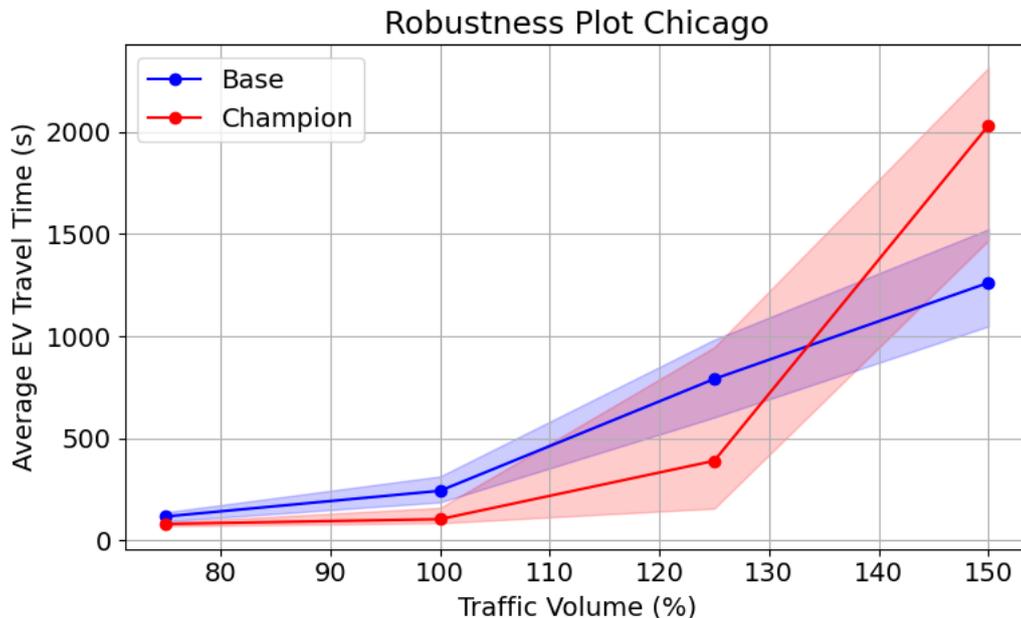


Figure 5.4: Performance comparison of champion Chicago solution under different traffic amounts.

#### 5.2.4 Impact on Non-Emergency Vehicles

Finally, we compare how the improvements to EV travel impacted the travel times for NEV traffic. Figures 5.9 through 5.14 show the average travel times for each generation, combined for all seeds, with individual plots drawn for uniform and one-point crossovers. The Chicago map shows a large improvement initially and a slow improvement over time with some variation between generations. Since the charts show an average of all solutions, not just the solutions with the best NEV times, the reason for such variation could be that some generations contain more sub-optimal solutions than others. There is huge variation in the travel times between generations on the DTSSB map. This could also be due to a few especially poor configurations, but more investigation is necessary to determine why there is more fluctuation than on the

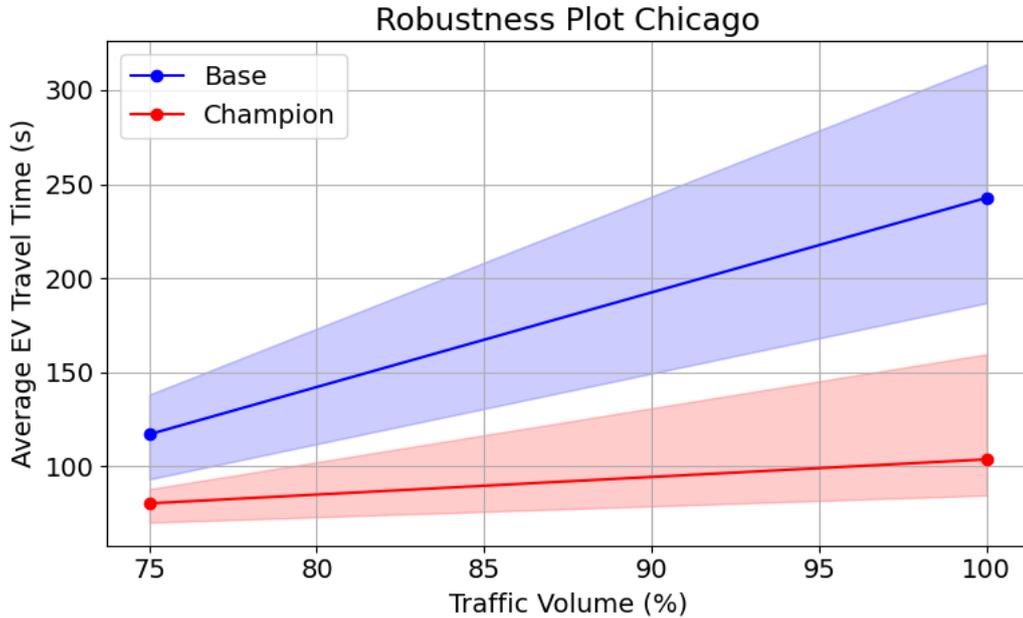


Figure 5.5: Close-up of champion performance for the Chicago map with 75% and 100% traffic volume.

Chicago map. There is a slight improvement to the NEV travel times between the first and final generation on the SJCS map but it mostly remains constant. Since the arrival times for this map are mostly dependent on the length of distance traveled and less on traffic or the intersection configurations themselves, this is to be expected. Interesting to see for this map is that the NEV travel times from the generated solutions are worse than the the real-world baseline solutions. We examined the types of intersections used in the real-world configuration and the generated configurations. The real-world configuration contains mainly priority-stops and a single traffic light, which the generated solutions often contain many traffic lights. In the real-world configuration, NEVs are able to proceed directly through priority stops if they have priority, or they can proceed as soon as the path is clear if they do not have priority. Since the generated solutions contain more traffic lights, NEVs are required to wait at a red light even if there is no conflicting traffic, which is likely the reason for the increased travel times on this map.

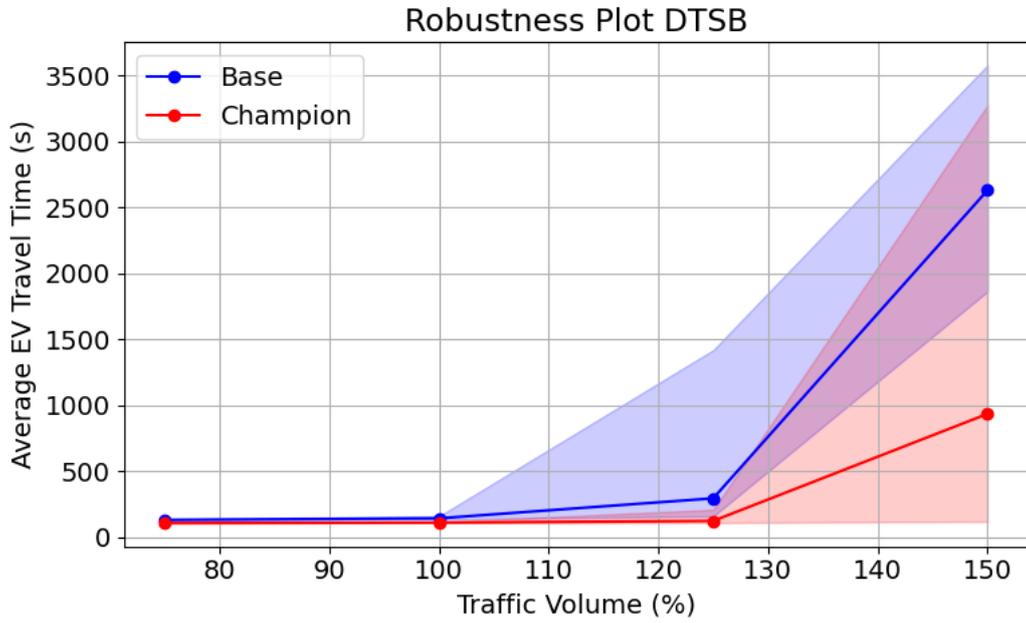


Figure 5.6: Performance comparison of best DTSB solution under different traffic amounts.

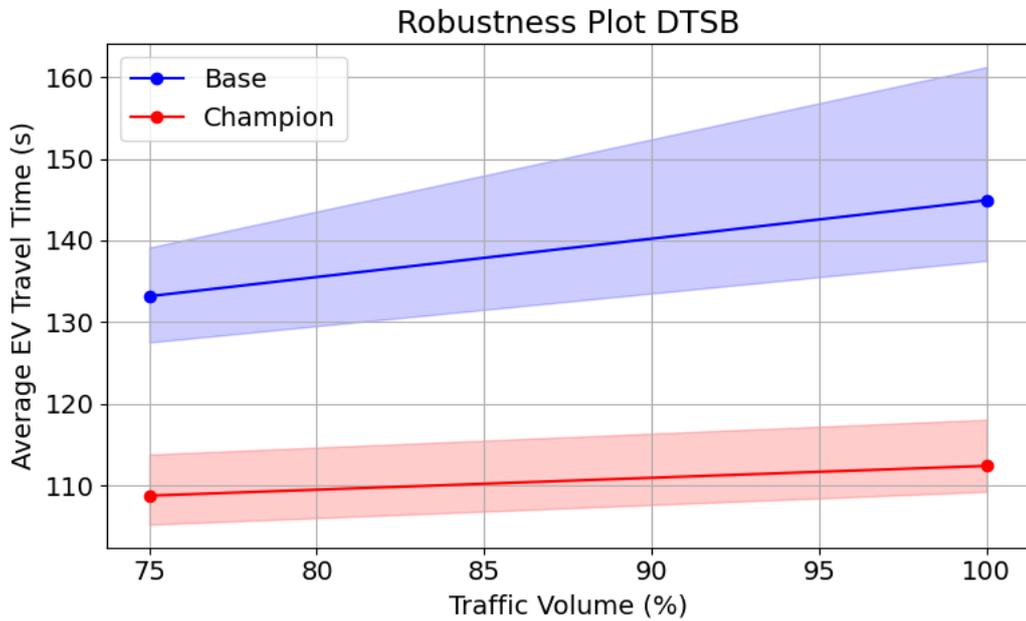


Figure 5.7: Close-up of champion performance for the DTSB map with 75% and 100% traffic volume.

| Map     | Traffic Amount | Number of Inter-sections | Simulated Time | Seed | Case       |
|---------|----------------|--------------------------|----------------|------|------------|
| Chicago | 4195 - 4775    | 27                       | 17:00 - 18:00  | 1998 | One-Point  |
| Chicago | 4169 - 4752    | 27                       | 17:00 - 18:00  | 1998 | Uniform    |
| Chicago | 4370           | 27                       | 17:00 - 18:00  | 1998 | Real-world |
| Chicago | 4189 - 4659    | 27                       | 17:00 - 18:00  | 2000 | One-Point  |
| Chicago | 4174 - 4738    | 27                       | 17:00 - 18:00  | 2000 | Uniform    |
| Chicago | 4370           | 27                       | 17:00 - 18:00  | 2000 | Real-world |
| Chicago | 4192 - 4709    | 27                       | 17:00 - 18:00  | 2002 | One-Point  |
| Chicago | 4241 - 4756    | 27                       | 17:00 - 18:00  | 2002 | Uniform    |
| Chicago | 4370           | 27                       | 17:00 - 18:00  | 2002 | Real-world |
| DTSB    | 11288 - 12761  | 33                       | 14:00 - 18:00  | 1998 | One-Point  |
| DTSB    | 11304 - 12895  | 33                       | 14:00 - 18:00  | 1998 | Uniform    |
| DTSB    | 11577          | 33                       | 14:00 - 18:00  | 1998 | Real-world |
| DTSB    | 11258 - 12448  | 33                       | 14:00 - 18:00  | 2000 | One-Point  |
| DTSB    | 11351 - 13172  | 33                       | 14:00 - 18:00  | 2000 | Uniform    |
| DTSB    | 11577          | 33                       | 14:00 - 18:00  | 2000 | Real-world |
| DTSB    | 11301 - 12736  | 33                       | 14:00 - 18:00  | 2002 | One-Point  |
| DTSB    | 11366 - 12816  | 33                       | 14:00 - 18:00  | 2002 | Uniform    |
| DTSB    | 11577          | 33                       | 14:00 - 18:00  | 2002 | Real-world |
| SJCS    | 6589 - 6842    | 34                       | 00:00 - 24:00  | 1998 | One-Point  |
| SJCS    | 6589 - 6873    | 34                       | 00:00 - 24:00  | 1998 | Uniform    |
| SJCS    | 6765           | 34                       | 00:00 - 24:00  | 1998 | Real-world |
| SJCS    | 6598 - 6828    | 34                       | 00:00 - 24:00  | 2000 | One-Point  |
| SJCS    | 6601 - 6860    | 34                       | 00:00 - 24:00  | 2000 | Uniform    |
| SJCS    | 6765           | 34                       | 00:00 - 24:00  | 2000 | Real-world |
| SJCS    | 6600 - 6839    | 34                       | 00:00 - 24:00  | 2002 | One-Point  |
| SJCS    | 6600 - 6877    | 34                       | 00:00 - 24:00  | 2002 | Uniform    |
| SJCS    | 6765           | 34                       | 00:00 - 24:00  | 2002 | Real-world |

Table 5.1: Simulation scenario details

| Scenario       |             |                  | Results       |                    |                    |
|----------------|-------------|------------------|---------------|--------------------|--------------------|
| Map            | Seed        | Case             | Final Fitness | Absolute Impr. (s) | Relative Impr. (%) |
| Chicago        | 1998        | real-world       | 180.85        | —                  | —                  |
| Chicago        | 1998        | one-point        | 77.56         | 103.29             | 57.11              |
| <b>Chicago</b> | <b>1998</b> | <b>uniform</b>   | <b>76.42</b>  | <b>104.43</b>      | <b>57.74</b>       |
| Chicago        | 2000        | real-world       | 202.76        | —                  | —                  |
| Chicago        | 2000        | one-point        | 76.47         | 126.29             | 62.29              |
| <b>Chicago</b> | <b>2000</b> | <b>uniform</b>   | <b>75.25</b>  | <b>127.51</b>      | <b>62.89</b>       |
| Chicago        | 2002        | real-world       | 193.16        | —                  | —                  |
| Chicago        | 2002        | one-point        | 74.7          | 118.46             | 61.33              |
| <b>Chicago</b> | <b>2002</b> | <b>uniform</b>   | <b>72.53</b>  | <b>120.63</b>      | <b>62.45</b>       |
| DTSB           | 1998        | real-world       | 159.10        | —                  | —                  |
| <b>DTSB</b>    | <b>1998</b> | <b>one-point</b> | <b>106.44</b> | <b>52.66</b>       | <b>33.10</b>       |
| DTSB           | 1998        | uniform          | 107.16        | 51.94              | 32.65              |
| DTSB           | 2000        | real-world       | 141.83        | —                  | —                  |
| DTSB           | 2000        | one-point        | 108.14        | 33.69              | 23.75              |
| <b>DTSB</b>    | <b>2000</b> | <b>uniform</b>   | <b>107.56</b> | <b>34.27</b>       | <b>24.16</b>       |
| DTSB           | 2002        | real-world       | 143.09        | —                  | —                  |
| <b>DTSB</b>    | <b>2002</b> | <b>one-point</b> | <b>104.54</b> | <b>38.55</b>       | <b>26.94</b>       |
| DTSB           | 2002        | uniform          | 106.05        | 37.04              | 25.89              |
| SJCS           | 1998        | real-world       | 441.92        | —                  | —                  |
| <b>SJCS</b>    | <b>1998</b> | <b>one-point</b> | <b>396.00</b> | <b>45.92</b>       | <b>10.39</b>       |
| SJCS           | 1998        | uniform          | 411.95        | 29.97              | 6.78               |
| SJCS           | 2000        | real-world       | 450.17        | —                  | —                  |
| <b>SJCS</b>    | <b>2000</b> | <b>one-point</b> | <b>403.20</b> | <b>46.97</b>       | <b>10.43</b>       |
| SJCS           | 2000        | uniform          | 408.25        | 41.92              | 9.31               |
| SJCS           | 2002        | real-world       | 422.77        | —                  | —                  |
| SJCS           | 2002        | one-point        | 410.13        | 12.64              | 2.99               |
| <b>SJCS</b>    | <b>2002</b> | <b>uniform</b>   | <b>402.71</b> | <b>20.06</b>       | <b>4.74</b>        |

Table 5.2: Final fitness of the best individual for each scenario and resulting improvement over the real-world configuration. The best result for each seed is in bold.

| Count | Percentage | Intersection Type                            |
|-------|------------|--|
| 24    | 14.81%     | 4-way and 3-way roundabout                   |
| 18    | 11.11%     | 4-way and 3-way traffic light                |
| 23    | 14.20%     | 4-way and 3-way traffic light (right-on-red) |
| 59    | 36.42%     | 4-way and 3-way all-way stop                 |
| 38    | 23.46%     | 4-way and 3-way priority stop                |

Table 5.3: Intersection type frequencies across final individuals for Chicago map.

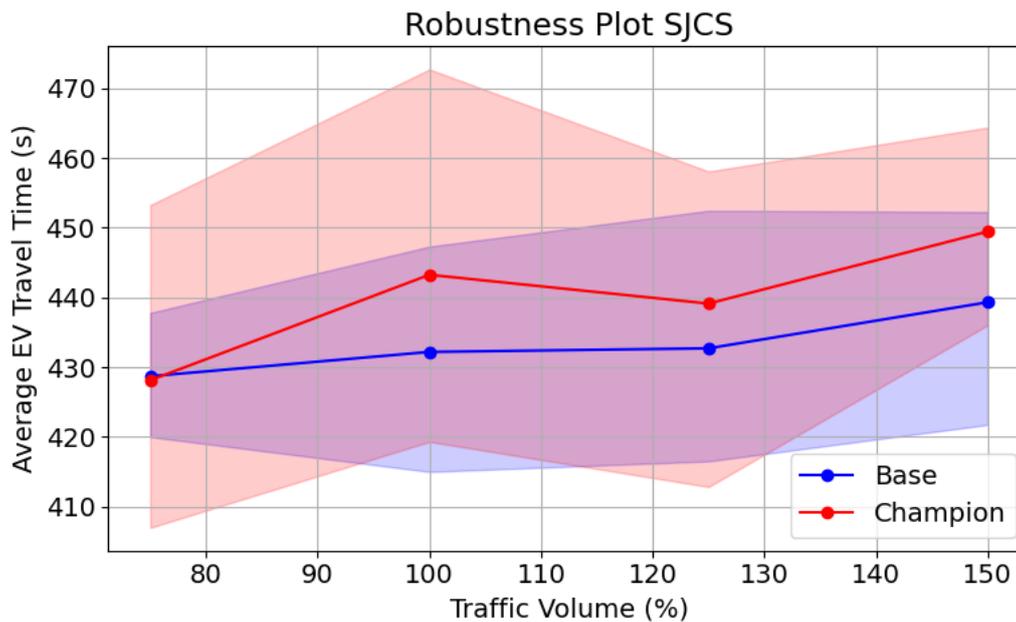


Figure 5.8: Performance comparison of best SJCS solution under different traffic amounts.

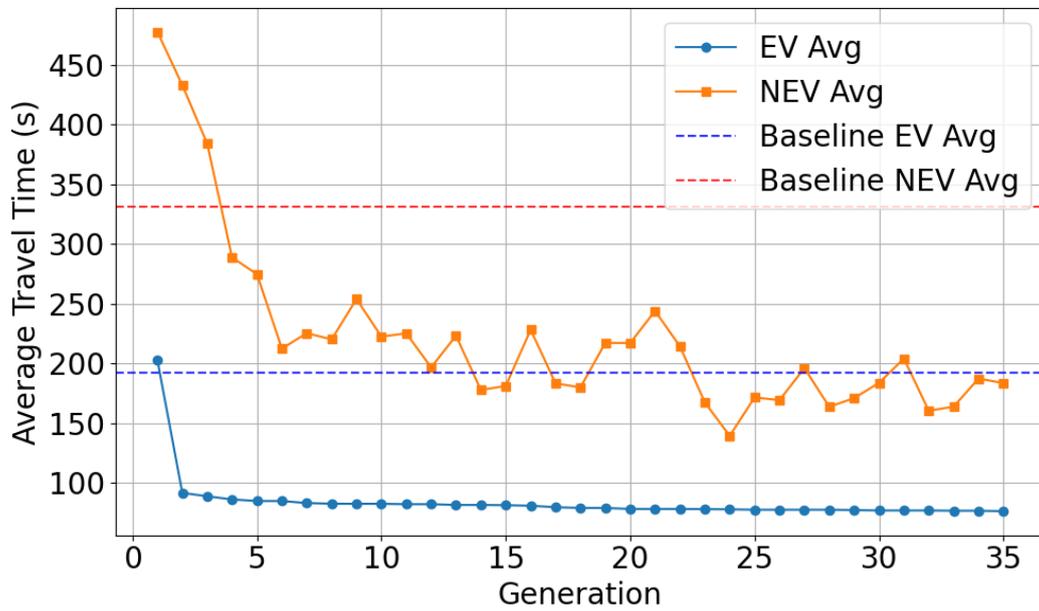


Figure 5.9: Average travel times for Chicago one-point

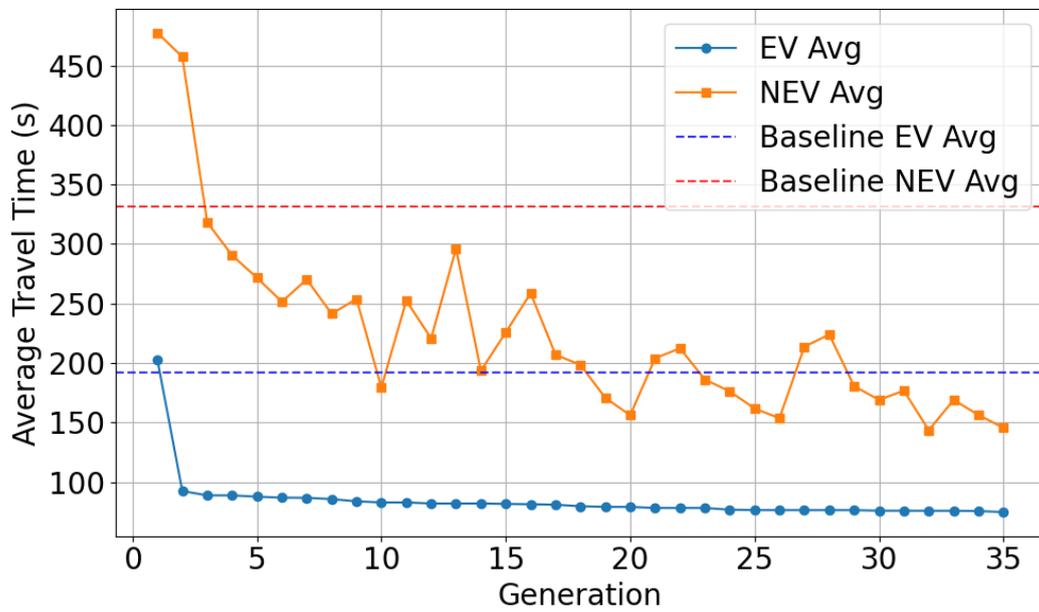


Figure 5.10: Average travel times for Chicago uniform

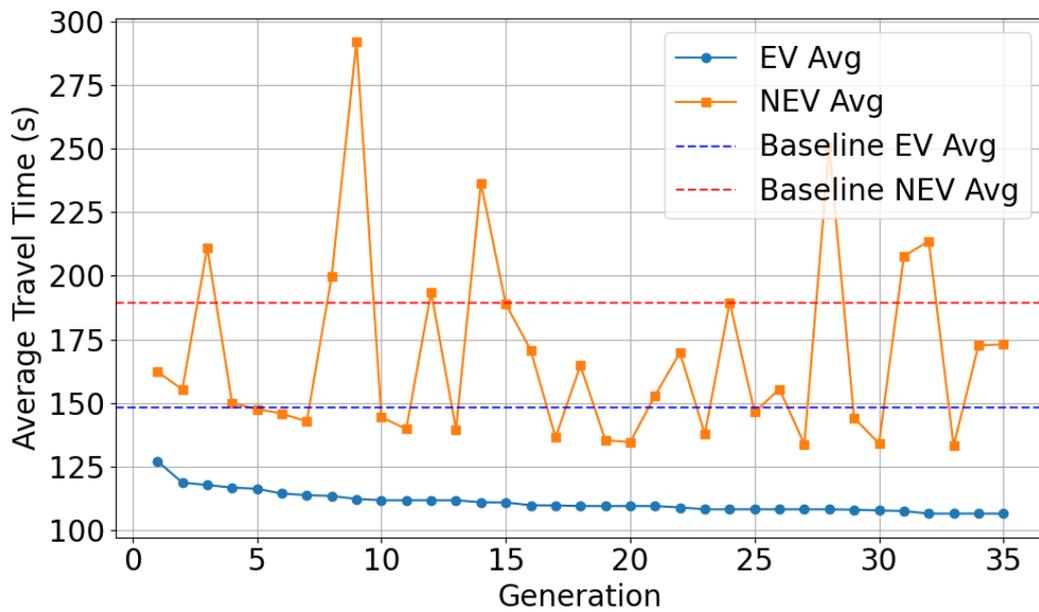


Figure 5.11: Average travel times for DTSB one-point

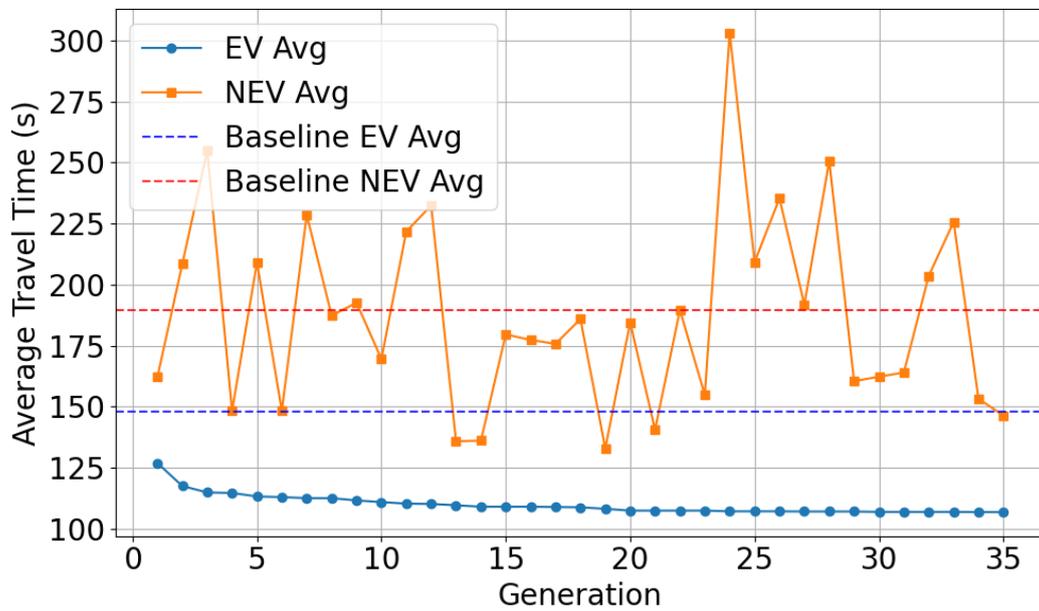


Figure 5.12: Average travel times for DTSB uniform

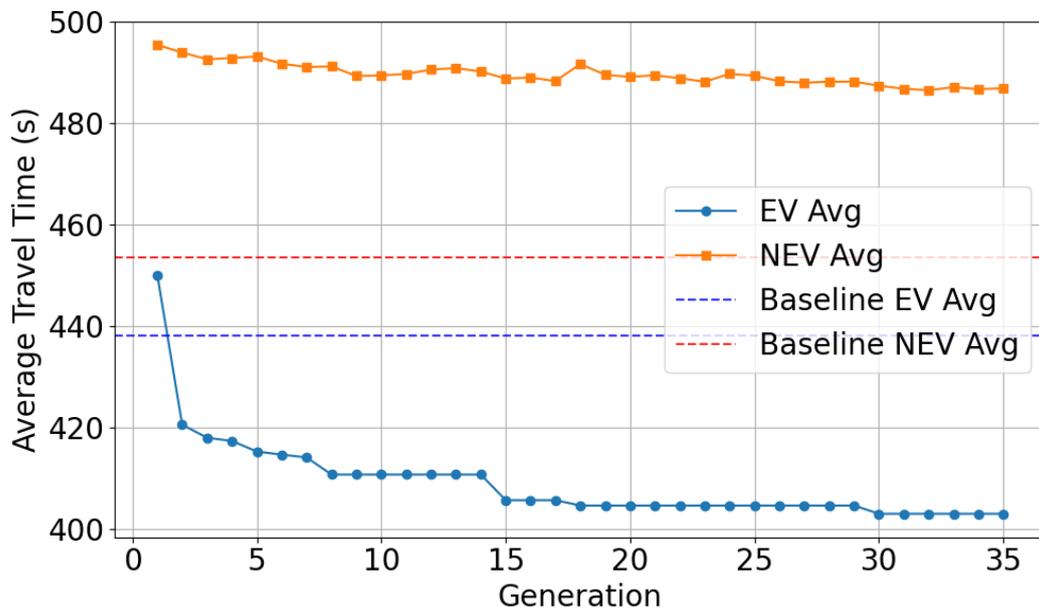


Figure 5.13: Average travel times for SJCS one-point

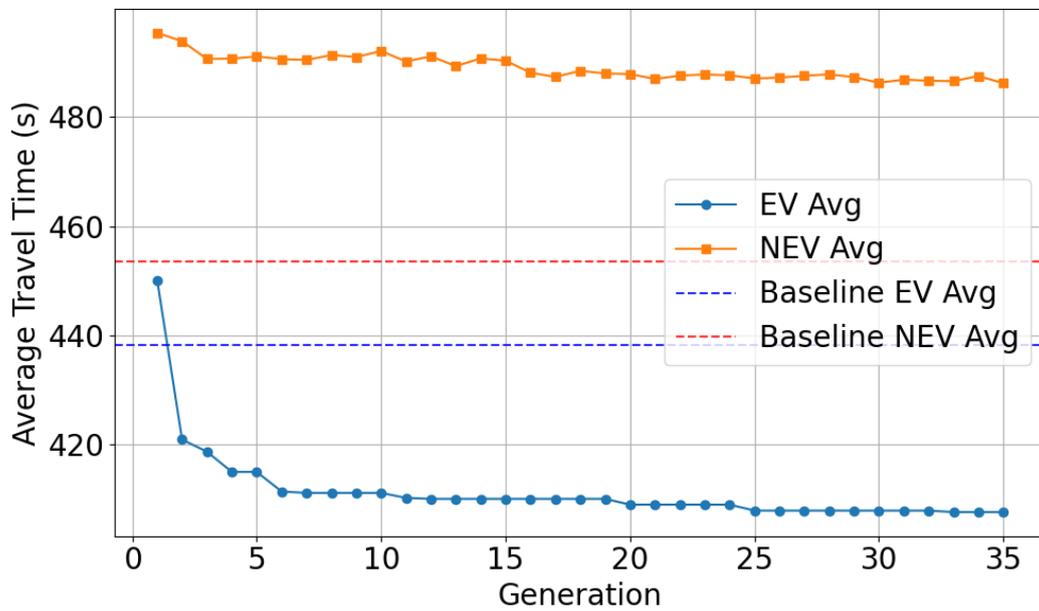


Figure 5.14: Average travel times for SJCS uniform

---

## 6 Conclusions and Future Work

We started this thesis by explaining our goals and motivation, including the research questions we aimed to address. We then gave the background knowledge required to understand the experiments performed. Next, we explored similar works to this, including those involving city planning and those which also had the goal of reducing EV travel times. We then explained how this work is different from previous research and the research gap we aim to fill. After that, we delved into the specifics about how the proposed solution actually works. We gave an overview of all the software and data used, and how the evolutionary algorithm was designed around this. We then gave details about the specific experiments ran and the results of those experiments, which were presented by answering the research questions posed in the beginning.

In this thesis it was shown that the proposed evolutionary approach was able to reduce the average travel times for EVs when compared to the real-world baseline. The improvement was greatest for the smaller, more complicated maps of Chicago and DTSB, with a decrease in travel times as much as 62.89%. There was less improvement for the larger map of SJCS, but travel times could still be reduced by as much as 10.43%. Both the one-point and uniform crossover methods generated configurations that improved the EV travel times over the times in the real-world configuration. The results of the experiments do not suggest that one crossover method is better than the other, as similar convergence behavior was observed for both methods and both were able to find a similar number of optimal solutions. The robustness of the solutions under different traffic amounts varied for each map, with the DTSB solution performing the same or better than the real-world configuration for all traffic volumes and the SJCS solution performing worse with all traffic amounts above 75%. The Chicago map was unable to perform well during the robustness tests for either configuration with any traffic amount above 75%. The impact of the configurations on NEV traffic also varied, with the most improvement on the Chicago map, slight improvement on the DTSB map, and no improvement on the SJCS map.

Although this thesis shows potential for an EA to reduce EV travel times by finding optimal intersection configurations, there were some limitations in the experiments. Firstly, there were some simplifications made when defining

---

the simulated scenarios, as pedestrians and bicycles were removed. This could especially impact the simulation of Chicago, since the area chosen has frequent pedestrian traffic. Also, the simulation of traffic lights only involved simple, timed traffic lights as we were unable to simulate using dynamic or coordinated traffic lights. The improvement seen on the Chicago and DTSB maps may not be as great if the traffic lights in the real-world baseline more closely mirrored the real configuration. The negative impact of not having dynamic traffic lights was especially obvious during the robustness tests on the Chicago map. The examination of travel times throughout the simulation shows that the generated solutions are not sufficient on this map once the simulation reaches full vehicle capacity. Therefore, further experiments using dynamic lights would be necessary to properly assess the improvement for such cities.

The results were also limited by the small number of seeds, the population size, and the reduced number of EA iterations. We may observe different findings when comparing the crossover methods after further testing with more iterations. The scope of simulations was also limited to three maps, all of which are locations in the US, which uses a grid-like structure for city layouts. The results may vary for cities with less rigid layouts. It is also worth noting that the maps sizes were limited to 27-34 intersections. Future work would therefore include addressing these limitations.

Other ideas include a more thorough examination of the impact on NEVs. The average travel times for NEVs were dependent on the map configuration and fluctuated between each generation. Unfortunately, the the simulation logs did not preserve which NEV travel times corresponded to each solution, only the times themselves were gathered for each generation. Further research is required to determine how direct the correlation is between EV travel times and NEV travel times. Further experiments could also include using a modified fitness function that factors in the travel times for NEVs, so that we can ensure more reliable travel times for them, since the currently observed travel times are unpredictable and sometimes longer than in the real-world configuration.

Another suggestion for future work involves combining the solution presented in this paper with solutions presented in related works, for example combining it with the works of (18) and (17), whose papers focused on planning cities at a larger level. The combination could be used to plan intersections after the basic layout for new cities has been determined. In order to do this, the idea in thesis could be expanded to include not just intersections, but also the

direction and number of lanes connecting each intersection. Another future collaboration would be to combine our solution with any of the related works where EV travel times are reduced via dynamic traffic light controls or dynamic path finding. This would mean using an EA to plan the best intersection configurations for EVs ahead of time, and then helping vehicles move efficiently through the city in real time. Overall, this thesis demonstrates that there is potential in using an EA to find ideal intersection configurations to reduce EV travel times. While there were some limitations in the experiments performed, there are also many possibilities for continued research and expansion in this area.

---

# Bibliography

- [1] E. L. d. S. Cabral, W. R. S. Castro, D. R. d. M. Florentino, D. d. A. Viana, J. F. d. Costa Junior, R. P. d. Souza, A. C. M. Rêgo, I. Araújo-Filho, and A. C. Medeiros, “Response time in the emergency services. Systematic review,” *Acta Cirúrgica Brasileira*, vol. 33, pp. 1110–1121, Dec. 2018. Publisher: Sociedade Brasileira para o Desenvolvimento da Pesquisa em Cirurgia.
- [2] J. D. Mayer, “Response Time and Its Significance in Medical Emergencies,” *Geographical Review*, vol. 70, no. 1, pp. 79–87, 1980.
- [3] S. Djahel, N. Smith, S. Wang, and J. Murphy, “Reducing emergency services response time in smart cities: An advanced adaptive and fuzzy approach,” in *2015 IEEE First International Smart Cities Conference (ISC2)*, pp. 1–8, Oct. 2015.
- [4] T. Buffington and O. A. Ezekoye, “Statistical analysis of fire department response times and effects on fire outcomes in the united states,” *Fire Technology*, vol. 55, pp. 2369–2393, November 2019.
- [5] “Community Assessment: Chicago Fire Department.” <https://firecares.nfors.org/departments/77379/chicago-fire-department>. Accessed: Jun 30, 2025.
- [6] “Community Assessment: City of South Bend Indiana Fire Department.” <https://firecares.nfors.org/departments/77889/city-of-south-bend-indiana-fire-department>. Accessed: Jun 30, 2025.
- [7] O. Kramer, *Genetic Algorithm Essentials*, vol. 679 of *Studies in Computational Intelligence*. Cham: Springer International Publishing, 2017.
- [8] S. N. Sivanandam and S. N. Deepa, *Introduction to genetic algorithms*. Berlin ; New York: Springer, 2007.
- [9] R. Kruse, C. Borgelt, F. Klawonn, C. Moewes, G. Ruß, and M. Steinbrecher, “Evolutionäre Algorithmen,” in *Computational Intelligence: Eine methodische Einführung in Künstliche Neuronale Netze, Evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze* (R. Kruse, C. Borgelt, F. Kla-

- wonn, C. Moewes, G. Ruß, and M. Steinbrecher, eds.), pp. 155–167, Wiesbaden: Vieweg+Teubner Verlag, 2011.
- [10] R. Kruse, S. Mostaghim, C. Borgelt, C. Braune, and M. Steinbrecher, *Computational Intelligence: A Methodological Introduction*. Texts in Computer Science, Cham: Springer International Publishing, 2022.
- [11] T. Bartz-Beielstein, J. Branke, J. Mehnen, and O. Mersmann, “Evolutionary algorithms,” *WIREs Data Mining and Knowledge Discovery*, vol. 4, no. 3, pp. 178–195, 2014.
- [12] H.-G. Beyer, E. Brucherseifer, W. Jakob, H. Pohlheim, B. Sendhoff, and T. B. To, “Evolutionary algorithms—terms and definitions.” <https://homepages.fhv.at/hgb/ea-glossary/ea-termsengl.html>, 2002. Accessed: June 16, 2025.
- [13] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT Press, fifth ed., 1999.
- [14] J. Blank, “ga\_basic.png.” <https://pymoo.org/algorithms/soo/ga.html>, 2020. Accessed: May 16, 2025.
- [15] J. Blank and K. Deb, “pymoo: Multi-objective optimization in python,” *IEEE Access*, vol. 8, pp. 89497–89509, 2020.
- [16] J. Blank, “GA: Genetic Algorithm.” <https://pymoo.org/algorithms/soo/ga.html>, 2020. Accessed: May 16, 2025.
- [17] C.-M. Feng and J.-J. Lin, “Using a genetic algorithm to generate alternative sketch maps for urban planning,” *Computers, Environment and Urban Systems*, vol. 23, pp. 91–108, Mar. 1999.
- [18] M. Alghamdi, “Smart city urban planning using an evolutionary deep learning model,” *Soft Computing*, vol. 28, pp. 447–459, Jan. 2024.
- [19] L. Cruz-Piris, M. A. Lopez-Carmona, and I. Marsa-Maestre, “Automated optimization of intersections using a genetic algorithm,” *IEEE Access*, vol. 7, pp. 15452–15468, 2019.
- [20] M. Alam, J. Ferreira, and J. Fonseca, “Introduction to intelligent transportation systems,” in *Intelligent Transportation Systems: Dependable Vehicular Communications for Improved Road Safety* (M. Alam, J. Fer-

- reira, and J. Fonseca, eds.), pp. 1–17, Cham: Springer International Publishing, 2016.
- [21] D. W. Matolak, “V2V Communication Channels: State of Knowledge, New Results, and What’s Next,” in *Communication Technologies for Vehicles* (M. Berbineau, M. Jonsson, J.-M. Bonnin, S. Cherkaoui, M. Aguado, C. Rico-Garcia, H. Ghannoum, R. Mehmood, and A. Vinel, eds.), (Berlin, Heidelberg), pp. 1–21, Springer, 2013.
- [22] M. M. Antony and R. Whenish, “Advanced driver assistance systems (adas),” in *Automotive Embedded Systems: Key Technologies, Innovations, and Applications* (M. Kathiresh and R. Neelaveni, eds.), pp. 165–181, Cham: Springer International Publishing, 2021.
- [23] J. Barrachina, P. Garrido, M. Fogue, F. J. Martinez, J.-C. Cano, C. T. Calafate, and P. Manzoni, “Reducing emergency services arrival time by using vehicular communications and Evolution Strategies,” *Expert Systems with Applications*, vol. 41, pp. 1206–1217, Mar. 2014.
- [24] N. Capodiecici, R. Cavicchioli, F. Muzzini, and L. Montagna, “Improving emergency response in the era of ADAS vehicles in the Smart City,” *ICT Express*, vol. 7, pp. 481–486, Dec. 2021.
- [25] A. El-Dalil, M. Sharkas, and M. Khedr, “Priority level mutualism for emergency vehicle using game theory,” in *2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pp. 75–80, June 2017.
- [26] L. Zhong and Y. Chen, “A Novel Real-Time Traffic Signal Control Strategy for Emergency Vehicles,” *IEEE Access*, vol. 10, pp. 19481–19492, 2022.
- [27] W. Kang, G. Xiong, Y. Lv, X. Dong, F. Zhu, and Q. Kong, “Traffic signal coordination for emergency vehicles,” in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 157–161, Oct. 2014. ISSN: 2153-0017.
- [28] M. A. Chentoufi and R. Ellaia, “Adaptive traffic signal optimization considering emergency vehicle preemption and tram priority using PVS algorithm,” in *Proceedings of the 3rd International Conference on Smart City Applications, SCA ’18*, (New York, NY, USA), pp. 1–8, Association for Computing Machinery, Oct. 2018.

- [29] P. S. Chakraborty, A. Tiwari, and P. R. Sinha, “Adaptive and Optimized Emergency Vehicle Dispatching Algorithm for Intelligent Traffic Management System,” *Procedia Computer Science*, vol. 57, pp. 1384–1393, Jan. 2015.
- [30] P. Alvarez Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, “Microscopic Traffic Simulation using SUMO,” in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, (Maui, USA), pp. 2575–2582, IEEE, Nov. 2018.
- [31] Q. Lu and K.-D. Kim, “A genetic algorithm approach for expedited crossing of emergency vehicles in connected and autonomous intersection traffic,” *Journal of Advanced Transportation*, vol. 2017, no. 1, p. 7318917, 2017.
- [32] German Aerospace Center (DLR) and others, “Emergency vehicle simulation.” <https://sumo.dlr.de/docs/Simulation/Emergency.html>, 2024. Last edited: Jan. 6, 2024. Accessed: Mar. 15, 2025.
- [33] German Aerospace Center (DLR) and others, “Faq.” <https://sumo.dlr.de/docs/FAQ.html>. Last edited: Jun. 1, 2025. Accessed: Aug. 24, 2025.
- [34] pymoo developers, “pymoo: Python multi-objective optimization framework – ga.py.” <https://github.com/anyoptimization/pymoo/blob/4745ecb/pymoo/algorithms/soo/nonconvex/ga.py#L22>, 2025. Accessed: Aug 6, 2025.
- [35] pymoo developers, “pymoo: Python multi-objective optimization framework – tournament.py.” <https://github.com/anyoptimization/pymoo/blob/4745ecb/pymoo/operators/selection/tournament.py#L26>, 2025. Accessed: Aug 6, 2025.
- [36] OpenStreetMap contributors, “Openstreetmap.” <https://www.openstreetmap.org/copyright>, 2025. Map data available under the Open Database License (ODbL).
- [37] Illinois Department of Transportation, “Traffic counts.” <https://www.gettingaroundillinois.com/Traffic%20Counts/index.html>. Accessed: Mar. 9, 2025.

- [38] Indiana Department of Transportation, “Traffic viewer.” [https://indot.public.ms2soft.com/TDMS.UI\\_Core/trafficviewer](https://indot.public.ms2soft.com/TDMS.UI_Core/trafficviewer). Accessed: Mar. 9, 2025.
- [39] Federal Highway Administration, “Traffic monitoring guide,” fhwa report no. fhwa-pl-17-003, U.S. Department of Transportation, Federal Highway Administration, Washington, DC, Oct. 2016. Updated October 2016. [Online]. Available: [https://www.fhwa.dot.gov/policyinformation/tmguidetmg\\_fhwa\\_pl\\_17\\_003.pdf](https://www.fhwa.dot.gov/policyinformation/tmguidetmg_fhwa_pl_17_003.pdf). Accessed: Apr. 26, 2025.
- [40] German Aerospace Center (DLR) and others, “Traffic lights.” [https://sumo.dlr.de/docs/Simulation/Traffic\\_Lights.html](https://sumo.dlr.de/docs/Simulation/Traffic_Lights.html). Last edited: Jun. 30, 2025. Accessed: Aug. 23, 2025.
- [41] German Aerospace Center (DLR) and others, “Trip.” <https://sumo.dlr.de/docs/Tools/Trip.html>. Last edited: July 18, 2025. Accessed: Aug. 3, 2025.
- [42] German Aerospace Center (DLR) and others, “duarouter.” <https://sumo.dlr.de/docs/duarouter.html>. Last edited: July 21, 2024. Accessed: July 30, 2025.
- [43] German Aerospace Center (DLR) and others, “netedit usage examples.” <https://sumo.dlr.de/docs/Netedit/neteditUsageExamples.html>. Last edited: Jan. 14, 2025. Accessed: Aug. 16, 2025.
- [44] K. Deb and S. Agrawal, “Understanding interactions among genetic algorithm parameters,” in *Foundations of Genetic Algorithms V* (W. Banzhaf and C. Reeves, eds.), (San Mateo, CA), pp. 265–286, Morgan Kaufmann, 1999. Accessed via author’s preprint version.

# Declaration of Authorship

Thesis:

Name:

Surname:

Date of birth:

Matriculation no.:

I herewith assure that I have written the present thesis independently, that the thesis has not been partially or fully submitted as graded academic work and that I have used no other means than the ones indicated. I have indicated all parts of the work in which sources are used according to their wording or according to their meaning.

I am aware of the fact that violations of copyright can lead to injunctive relief and claims for damages of the author as well as a penalty by the law enforcement agency.

Furthermore, I confirm that I am aware that the use of content (including but not limited to text, figures, images and code) generated by artificial intelligence (AI) in the thesis must be disclosed. In those cases I have specified the AI system used, I have marked the specific sections of the thesis where AI-generated content was used, and I have provided a brief explanation of the level of detail at which the AI system was used to generate the content. I also stated the reason for using the tools.

I assure that even if a generative AI system has been used, the scientific contribution has been made entirely by myself.

Laurel Maisha Raven

Atlanta, Georgia, August 25, 2025