NTNU
Norwegian University of
Science and Technology

Real Time Intersection Management using a Multiobjective Evolutionary Algorithm

Håkon Ørstavik Dissen
Jostein Aune Solaas

Master of Science in Computer Science

Submission date:	June 2016
Supervisor:	Keith Downing, IDI
Co-supervisor:	Kazi Shah Nawaz Ripon, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem description

The aim of this masters thesis is to explore the usage of a Multiobjective Evolutionary Algorithm (MOEA) to manage an intersection in real time. In order to achieve this an intersection manager that divides the time into single time steps has to be implemented. The vehicular behaviour in single time steps can then be optimised by themselves. There is no direct way to map optimisation objectives in single time steps to overall performance. Different optimisation objectives, with different goals in terms of overall performance, will be used at the same time. The performance evaluation of the intersection manager will be performed in a simulator.

Supervisor: Keith Downing
Co-supervisor: Kazi Shah Nawaz Ripon

Summary

The purpose of this thesis is to investigate the real time use of a Multiobjective Evolutionary Algorithm (MOEA) for intersection management.

Intersection managers (IMs) using deterministic methods for real time intersection management of autonomous vehicles, have been shown to drastically improve traffic flow and congestion over conventional traffic lights. Others, using evolutionary algorithms or other search algorithms, have found methods for approaching the optimal sequencing of vehicles in single states. However, a structured literary review revealed no previous studies on the combination of these methods. In this thesis we used an MOEA to optimise single time steps based on several objectives, and then examined how this translated to behaviour over a continuous time frame.

We have proposed and implemented an IM. In order to experimentally test it a simulator was first adapted for our purposes. Splitting the continuous problem into several discrete time steps introduced the need for two time step parameters. In order to ensure a fair evaluation of the IM, and to study the effects of different time steps, the IM was tested with different values for those parameters. The IM was then tested for specific scenarios, and for continuous operation with different amounts of traffic. To compare the performance of the IM in different settings four performance metrics were used: throughput, mean evacuation time (MET), total loss of kinetic energy and the amount of collisions.

After finding reasonable values for the time step parameters, the IM was shown to route traffic through the intersection with an MET close to the measured optimal value for low and medium amounts of traffic. However, for high amounts of traffic, the complexity of optimising each state becomes too large. As a consequence, the MOEA failed to avoid collisions within the given amount of evaluations. The total kinetic energy lost was found to be dependent on the time step parameters, where smaller values lead to a greater loss of kinetic energy.

Sammendrag

(This is a Norwegian translation of the abstract)

Formålet med denne avhandlingen er å utforske bruken av en multiobjektiv evolusjonær algoritme (MOEA) til håndtering av autonome biler i kryss.

Bruk av en "intersection manager" (IM) som benytter seg av deterministiske metoder for håndtering av autonome biler i kryss i sanntid, har i tidligere forsøk vist å forbedre trafikkflyten betydelig sammenliknet med vanlige trafikklys. I andre studier, der man har brukt evolusjonære algoritmer eller andre søkealgoritmer, har man utviklet metoder for å tilnærme seg optimal sekvensering av biler i enkelttilfeller. Et strukturert litteratursøk fant ingen tidligere studier der man hadde kombinert disse to metodene. I denne oppgaven benyttet vi en MOEA for å optimalisere enkeltsteg basert på flere mål, og så sammenliknet vi hvordan dette påvirket systemets oppførsel over tid.

Vi har foreslått og implementert en IM. For å kunne utføre eksperimentelle tester av denne benyttet vi en simulator modifisert for å dekke våre behov. Å dele det kontinuerlige problemet i enkelte tidssteg introduserte behovet for to tidsstegparamtere. For å sikre en rettferdig evaluering av vår IM, og for å kunne studere effektene av forskjellige tidsstegstørrelser, ble denne testet med forskjellige verdier for de to tidsparameterene. Deretter evaluerte vi vår IM ved ulike, spesifiserte betingelser, og for kontinuerlig operasjon med ulike trafikkmengder. Vi brukte fire forskjellige ytelsesmål: gjennomstrømming, gjennomsnittlig evakueringstid, totalt tap av kinetisk energi og antall kollisjoner.

Etter at rimelige verdier for tidsstegparameterene var funnet ble det vist at vår IM, ved små og mellomstore mengder trafikk, er i stand til å styre trafikken gjennom krysset med en gjennomsnittlig evakueringstid tilnærmet den målte optimalverdien. Men ved større trafikkmengder vokser kompleksiteten av å optimalisere hver enkelttilstand for fort. Som en konsekvens av dette fant vi at vår IM ikke alltid er i stand til å finne en løsning uten en kollisjon innenfor tidsrammene. Den totale mengden tapt kinetisk energi ble vist å være avhengig av tidsstegparameterene, der lavere verdier forårsaket et høyere tap av kinetisk energi.

Preface

This thesis concludes our master's degree in Computer Science at The Norwegian University of Science and Technology (NTNU). The assignment was given in January 2016 and completed in June 2016.

We would like to thank our supervisor Kazi Shah Nawaz Ripon, PhD, for his valuable guidance and support throughout the project. In addition, we would like to thank Professor Keith Downing and Professor Pauline Haddow for their help with the project.

Håkon Dissen and *Jostein Solaas*

Table of Contents

List of Tables

List of Figures

Abbreviations

IM	=	Intersection Manager
MOEA	=	Multiobjective Evolutionary Algorithm
EA	=	Evolutionary Algorithm
GA	=	Genetic Algorithm
MOP	=	Multiobjective Optimisation Problem
V2V	=	Vehicle to Vehicle
V2I	=	Vehicle to Intersection
CSG	=	Compatible Stream Groups
FG	=	Fundamental mini-groups
DM	=	Decision Maker
EP	=	Evolutionary Processor
MET	=	Mean Evacuation Time

Chapter 1

Introduction

In this work we have studied the field of intersection management. We present a method for real time intersection management by using a Multiobjective Evolutionary Algorithm (MOEA). The method involves dividing the continuous problem of intersection management into independent time steps. The combined independent behaviour in these time steps make up the continuous behaviour of the proposed method.

This chapter is the introductory chapter. Section 1.1 provides an introduction to the background and motivation of this thesis. Then, section 1.2 explains the research goal and questions before section 1.3 sums up our contributions to the field and 1.4 gives a brief introduction to the research method. Finally, section 1.5 provides an overview of the whole thesis.

1.1 Background and motivation

The main motivation of intersection management is to reduce the amount of time vehicles spend in intersections as a way to ease traffic congestion, mostly in urban scenarios. Dresner et al. [8] state that, in 2008 Americans were spending an average of 46 hours a year per capita in congested traffic, up from 16 hours in 1982. In the EU the total cost of traffic congestion was estimated at 1% of total GDP in 2010 [18]. Another goal is to reduce the total emissions of cars idling by, or traversing an intersection.

No matter the focus of the optimisation, collision avoidance is always essential in intersection management approaches. Restricting vehicles to non-conflicting trajectories, like many current intersection management implementations [24, 25, 22] (including regular traffic lights), puts some restrictions on available methods, but ensures the safety of all the cars involved. The introduction of autonomous vehicles allows for fine-level control and predictability of vehicle behaviour [8]. Therefore, this work will impose no restrictions on vehicles sharing the space in the intersection, other than the fact that they should not collide. It can be interesting to look at the problem in the context of multiple intersections working together, but we will focus on single intersections.

Most of the current methods for intersection management, presented in chapter 2, use deterministic algorithms for continuous intersection management or explore the use of more stochastic methods for optimising single scheduling scenarios. This work proposes and investigates an intersection manager (IM) based on a hybrid of those two concepts. It looks at splitting the continuous problem of intersection management into smaller independent scenarios, then it explores the usage of an MOEA to find solutions for each scenario. This work also investigates how those scenarios together translate into continuous behaviour.

1.2 Research Goal and Questions

Goal statement: *The goal of the project is to investigate the real time use of a Multiobjective Evolutionary Algorithm (MOEA) to manage traffic through an intersection.*

Most of the current literature on intersection management is focusing on using MOEAs to optimise parameters for traffic light intersections, or specific algorithms for real time control. Our goal is to look at the effectiveness of real time usage of an MOEA given full control of the speeds of all incoming cars. In this work, *effectiveness* is evaluated by looking at four different performance metrics, namely:

- Throughput. Defined as the amount of cars that pass through the intersection during a simulation.
- The mean evacuation time (MET). MET is a measure of the average time the passing vehicles spend traversing the intersection.
- The amount of collisions. There is no inherent guarantee that the proposed IM will avoid collisions. Therefore, exploring what scenarios, and internal shortcomings, that cause collisions is important.
- The total kinetic energy lost. This is done by measuring the total kinetic energy lost from reducing the speeds of vehicles. Other losses of energy are not considered.

In order to give the MOEA low level control of traffic in the intersection the vehicles will be modelled as travelling along a predefined path, with a speed determined by the MOEA.

Research question 1: *What are the effects of using independent time step modelling to determine actions for the system, when looking at effects at an infinite time horizon?*

The goal of this research question is to study how splitting the large problem of organising the cars over time into smaller time steps affects performance. It is hard to predict whether the algorithm can be tuned to avoid problems that could occur over time, such as single lane starvation or deadlocks. Here we will also look at the effects of using different values for the time step sizes: t_{main} that determines for how long each solution is deployed and t_{sim} that specifies how much extra time beyond t_{main} each solution is evaluated for. These variables are further described in section 3.1.1.

Single lane starvation is when one lane is left unreasonably long without releasing any cars. For instance, this could occur if other lanes have more cars; dominating in terms of throughput. Deadlocks are when the vehicles have positioned themselves so that they cannot move without colliding.

Research question 2: *How does the complexity of solving the independent time steps scale with the amount of cars present in the intersection?*

The more vehicles present within the control range of the intersection, the higher the complexity of the solution space the MOEA has to traverse. We wish to investigate the scaling of the system, and see if it is plausible for real time usage in the real world.

Research question 3: *How do the chosen objective functions contribute to the effectiveness of the intersection manager?*

When using independent time step modelling there is no direct connection between the objectives being optimised for and the overall performance metrics. Understanding how the chosen objectives contribute to the overall effectiveness of the IM, and under what circumstances they conflict, is important to understand why and how it fails/works. The optimisation objectives chosen for the IM are available in section 3.2.1.

1.3 Contributions

The field of intersection management has changed in the advent of autonomous vehicles. As they provide much better support for implementing central optimal intersection routing strategies this thesis contributes a method to that end. In addition, because the proposed method of dividing the continuous problem into smaller discrete time steps has never been attempted for intersection management (to the authors' knowledge), this thesis aims to provide insight as to whether that is a good approach to intersection management.

1.4 Method

In order to answer the research questions posed in the last section we are going to implement an IM that will use an MOEA to find vehicle speeds in accordance with the given objectives. The speeds will be deployed for a given amount of time, before the process repeats again. The IM will have the ability to explicitly control the speed of every car at every time step. A simulator, outlined in section 3.1.5, will be used in order to evaluate the fitness of each solution.

The goal of the IM itself is to optimise its performance in terms of the performance metrics. There is no direct relation between the performance metrics, evaluated over time, and what is being optimised for. Therefore it was decided to use several distinct objectives, each objective seeking to better the behaviour of the IM at a single independent time step. In order to avoid convergence toward a solution adapted for just one of these partial objectives, and instead get a more diverse Pareto-set for decision making, an MOEA was used instead of a regular Evolutionary Algorithm.

Complete replication of real world scenarios is not a goal of this work. Therefore, the simulator used to evaluate solutions will also be used to simulate the real world. No simulator was found to meet all of the requirements in this project. Thus, we have created a modified version of the AIM4 simulator [1]. The vehicles will be modelled as single entities travelling along predefined paths (explained in section 3.1.2), where the only variable parameter is a target speed that the vehicle should accelerate/decelerate to. Vehicle collision and control after traversing through the intersection is considered beyond the scope of this thesis. So are the interactions between multiple intersections. The intersection here meaning the control area of the IM, defined in section 3.1.1.

1.5 Overview

This thesis is divided into five chapters. Chapter 2 explains the background and related work in intersection management. It also discusses MOEAs, and why they are relevant for this thesis. Chapter 3 revolves around the technical decisions made for the implementations in the experiments, in addition to the methodology used in later experiments. Chapter 4 describes the experimental setup in detail. It also provides the results of the experiments and discusses the findings. In chapter 5 we present a conclusion of our studies, and make suggestions for further research.

Chapter 2

Background

This chapter explains the background of intersection management and how the field has evolved to its current state.

Intersection management is a problem that has been tackled in many different ways. As such, the methods presented in this chapter have very different approaches, and assume different levels of capabilities of the drivers of the cars. Recent methods often assume the driver to be autonomous [8, 25, 22], and will therefore allow new and creative approaches that are otherwise impossible when the vehicles are under human control. However, the autonomous approaches all build on ideas and results from research on human drivers, and have to be understood in that context.

A problem with the research into intersection management as it stands is that the papers presented do not use the same system for benchmarking their results. They use different simulators with different parameters and often completely different simulation models. The result is that they can be difficult to compare based on evidence of performance. A common method of measurement is comparison to regular traffic lights, but no standard for *normal* has been defined. However, the methods presented differ so greatly that a comparison of basic methods is still worthwhile. This chapter will also point out important inspirational sources for this thesis.

After discussing intersection management, we will discuss Evolutionary Algorithms (EAs), Genetic Algorithms (GAs), Multiobjective Optimisation Problems (MOPs) and MOEAs.

As described in section 2.4, very little related work was found. To the authors' knowledge, no complete related projects exist. The methods outlined in section 2.2 are the most relevant projects found for this thesis.

2.1 Intersection management

Perhaps the most common way of managing busy intersections, aside from explicit right of way rules or roundabouts, is by traffic lights. Fixed time cycles are the simplest method of traffic light operation.

In 1981 a system called SCOOT (Split Cycle Offset Optimisation Technique) [13] was introduced. SCOOT allows regular traffic lights to adapt to incoming traffic with sensors along the road. While SCOOT is an improvement over the normal timed cycles approach it is old and does not take advantage of modern vehicle to vehicle communications. It also relies on infrastructure in the intersection to sense the locations of cars, even though new cars may know their own location.

Any method based on traffic lights guarantees collision safety as long as all vehicles adhere to the rules of the intersection, and no external effects (such as weather) act on the vehicles. This sets a standard for safety that other methods will have to tangent or beat in order to be considered viable. We find it unlikely that the public will adopt a new approach to intersection management if it can not be shown to guarantee safe traversal of intersections.

2.1.1 Communication

All of the intersection management approaches in this chapter base themselves on some sort of communication. Two different communication paradigms are most common: vehicle to vehicle (V2V) , and vehicle to infrastructure (V2I). V2I may also be referenced as vehicle infrastructure integration in some literature [25]. The actual workings of such communication protocols are beyond the scope of this thesis, but the main principles are relevant. In V2I, all vehicles communicate with a central server that can make decisions in regards to intersection management or simply relay messages safely between the cars. The disadvantage of V2I is that every intersection has to have some sort of physical infrastructure, severely limiting the ease of deployment. In contrast, V2V allows vehicles to communicate with each other and requires no central infrastructure. This removes the requirement for a central server, but introduces complexity in terms of vehicle communication. Researchers have come up with effective methods for organising vehicle ad-hoc networks [20], allowing control systems to be managed by the cars internally [17]. Vehicle ad-hoc networks in this context means a network that exists only for a limited amount of vehicles that need to communicate, where vehicles are added to the network as they approach the area or group affected, and are taken out as they leave. These networks can be used to manage the traffic for a given area, without requiring further infrastructure, such as a central server.

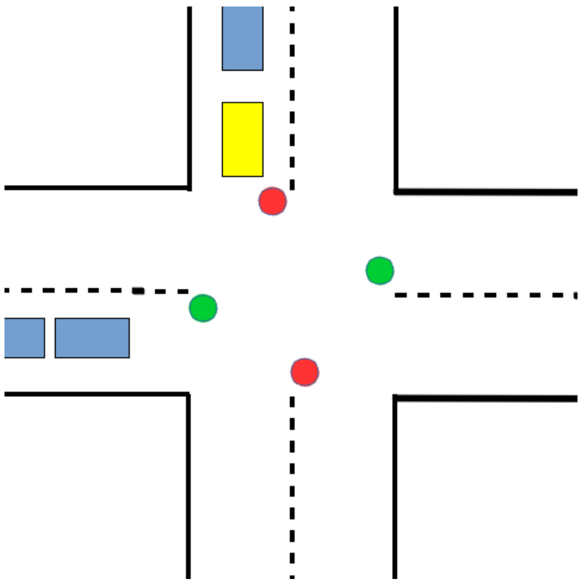

Figure 2.1: VTL in action. The yellow car, being the closest to the intersection, has been chosen as the leader. It broadcasts a green light for conflicting lanes and a red light for its own lane.

2.1.2 Virtual traffic lights

The introduction of V2V allows cars approaching intersections to communicate. VTL (Virtual Traffic Lights) [10] is a method that clearly shows some benefits from V2V. The system works by virtualising a regular traffic light intersection. V2V is key for this approach, because it allows the cars to communicate instructions to each other over an ad hoc network at the intersection.

The cars involved in an intersection elect a leader amongst themselves. The leader then takes control of the intersection and broadcasts what signal affects each incoming road at what time [10]. How each agent communicates the phase of the light-signal to its driver is undetermined, but a system is proposed where each car has a monitor on the dashboard showing the driver the current state. This could easily be extended to autonomous drivers.

VTL can be broken into three simple steps [10]. First, the vehicles approaching an intersection communicate amongst themselves to determine a leader for an intersection, unless one has already been chosen. Second, the leader assigns his incoming lane a virtual red light (this information is distributed over the network), and determines the light timings for the other lanes based on information gathered from other vehicles. Finally, when the leader determines that he is allowed to pass through the intersection, he passes the role of leader to any car that is waiting in the front of any queue with a red light. If no other car is present no new leader will be assigned. The process starts over when a new set of cars are approaching the intersection. See figure 2.1 for a visual representation of the first and second step.

The implementation relies on the following assumptions [10, 17]:

- All vehicles are *DSRC* [14] devices, meaning they can communicate with each other.
- All vehicles share the same digital road map, guaranteeing that they agree on where in the world they are.
- All vehicles are equipped with GPS, guaranteeing that they share a global time and can know their position with lane-level accuracy.
- The security, reliability and latency of the communication is assumed to be adequate for VTL.

When compared to an unintelligent IM, VTL improves upon two main areas:

- When the traffic is low the wait time is significantly reduced; no vehicles have to wait to cross an empty intersection.
- When there are high amounts of traffic in intersections that do not have a traffic light, VTL can dynamically create one.

According to Ferreira et al. [10] VTL improves traffic-flow by more than 60% over a simulation of regular traffic lights in Porto, Portugal. A subsequent study found throughput improvements over regular traffic lights by up to 35% in a simulated single intersection [17].

VTL is interesting in the scope of this work because it shows that infrastructure normally existing in the physical intersections can be moved onto virtual infrastructure on an ad hoc network between the vehicles. However, the use of traffic lights in itself is a limiting factor. The security of VTL is compromised because two leaders may be elected at once due to packet loss. However, in a realistic simulation it has been shown that this can quickly be mitigated [17].

2.2 Autonomous Intersection Management

Autonomous vehicles introduce capabilities far beyond the capabilities of human drivers. A very important feature of these new vehicles, in regards to the problem of intersection management, is the capabilities of cars to sense their location in the world and use sophisticated measures to plan their paths of travel [8]. This allows the intersection controller much finer control over the specific vehicles, opening up for more advanced methods of IM.

Because autonomous vehicles have such fine control of their movements newer Autonomous Intersection Management (AIM) research [8, 25, 22] use a system of intersection organising where cars have to travel along predefined trajectories. These trajectories traverse the intersection from one source lane to one destination lane. Trajectories may cross each others paths, and such trajectories are referred to as *conflicting* trajectories. Approaches

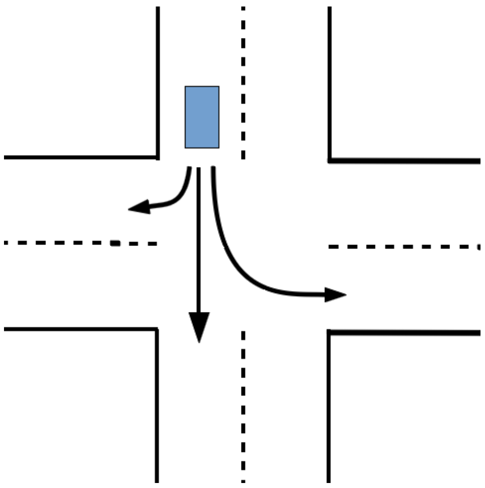

Figure 2.2: The three possible trajectories the vehicle may follow when travelling from the south bound lane.

differ in whether they allow simultaneous traversal of conflicting trajectories. Trajectories are also referred to as *streams*. There are no duplicate trajectories. Figure 2.2 shows the trajectories of the southbound lane in a simple four way intersection.

FCFS (based on the first-come first-served principle of the same name) is an intersection control mechanism proposed by Dresner and Stone of the *AIM project* at the University of Texas at Austin [8]. It relies on a central IM that delegates time in an intersection, known as reservations, to vehicles that wish to travel through on predefined trajectories. The IM communicates with the vehicles over V2I. Unlike other systems [25, 22], vehicles wanting to travel along conflicting trajectories are allowed to occupy the intersection at the same time. In order to avoid collisions the IM breaks the centre of the intersection down to a square grid (as shown in fig 2.3). This grid, over time, can be viewed as a shared resource between the vehicle agents [22]. The problem now becomes allocating this shared resource. If a vehicle wants to pass through it asks the IM if it may pass at a certain time and at a certain speed, it asks for a reservation, and is allowed if the IM can verify that it will not collide with any other traversals it has scheduled. The collision check is done by the vehicles reporting their planned paths and checking if any of the squares the vehicle is going to occupy will be occupied by another at the same time. Just like a hotel holding reservations, the agents do not need to know the system with which the manager approves reservations, just whether they have been given one or not. The hotel manager will accept reservations based on who asks first, unless there is a conflicting reservation where the car in front of the asking car has reserved a time of passage after the one attempting to be scheduled. It is important to note that the system does not do anything in particular to achieve the *optimal* ordering of the cars. However, the authors suggest that the IM could delay its responses in order to evaluate the ordering.

FCFS has been shown to significantly outperform traffic lights, stop signs, and overpass

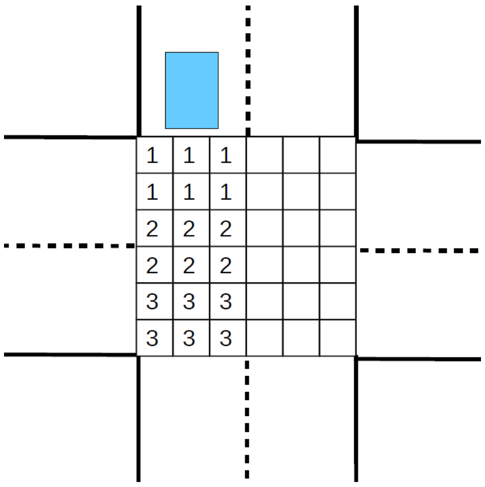

Figure 2.3: What grid cells a car travelling south at a speed of 2 cells per time step is going to occupy.

-regulated intersections in terms of vehicle delay [8, 7].

The method proposed in this thesis builds upon some of the ideas introduced in AIM, most importantly the idea that the cars can simulate their own behaviour based on parameters individual to each car, and having cars travel along predefined paths. In addition, the fact that autonomous vehicles can perform maneuvers, thought too dangerous for humans, in order to save time makes the field of AIM more viable.

2.2.1 Vehicle sequencing

As seen previously in this chapter AIM is generally managed by some explicit protocol; either implemented in a central controller or distributed among the agents passing the intersection. While these solutions to AIM gave improved throughput performance compared to standard actuated traffic lights, their solutions do not necessarily provide the *optimal* vehicle sequencing.

Wu et al. [21] showed that vehicles passing an intersection can be modelled as an ordering problem, and propose a searching algorithm to find the optimal order for scenarios of vehicles approaching an intersection. While they show that their algorithm is, in fact, able to find the optimal ordering they admit that the computational expense of doing so is too high for the system to be put to use in a real-world scenario. Yan et al. [25] build on their work and propose a genetic searching approach that also takes into account other properties of vehicle sequencing in order to reduce the complexity of the problem.

Yan et al. [25] starts by looking at the intersection as a set of stream groups. Stream groups are defined as sets of vehicle trajectories. They then go on to divide the streams into subgroups known as Compatible Stream Groups (CSGs), which consist of trajectories that do not intersect, guaranteeing that vehicles will not collide as long as only one CSG is active at a time. In a simple two-lane four-way intersection two simple CSGs would be the paths going straight through, while perpendicular to each other. Each stream may be present in more than one CSG.

The vehicles in each group are divided into smaller groups known as Fundamental mini-groups (FGs) that will pass the intersection as one entity. These groups derive from properties obtained from optimal solutions and will therefore not jeopardise the optimality of the problem [23, 24]. Combining the cars into FGs reduces the complexity of the problem, allowing the GA to find a solution faster [25].

Now that the problem has been simplified into the ordering of passages of FGs Yan et al. [25] propose using a GA (described in section 2.3) to determine a good order. They use the overall evacuation time (the time until the last vehicle has exited the intersection) of each solution as an objective function and determine pairings via the roulette wheel. In addition to the GA they propose a dynamic programming approach that is sure to find the optimal solution, albeit in a very long time (dynamic programming uses 6.24s for a 100 vehicle problem while the GA finds a reasonable solution in 0.119s). The use of the dynamic programming algorithm is primarily to benchmark the solutions found by the GA. For the worst case scenario, consisting of four lanes each way in a four-way intersection, the GA is able to get within 10.6% of the optimal solution after only 0.642 seconds [25]. The best case scenario of ten vehicles in two lanes can be solved by the GA in 0.075 seconds, yielding optimal results.

It is important to note that this approach does not use more than one objective function, namely overall evacuation time. There is no consideration of other important objectives, such as environmental concerns.

2.3 Evolutionary Algorithms

Search and optimisation problems often present with large and complex search spaces. EAs are used to find a good solution to these problems, without exploring the entire solution space. They work by evolving new solutions from old ones, and favouring the selection of those that perform well in terms of some fitness function. The concept is based on the Theory of Evolution. Although an EA can be used to quickly navigate a large subspace in order to get good solutions it is not guaranteed to find the optimal ones. In this thesis, one of the main goals is to see if it is possible to find a suitable solution for the crossing, without having to wait too long. Suitable here refers to a solution that, to some extent, satisfies the objectives put forth in the methodology chapter (section 3.2.1). *Too long* is the notion that any real time system is constrained by time limits in its domain. This makes EAs ideal for this application, as they can find good, not optimal, solutions pretty fast. Section 3.1.4 explains how the time constraint is handled in this work. In accordance with the research questions the solution should also incorporate some way of dealing with multiple objective functions. This section will explain the advantages of using an MOEA to control the intersection, as explained in chapter 3.

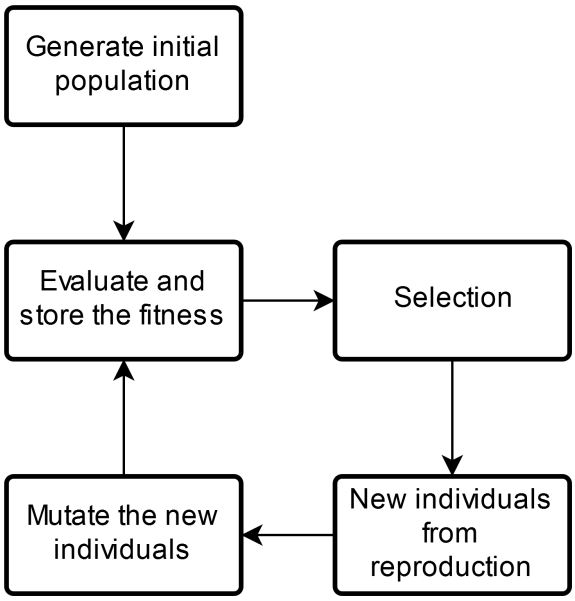
Generate initial population
Evaluate and store the fitness
Selection
Mutate the new individuals
New individuals from reproduction

Figure 2.4: The standard EA loop

EAs need to be able to maintain a population of solutions, and have a way to mate two solutions with each other. Individuals also need to be able to reproduce and create new instances through genetic inheritance. In addition, some variance is needed to evolve new traits[11].

Most EAs follow a standard loop [11], as shown in figure 2.4.

1. Generate an initial population. There are many ways of doing this, but a good starting point often ensures a diverse population.
2. Evaluate and store the fitness of each member of the population.
3. Select the members of the population that should be able to reproduce based on the individuals' fitness. A higher fitness should allow for a higher chance of reproduction.
4. Create new individuals based on the pairings from the last step. These should inherit their traits from their parents.
5. Introduce small variations in the new individuals (mutate) so that new traits may emerge.
6. Unless the stopping condition is met, repeat from step 2.

The stopping condition depends on the problem, and may be determined by some goal fitness or some time of execution. Unless the objective function evaluating fitness for each individual has no upper bound, the stopping criteria can be hard to determine for certain problems.

GAs [12] are a special case of EAs where the GAs perform the selection and mating procedure explained earlier by storing members of the population as genotypes. These genotypes can have a multitude of representations, but generally they are comprised of some vector of values.

2.3.1 Multiobjective Optimisation Problems

EAs, as explained earlier in this chapter, focus on optimising the solution for a single objective. It is often the case that the optimisation should consider several different objectives, creating a multiobjective optimisation problem(MOP). An MOP is a problem where multiple objective functions (denoted as the vector $F(x)$) should be minimised with regards to a vector x, where x is a solution (also known as a decision vector). In addition we can also have a set of constraints, here denoted as C. See equation 2.1 for a formal definition [26].

$$\begin{aligned} \underset{x}{\text{minimise}} \quad & F(x) = [f_1(x), \dots, f_i(x)] \\ \text{subject to} \quad & c \in C. \end{aligned} \tag{2.1}$$

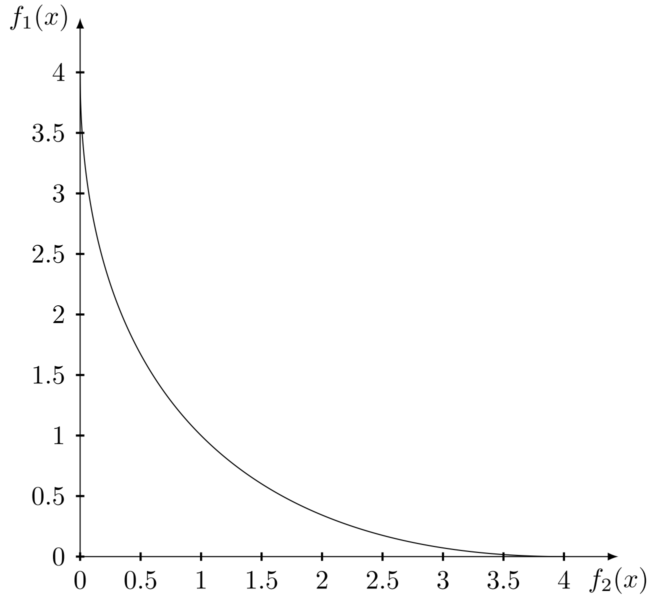

Figure 2.5: The Pareto-front of the problem described by minimising the functions of equation 2.2. Any solution below the line is considered invalid.

Objectives often conflict with each other, where if one objective improves it may repress another. The result of this is that there exists several solutions to an MOP, each called a

Pareto optimal solution. A solution is said to be Pareto optimal if there exists no other solution that dominates it, where domination is defined as follows [15]:
Given two vectors of objective scores of solutions a and b; a dominates b ($b \prec a$) if there is no objective in which b has a better score than a and a has a higher score than b in at least one objective.
The *Pareto-set* refers to the set of all Pareto optimal solutions. These solutions form a *Pareto-front* when projected in the objective space [26].

Consider the following example MOP (used in [5], originally from [16]). We wish to minimise the following functions $f_1(x)$ and $f_2(x)$ with regards to x:

$$\begin{aligned} f_1(x) &= x^2 \\ f_2(x) &= (x-2)^2 \end{aligned} \tag{2.2}$$

The variable bounds are set to $x \in [-10, 10]$. The Pareto-front (shown in figure 2.5) for this problem lies in the range $x \in [0, 2]$. Any value on the line is non-dominated, as no other value for both functions can improve upon one objective, while not impeding the other. All values above the line are dominated by some value on the line. All values under the line are invalid, as they are not possible to obtain from the problem.

2.3.2 Multiobjective Evolutionary Algorithms

When attempting to find solutions for an MOP it can seem reasonable to simply test out all possible combinations of variables, and measure the score of each objective. However, because many such problems (like finding an optimal speed for every vehicle in an intersection) have a rather large search space it is often not reasonable to do so. In this work especially, the time constraints for finding a solution are so strict that it is completely out of the question to simply run through all the alternatives. MOEAs, like regular EAs, present an efficient method of finding a set of good, if not optimal, solutions to a problem within a reasonable time frame.

The main difference between an MOEA and a regular EA is in the way that different solutions are compared. It is intuitive to simply select some method of linearly combining the objectives [11] with some weighting factor (shown in equation 2.3) resulting in a single score for each member of the population. However, this method does not consider the often non-linear combination of objectives, nor does it consider early dominance from solutions tuned to only one of the objectives. In order to improve upon this, several MOEAs [5, 11, 26] rank solutions based on position in terms of domination and distance from neighbours in the solution space.

$$f(x) = \sum_i w_i f_i(x) \tag{2.3}$$

MOEAs produce a whole set of valid solutions [11], not just a single best solution. Therefore it is important to have some way of selecting a good candidate from the set of possible solutions. This is normally done by some decision maker (DM) (often a human operator), so that good trade-offs can be evaluated. However, this work requires the whole procedure to be automatic, and will therefore require an autonomous DM. In a paper published in 2007 Ferreira et al. [9] propose a method of determining the single best solution after being given the set of non-dominated ones by comparing the solutions via a metric known as *stress*. In this thesis we implemented a very simple DM described in section 3.2.4.

In this thesis we will be relying on a genetic MOEA, namely NSGA-II [5]. Selecting a good genotype is important in this work. An example genotype is the one used by Yan et al. [25] to encode the sequence of vehicles passing the intersection. Their best results (ignoring computation time) came from using a binary encoding. Selecting a good representation, so that pairing two solutions via crossover can make a better individual, is important. Simplicity and computation speed are also important, however. We will therefore use a continuous encoding scheme where the values for each car are sorted in terms of when they spawned. Provided we use some sort of sequential crossover method, this increases the probability that vehicles likely to interact more will maintain their relationships after a crossover.

2.3.3 NSGA-II

There are many MOEA algorithms. NSGA-II was chosen because it is easy to understand, has few parameters, and is very well known. NSGA-II was introduced in 2002 by Deb et. al. [6]. When it was introduced one of its main features was the low complexity[1] of sorting solutions, but that is not a focus in this thesis. Further strides in the field of MOEAs have been made since then, but that is beyond the scope of this thesis. This section will describe the MOEA chosen, and highlight some of its relevant features.

The main loop of NSGA-II is similar to the one of regular EAs described in section 2.3. The generation of an initial population, and the scoring of each solutions fitness, is done on a problem by problem basis. In this work the initial population is random and the fitness consists of multiple objective scores (the objectives are defined in section 3.2.1). In addition, mutation and the mechanics of reproduction are problem specific. The methodology chapter (section 3.2) describes the approach used for those in this project.

The main difference between a regular GA and NSGA-II is the way it does parent selection and population selection. Both operations require two metrics to score each individual (i) in the population known as the *nondomination rank* (i_{rank}) and *crowding distance* ($i_{distance}$).

The i_{rank} of an individual is the Pareto-front that the individual belongs to, starting from 0. In other words, when looking at a population all the immediately non-dominated individuals have rank 0. When you remove those with rank 0 from the pool whatever solutions are then non-dominated have a rank of 1. This continues until no solutions are left in the population.

The crowing distance of a solution is calculated in order to favour solutions that lie further away from others in the search space, thereby making it more likely to find a more

[1]$O(MN^2)$, where N is the amount of solutions in a population and M is the amount of objectives

uniformly spread out Pareto-front. It is calculated by summing up the normalised distance in the search space to a solutions two closest neighbours of the same non-dominated front for each objective. The distance is infinite if no neighbour exists to one side. The normalised *distance* between two individuals (I_1, I_2) for one objective is defined in equation 2.4, where f is one of the objective functions and f_{max} and f_{min} are the respective minimum and maximum currently found of that objective.

$$distance(I_1, I_2, f) = \left| \frac{f(I_1) - f(I_2)}{f_{max} - f_{min}} \right| \tag{2.4}$$

When comparing two individuals for retainment in the population or parent selection (which is done by binary tournament selection) they are first ordered by the individuals' i_{rank}, ties are then broken by $i_{distance}$. NSGA-II has elitism (it always retains the best individuals), where retainment is done on a combination of both the parent generation and the new individuals.

2.4 Structured literature review

In order to familiarise ourselves with the field we started with an informal exploratory search on Google Scholar. This lead us to the works of Dresner et al. [7]. After reading their articles on FCFS and looking at their cited literature we familiarised ourselves with the field, and started a more formal literature review process.

The search engines used were Google Scholar and the IEEE Xplore search engine. The search terms, presented in table 2.1, were combined by combining the search terms in a group with the OR operator. Those groups were then combined with the AND operator. For example, a paper would be included in the search if it contained the terms ”Intersection”, ”Evolutionary algorithm”, ”Vehicle”, ”Autonomous”, ”Road” and ”Simulation”. No articles from before 2006 were included.

Table 2.1: The search terms used for the structured literature review. Each row represents a search term within each group.

Group 1	Group 2	Group 3	Group 4	Group 5	Group 6
Intersection	Evolutionary Algorithm	Car	Autonomous	Road	Simulation
Crossing	MOEA	Vehicle	Self-driven	Traffic	
Junction	Genetic Algorithm				
	Multi Objective				

After the search was completed the resulting articles were first filtered based on title. If the title seemed related to the project the papers were filtered on abstracts. The process resulted in very few articles. The articles were then read, again filtering out some. In addition, when an article referenced literature that seemed relevant, those were also considered regardless of their age. The process produced few related works; no projects were found to use the same independent time step modelling. More literature was sourced from informal searches.

2.5 Traffic Simulators

There are two applications for a simulator in this report. First, and most importantly, we need a simulator when evolving solutions with an EA. The objective function scoring each population will have to simulate the intersection. Then, collecting statistics such as distance travelled, stoppage time, vehicle locations and loss of kinetic energy, in order to give the population a score for further use in the traffic optimisation algorithm. Second, we will use a simulator when reviewing the solution presented for traffic optimisation. The results of this simulation will present the overall performance of our solution.

This section will discuss what features are important in our choice of simulator. Then we will discuss several existing simulators and explain the choices made for this project.

2.5.1 Features

By using a simulator that has already been made and tested, we have the advantage of not having to focus our work on the behaviour of autonomous vehicles in traffic. The simulator should be able to control the cars in a reasonable way, with the only input being the speed of the car. The low level properties of the cars, such as turning, acceleration and deceleration will be factors considered in the simulator.

Communication between vehicles and the infrastructure is important in this project. The simulator should provide the functionality to incorporate a central controller that can communicate with the cars in the intersection.

When calculating the score for the EA, we will need recorded data from the simulation. The data should contain information on a per vehicle basis and performance score for the whole intersection such as throughput.

Another feature we are looking for is modularity. We want to have the opportunity to incorporate our own code, modify the existing code and be able to review the code. This gives the added advantage that the simulator does not have to include all the needed features, instead we can implement them ourselves.

During testing of our system, we will need to use different setups for the simulation. Parameters such as size of the intersection(i.e. number of lanes), traffic density and speed limit will be taken as input before each run.

We are not evaluating the simulators based on runtime speed. There is a lot to learn from having a visualised representation of the intersection, where we can analyse the behaviour of the vehicles. There might be an alternative to this that is a lot faster, that in turn would make the MOEA a lot faster, but this is not the focus of our project.

2.5.2 Simulators

Several traffic simulators already exist [2, 3, 19]. These are being used to optimise traffic, predict traffic in the future and test optimisation strategies. These simulators are made to function for different environments and different areas of use. AIMSUN [2] and VISSIM [19] are hybrids of microscopic and macroscopic simulators. This means that they simulate larger areas than an intersection with lower details than microscopic simulators. For us this means that the simulator can not simulate each car in detail. CORSIM [3] is a microscopic traffic simulator, but the program is not made so that we can use our own code to control the simulation.

The main factors against the simulators mentioned so far are:

- No modularity. They are made as a complete package, and therefore it is difficult for us to add features.
- Complexity. Since they are made as a package, there are a lot of features we do not want to use. We predict that the effort it would take to learn these simulators would be better spent making our own.
- Licensing. These simulators are made as commercial alternatives to be sold to governments/other infrastructure managers, so that they can create efficient road maps, intersections and freeways.

The AIM project mentioned in section 2.2 provides the source code for the simulator they use in their projects. The features of this simulator are as follows:

- Low level details. The simulator focuses on one or more intersections with a high level of detail, i.e. it is a microscopic traffic simulator. Each car has its own rules making it possible to control each car individually.
- V2I communication. The project uses communication between cars and a central controller.
- Controlling vehicles. To solve the problem of controlling cars the simulator uses a path following system, as described in section 2.2. The assumption is that each car follows a predefined path, where the path is one lane into the intersection and another out. The system does not deal with collisions outside the intersection. This is reasonable, since the IM should only handle collisions inside the intersection. Cars colliding outside of the intersection are ignored in the simulation, as this is considered beyond the scope of the simulator [8].
- Input parameters. The parameters they use to initialise the simulation are: traffic level(i.e. 1000 vehicles per hour per lane), speed limit, stopping distance before intersection, number of east-bound/west-bound Roads, number of north-bound/south-bound Roads and the number of lanes per road. These parameters cover the size of

the intersection, the traffic density and speed limit as we mentioned in the section 2.5.1.

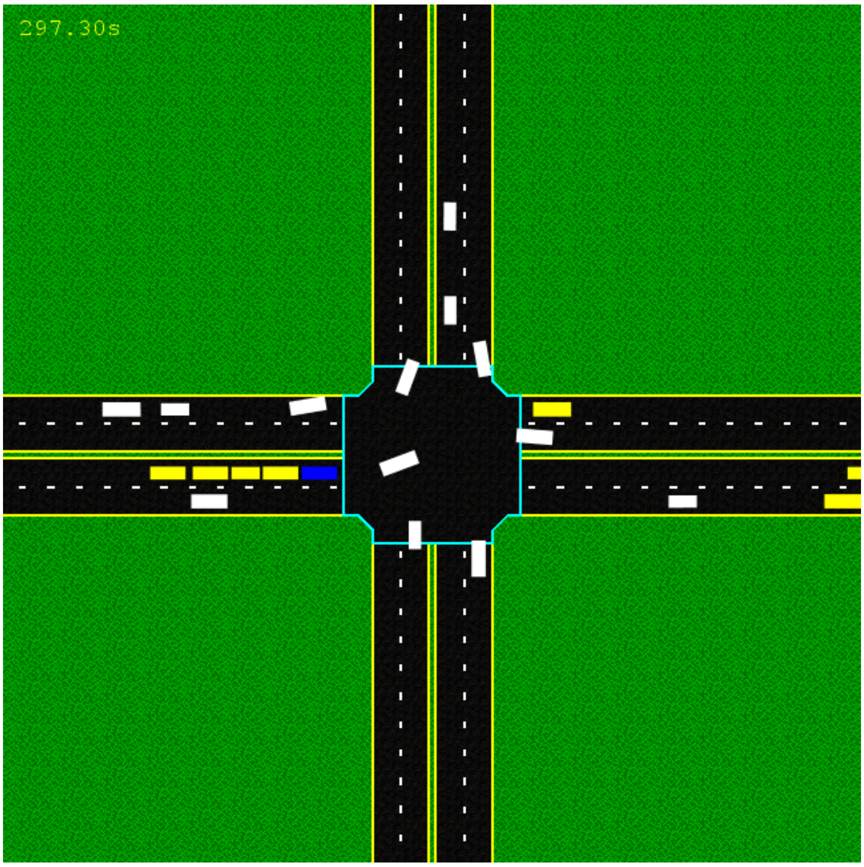

Figure 2.6: FCFS routing cars through the intersection, simulated on the simulator of the AIM project.

As explained in section 3.1.5 this work will modify the simulator created in the AIM project. Dresner et al. [8] introduced the concept of cars following predefined paths to deal with the low level control of vehicles (shown in figure 2.6), and the AIM project implemented an example of similar behaviour in their simulator. However, this had to be reimplemented for this thesis. The vehicles in the simulator follow predefined paths in order to relay their travel plans to the IM for scheduling, but they do not all follow the same paths. The paths of the vehicles are dependent on their speed and other parameters. In addition to this, the simulator also has the input parameters and the vehicle to infrastructure communications needed in this work. The simulator lacks the ability to calculate the data needed for our objective functions.

Chapter 3

Methodology

This section outlines the functionality of the system as a whole. The intersection manager explained in this chapter is a hybrid of two approaches described in the background chapter. It builds on the idea of predictable planned trajectories [8, 22], agreed upon between the IM and the vehicles. The IM then, in a similar fashion to other intersection optimisation methods [25, 22, 21], attempts to find a good solution for one state at a time. In order to translate the behaviour in each state into continuous behaviour, and to give the IM more explicit control of the vehicles in its control area, the IM is given direct control of the target speed of each vehicle.

This chapter introduces the IM that was implemented for this thesis. First, there is a section (3.1) describing the functional parts of the system. Second, a section (3.2) that describes the MOEA setup. Finally, section 3.3 explains the performance metrics that the system will be scored on in the experiments in chapter 4 before 3.4 sums up the experimental parameters.

3.1 System overview

The goal of the system is to control the behaviour of cars in an intersection by controlling the speed of each vehicle. The speed of each vehicle is optimised based on the objectives specified in section 3.2.1. In order to achieve this we have implemented a system with a central IM that communicates with each car. The system can be split into three main parts (shown in figure 3.1). The *IM*, the *Evolutionary Processor* (EP) and the *simulator*.

The IM's job is to read the current *state* of the physical intersection and pass that to the EP so that it can develop a *speed vector* $\mathbf{v}$. In the context of the EP each speed vector is an individual; a solution to the problem it is trying to solve. It is important to note that $\mathbf{v}$ is not a multi-dimensional vector specifying the speed of a single vehicle, but rather

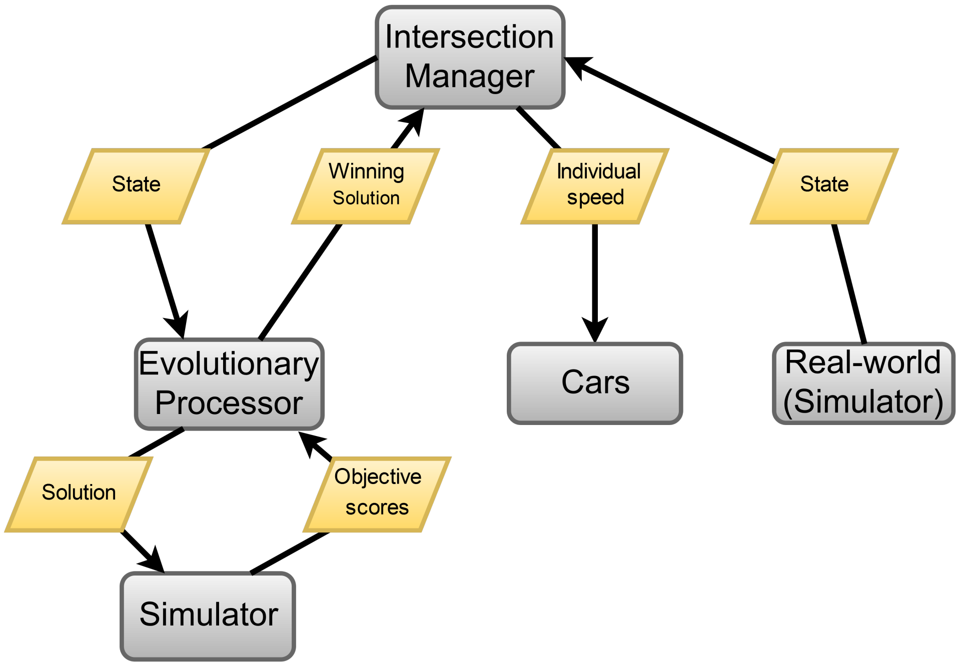

Figure 3.1: A complete overview of the system. Grey rounded squares signify entities. Yellow slanted rectangles are the information shared between them.

an n-dimensional vector describing the target speed of n vehicles (described in section 3.2.2). The vehicles travel along predefined trajectories, so their speed can be described with one variable. A *state* consists of every piece of information needed to recreate the current physical world, including vehicles and their current speeds, in a simulator.

The EP evaluates the individuals, using an MOEA described in section 3.2, by emulating them in multiple instances of the simulator. Once the stopping conditions of the MOEA are met (described in section 3.1.4) the DM (described in section 3.2.4) selects one of the available Pareto-optimal solutions, before the IM is returned a speed vector. The IM then sets the target speed of each car in the intersection in accordance with the solution. This determines how fast the cars should travel for the next time step.

We assume that the vehicles are reporting perfect positions and speeds to the IM. This means that the IM's internal representation of the intersection is a perfect representation of the real world. Then, as long as the simulator used by the EP is able to perfectly simulate the real world, the resulting speed vector should produce a result identical to the one that was simulated in the EP. Because we are using the same deterministic simulator for representing the real world and evaluating the solutions, perfect reproduction between the modelled world and the real one can be guaranteed.

3.1.1 The Intersection Manager

The IM is responsible for controlling the speed of the cars in the intersection. The *control area* of the IM is defined as the whole intersection, including the *spawning zones* (see 3.1.3). The IM does not control the areas outside the intersection, nor does the EP consider what happens there when evaluating objectives. The IM will only take new vehicles into account when calculating a new solution. The vehicles are themselves responsible for collision avoidance before they are under the control of the IM. Collision avoidance before IM control is considered beyond the scope of this work. Therefore, such collisions are avoided by not allowing two cars to occupy each spawn zone at the same time. The speeds of spawning cars are also scaled so that the IM will have time to stop them before they leave the spawn zone. The spawn zones can be seen in figure 3.2.

Communication between the IM and the vehicles is modelled as a two way system similar to the V2I protocol explained in 2.1.1. In order to find a suitable speed vector the IM gets all the relevant vehicle information from each vehicle. Once one has been found it transmits back a target speed as according to the system specifications. The model does not consider loss of data or transmission delays.

In order to remain consistent between the experiments the intersection itself never changes. There are two lanes going in each direction, the same trajectories are always used, and we never change the dimensions of the lanes or the cars. There is a global speed limit that all cars must adhere to. This is always set to $25\mathrm{m/s}$, unless it is lowered to aid in the creation of scenarios. Scenario creation is explained in section 3.1.3.

When managing the intersection the IM is concerned with two time step sizes. We call these t_{main} and $t_{evalulate}$. t_{main} is the lifetime of one solution in the real world; once the EP has decided on a solution, that solution is used for the next t_{main} seconds, before the EP is ordered to come up with a new solution. However, if the EP had only validated the solution for $t + t_{main}$ (where t is the current time) there are no guarantees that the intersection does not end up in an *unsolvable state* [1]. In order to guarantee that the IM never approves a solution that is unsolvable the EP is also asked to evaluate the solutions for an extra amount of time. This time is represented by the variable t_{sim}. The total amount of time that the EP evaluates each solution is then given by:

$$t_{evaluate} = t_{main} + t_{sim} \tag{3.1}$$

Figure 3.3 shows the function of these time steps visually.

Because the state of an intersection can be incredibly complex, there is no simple way to calculate the minimum value for t_{sim} to keep the system safe. Different values for the time parameters are explored in experiment section 4.1.

3.1.2 The vehicles

Before entering the intersection each car will decide on a path. The choice of path is determined via a look-up table where cars say what lane they are in, and what lane they wish to end up in. For every incoming lane there are two paths that end in either a turn or going straight. This is shown in figure 3.4. After deciding on the path to follow the car is not allowed to switch lanes or diverge from the path in any way. This path-following behaviour ensures that the IM only has to regulate the speed of the vehicles. It also ensures that the behaviour of the cars is predictable. This is important to ensure that the behaviour of a combination of speed vector and state are consistent across multiple instances of the simulator. In order to avoid the need for lane-changing, and to ensure that no vehicle must deviate from its trajectory to reach its intended goal, they spawn in the path that terminates at the vehicles goal. In a real world application this would entail that the vehicles need to change lanes before reaching the control area of the IM.

[1] *Unsolvable state* means a state where a collision is unavoidable for the next t_{main}.

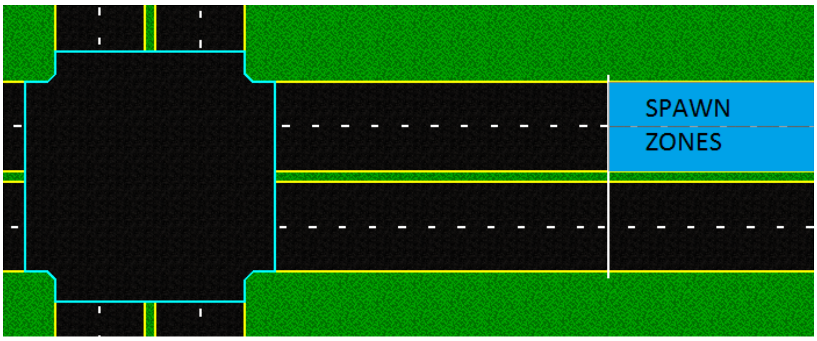

Figure 3.2: The eastern spawn zones. Each lane has its own. They are not to scale, and are much smaller and further away from the intersection in the actual simulator.

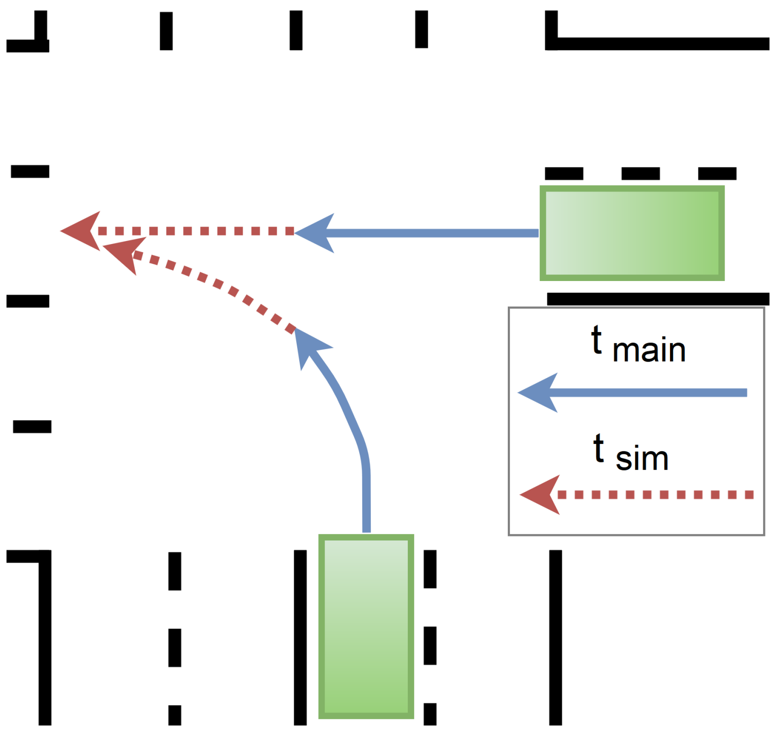

Figure 3.3: A visual representation of the difference between t_{sim} and t_{main}. $t_{evaluate}$ is the time it takes for the cars to travel along both the solid blue and the dotted red lines.

When a car enters the intersection it will receive a message from the controller containing the speed the car should maintain. The first message is received once the IM is ready; at the next start of a t_{main}. Upon receiving a message the car will either accelerate or decelerate until it has reached the provided speed. It may receive a new message before it has reached this speed; the car should then change its goal speed to the new one. Vehicles will accelerate or decelerate as fast as possible to reach their target speed.

Table 3.1: The specifics of the vehicle classes. These values are the same as used by Dresner et al. [8].

Vehicle type	**Acceleration** (m/s^2)	**Deceleration** (m/s^2)
Van	3.08	-30.0
Sedan	3.25	-39.0
Coupe	4.50	-45.0
SUV	3.83	-39.0

In order to better simulate real-world conditions there are four types of vehicles: coupes, vans, SUVs and sedans. They have slightly different sizes and acceleration/deceleration

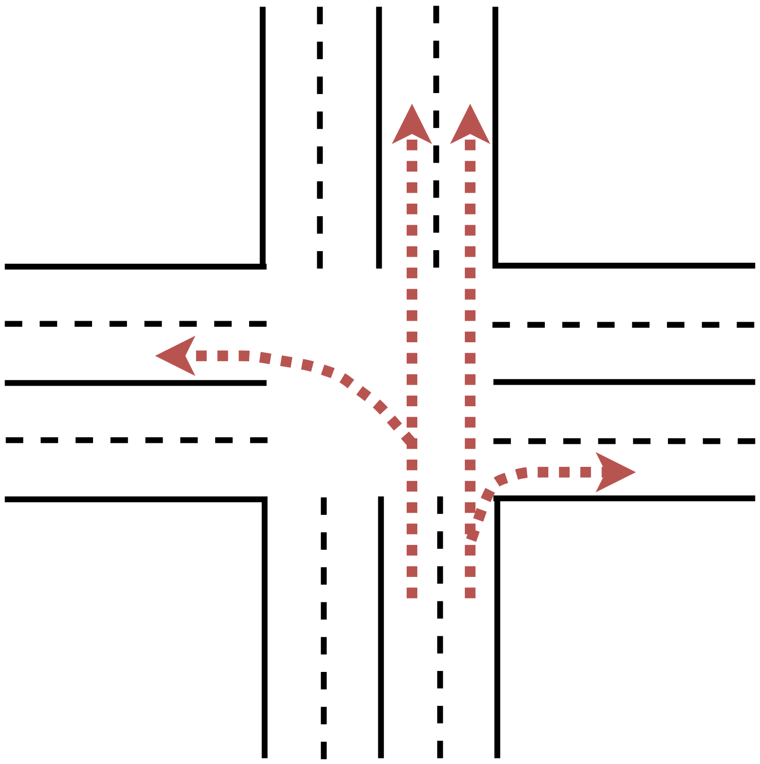

Figure 3.4: The possible paths of travel when spawning in the left or right southern spawn points. This pattern is repeated for all four incoming directions.

speeds. The differences in acceleration can be seen in table 3.1. Every type of vehicle has the same mass.

3.1.3 Spawning vehicles

Spawning vehicles is done by *spawn zones*. Vehicles may only spawn in the simulator that represents the real world. The EP assumes that no new vehicles need to be controlled while a solution is deployed. The rate at which vehicles spawn is determined by the simulator parameter known as *spawn rate*. In order for a vehicle to spawn two conditions have to be met:

- The spawn zone must want to spawn a vehicle. How often this happens is determined by the *spawn rate* parameter.
- The spawn zone must not be occupied by another car. This is to make sure that no collisions occur inside the spawn zones before the IM has control of the cars.

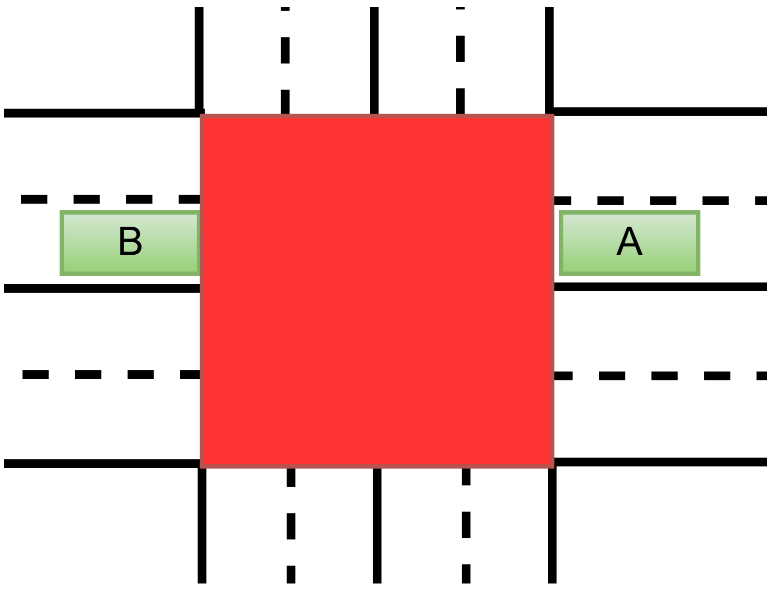

Figure 3.5: The inner intersection is displayed in red. Car B is counted as having evacuated the inner intersection. Car A has yet to enter it.

The spawn rate is a number between 0 and 1 that represents the probability that the *spawn zone* is going to spawn a vehicle in one second in the real world simulator. This means that there is no direct way to set the amount of vehicles arriving in $\frac{vehicles}{hour}$. The initial speed given to a vehicle as it spawns is dependent on the amount of time left of the current time step. The IM can only take control of a new vehicle when a new time step starts. Therefore, to avoid cars spawning into unsolvable states, the speed of the new vehicle is set so that it is possible to stop it before it exits the spawn zone. This makes the $\frac{vehicles}{hour}$

dependent on the size of t_{main}. A measured relationship where collisions are ignored, and cars evacuated as fast as possible is shown in table 3.2 for $t_{main} = 1.5s$. This table also includes a measure of $\frac{vehicles}{300s}$, which would be the expected throughput in section 4.2 if all vehicles could pass through the intersection while maintaining the speed limit.

Table 3.2: A measured relationship between the spawn rate, $\frac{vehicles}{hour}$ and $\frac{vehicles}{300s}$. It was generated by running the simulator for one hour with each spawn rate, ignoring collisions and evacuating cars from the spawn zones and intersection as fast as possible. $\frac{vehicles}{300s}$ was included to allow comparison with the results in the experiments.

Spawn rate	$\frac{\textbf{vehicles}}{\textbf{hour}}$	$\frac{\textbf{vehicles}}{\textbf{300s}}$
0.1	2972	247.67
0.2	5529	460.75
0.3	7688	640.67
0.4	9478	789.83
0.5	11234	936.17
0.6	12746	1062.17
0.7	14233	1186.08
0.8	15570	1297.50
0.9	17064	1422.00
1.0	21822	1818.50

In order to compare different configurations of the system there is a need for reproducible scenarios in the simulator. We have solved this in two different ways, depending on the experiment and its goals. The first way is the one previously described in this section. We simply re-run the experiment with different parameters and the same spawn rates for a long time. However, when more accurate reproducibility is needed, as in section 4.1, we have precomputed *scenarios*. These scenarios are created by running the simulator for a given amount of time, with known good values, until a set amount of cars have been spawned into the system. The scenarios are then manually inspected to make sure that no unsolvable state is present. For instance, when wanting to test with a high amount of cars the system is set to spawn cars until 40 cars are present, before leaving the inner intersection (as shown in figure 3.5). A sample scenario can be seen in figure 4.1. If it is observed that we are unable to create scenarios with a certain amount of cars because the cars are successfully evacuated too fast we lower the speed limit in order to evacuate cars more slowly. This has the side effect that no cars in the generated scenario will start with a speed that is faster than the speed limit used. The IM used in the experiments is free to accelerate the cars up to its own speed limit.

3.1.4 Time modelling

Some simplifications have been made when modelling real time in the system. The need for such simplifications arose because outside variables, such as hardware and computing time, were not considered. The first simplification is that we consider the calculations done in the controller, by the EP, as instant. This means that the simulation of the real world must stop and wait for the EP to calculate the speed for each vehicle. This can be justified, and relies on the assumption that the system has a way of evaluating what the state of the intersection will be at a given time in the future ($t_{future} = t_{current} + t_{main}$). The time it takes the IM to evaluate the future state, here given as $t_{compute}$, also has to be significantly smaller than t_{main}. If the time of the intersection is currently $t_{current}$ and the controller is currently employing the solution obtained for $t_{current}$, it can, using the simulator, obtain the state that the intersection will be in at t_{future}. Because the state at t_{future} is known the IM may start processing a solution for t_{future} at $t_{current}$. This shows that there exists an amount of time ($t_{computation}$) given by:

$$t_{computation} = t_{future} - (t_{current} + t_{statecalculation}) = t_{main} - t_{statecalculation} \quad (3.2)$$

, where $t_{statecalculation}$ is the time it takes the IM to calculate the next state. As long as $t_{statecalculation} < t_{main}$ it follows from equation 3.2 that $t_{computation} > 0$. Therefore there exists some amount of solution evaluations that the EP has time to perform. The actual amount of evaluations given realistic current hardware is beyond the scope of this work.

We use a parameter (e_v) to decide the computation time for the EP measured in the amount of solutions evaluated. This provides the benefit of our controller not being dependent on the hardware it runs on. It should however be noted, as evident in section 4.3, that the actual evaluation time of a solution depends on the amount of cars present in the intersection. In order to keep the amount of "time" given to the EP consistent across experiments a global amount of evaluations per second ($\frac{e}{second}$) is set for each experiment. The amount of evaluations per t_{main} of real time is then given by

$$e_v = \frac{e}{second} * t_{main} \quad (3.3)$$

3.1.5 The Simulator

There are two main applications for a simulator. First, and most importantly, for the EP to work we need a simulator when evolving solutions with an EA. The objective functions scoring each individual rely on simulations of the intersection. Then, statistics are collected (see 3.2.1) such as distance travelled, throughput of inner intersection, vehicle wait-time and total kinetic energy lost, in order to give the individuals in the population a score for further use in the traffic optimisation algorithm. Second, the simulator is used to simulate the real world. In a real-world implementation of the system the simulator would only be used in the EP.

The simulator is a modified version of the AIM4 Simulator v1.0.3 (the same as used by [7]). By adapting a simulator that has already been made and tested, we have the advantage of not having to focus our work on the behaviour of autonomous vehicles in traffic.

The AIM4 simulator was chosen because it has support for implementing custom vehicle behaviour. This allowed the implementation of trajectory-following behaviour, described in 3.1.2. In addition, the simulator was written in Java, which made it possible to combine with the chosen MOEA framework, as introduced in 3.2. The AIM4 simulator was available with source code, making it possible to investigate un-documented behaviour and to modify. See Section 2.5.2 for more details about the choice of simulator. Unfortunately the APIs provided by the simulator did not include much of the functionality needed for this work, so large parts of it (besides the graphical presentation) had to be rebuilt. This was also necessary to improve the speed of the simulator, as that was a much more important feature in this project than the one for which the simulator was originally intended. Any references to the *simulator* outside of this subsection refers to the one implemented for this work.

The simulator is used to evaluate speed vectors for the EP in the following way. First, it is given a state s_0 to spawn from. Second, the simulator is given the speed vector it is to evaluate. The simulator then relays the speed information to the vehicles and simulates for a given time ($t_{evaluate}$) before returning the relevant metrics for the objective functions. Because the simulator has to be able to generate objective scores for a lot of different speed vectors it was important to use a completely deterministic simulator. One guarantee of the simulator is therefore that it will always end up in the exact same state when provided with the same starting state and speed vector.

3.1.6 Observing the IM

Running the IM implemented in this thesis takes a long time. For example, one run with the parameters used in section 4.2 and a spawn rate of 1.0 took 35 hours and 3 minutes[2] to complete. A spawn rate of 1.0 is the worst case, as the simulator runs faster with less vehicles. The speed of the real world simulation was too slow to manually observe. Therefore, in order to manually observe the behaviour of the IM, there was a need to create a system to record and replay the IM's behaviour. This was done by storing each state between time steps, and the speed vector found by the IM for that time step. After the completion of an experiment, a simulator could then be instantiated from the states, and the chosen solution to each state would be immediately available. Thus allowing for continuous playback.

In addition, the graphical interface of the simulator was extended so that cars slowing down were coloured red, and vehicles speeding up were coloured green (figure 4.1 shows the simulator while active). This allowed us to make observations as to what vehicles were being prioritised for traversal in which scenarios. Collisions were marked by colouring the vehicles blue.

[2]On an intel i7 4770. Several computers were used to gather the data needed in this thesis.

3.2 MOEA details

We used The MOEA Framework [4] to implement the MOEA. This framework provides several MOEAs. We will use NSGA-II (described in section 2.3.3). All objective functions need to be minimisation objectives in this framework. Therefore, we will negate any optimisation objectives to convert them to minimisation objectives.

The MOEA tries to optimise the speed of each car based on a given set of objective functions. Given a population P of the MOEA and n cars each having a speed v, the vector for the population will be represented as:

$$P = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n] \tag{3.4}$$

where $\mathbf{v}_i$ denotes a speed vector, as discussed previously in this chapter.

The elitism in NSGA-II is important. It ensures that a collision avoiding solution is available from the population when the stopping conditions are met, as long as the MOEA was, at some point, able to find one.

3.2.1 Objectives

The objective function scores are calculated by simulating the behaviour of the cars in the simulator and gathering data for the given $t_{evaluate}$.

The optimisation goals for the MOEA are to maximise throughput in the inner intersection, maximise the distance travelled for each car, minimise total stoppage time and minimise total kinetic energy lost. The definitions of these are presented below. It is important to note that while these objectives are important in terms of performance, they are not the final metrics by which the system is measured. For performance metrics see section 3.3.

In order to choose one solution based on a weighted average (the method for choosing a solution is described in section 3.2.4) every objective value is scaled between 0 and 1. This is done by finding the maximum value for the given objective, while ignoring constraints, and dividing the objective score by the maximum value. Maximum objective scores for distance travelled and throughput in the inner intersection are found by running a simulation that ignores collisions with every vehicle speed set to the speed limit, then reading the objective scores from that simulation. The maximum values for total kinetic energy wasted and total stoppage time are found by running the simulator with all target speeds set to 0.

Maximise distance travelled

In order to make the vehicles move as fast through the intersection as possible they should move as far as possible in every time step. Therefore the goal of this objective is to maximise the total cumulative distance travelled for all vehicles, represented as minimising the negated distance travelled D. The concrete objective is as follows:

$$\text{minimise} \quad D = -\sum_{v \in V} distanceTravelled(v) \tag{3.5}$$

where V is the set of all vehicles and $distanceTravelled(v)$ is the distance travelled by v in the last $t_{evaluate}$.

Maximise throughput in inner intersection

This objective is also referred to as tiny throughput. Throughput is defined as the number of cars exiting the *inner intersection* during a simulation. The *inner intersection* is defined as the area not made up by the lanes (shown in figure 3.5). Maximising this objective is important to get as many cars as possible through the intersection, thereby avoiding congestion and deadlocks. The objective is defined as:

$$\text{minimise} \quad T = -n \tag{3.6}$$

where n is the amount of cars that have vacated the inner intersection and T is the negated total throughput.

Minimise total stoppage time

This objective is also referred to as the starvation objective. The goal of this objective is to minimise starvation. Starvation meaning the effect where a single car is halted for a prolonged amount of time in order to let other cars pass. Because every time step is considered individually as single problems an objective was set up where the total stoppage time of each car that was also stopped in the considered time step are added together. The equation below explains how the objective S is calculated.

$$\text{minimise} \quad S = \sum_{s \in V_{stopped}} s \tag{3.7}$$

where s is the total amount of time each vehicle has been stopped and $V_{stopped}$ is the set of stoppage times of all cars that have not moved in the considered $t_{evaluate}$. This does not violate the notion that the time steps are independent. The vehicles carry the information about how long they have been physically stopped, and only report a single value to the IM at each time step. The IM itself does not have to remember anything.

Minimise total kinetic energy lost

This objective seeks to minimise the total amount of energy wasted, which is also one of the global metrics that the system is scored on.

$$\text{minimise} \quad E = \sum_{i=1}^{N} E_i \tag{3.8}$$

where E_i is the kinetic energy lost in each car. Because we only want to look at wasted kinetic energy E_i works out to the following:

$$E_i = \max\left(\frac{1}{2}m(v_{\text{end}}^2 - v_{\text{start}}^2), 0\right) \tag{3.9}$$

We know that the car will either decelerate or accelerate in order to match a given speed in a simulation. This allows us to avoid looking at smaller time steps. Here, v_{end} is the speed of the vehicle at the end of the evaluation and v_{start} is the speed of the vehicle at the beginning of the evaluation.

3.2.2 Genetic encoding

The speed of each vehicle in the intersection is represented by a real number (s_i). The genotype is made up of an array of these numbers. This is equivalent to the speed vector previously mentioned in this chapter.

$$\mathbf{v} = [s_1, s_2, \ldots, s_n] \quad s_i \in [0, \text{SpeedLimit}] \tag{3.10}$$

where **v** represents the speed vector and s_i represents the speed for vehicle i.

The genotype is ordered so that new cars are appended on the right side; car $i+1$ spawned right after car i. Based on the assumption that vehicles that spawn close to each other in time are more likely to arrive at the inner intersection at similar times, this ensures that cars that are likely to interact will be close in the genotype. This is beneficial as it allows the evolutionary algorithm to solve problems occurring between a subset of cars, and then keeping those solutions after crossover.

3.2.3 Variation

When individuals are picked for reproduction they are paired with another parent. The outcome of the reproduction depends on two probabilities. The first being the crossover rate($P_{crossover}$) and the second being the mutation rate($P_{mutation}$). Given that a crossover should be applied, the tail of the parents genome are switched from a random index. This results in two new individuals based on both the parents. After crossover, the children might mutate. Each s_i in the new individual has the probability $P_{mutation}$ of mutating. When a mutation occurs, a new speed is picked based on a random draw. The random draw works as follows: A random number between -1 and the speed limit + 1 is generated. If the number is below 0, it is set to 0. If the number is above the speed limit, it is set to the speed limit. The reason we have these margins at each end, is to make sure there is a nonzero probability of hitting both 0 and the speed limit.

For this work, the parameters have been set to:

- $P_{crossover} = 0.9$
- $P_{mutation} = 0.1$

3.2.4 Choosing a solution

When the MOEA returns a Pareto-set, i.e. a population of several non-dominated solutions, the problem of picking the preferred individual have to be solved in order to automate the system. In order to achieve this, we utilise a simple DM. A complex DM was avoided in order to make it easier to understand the behaviour of the system as a whole.

The non-dominated population is sorted based on two factors. Since the MOEA may not have enough evaluations to find a solution where there are no collisions, the population is first sorted on the number of collisions. Then, if the number of collisions are equal, each remaining objective score is multiplied by their corresponding weight (α) and summed up. This value is the secondary sorting option. In a run where there are no collisions, this function prioritises the objectives based on the weights provided to the IM. As mentioned in section 3.2.1 the objective scores are normalised based on the worst or best case score of that objective.

3.3 Measuring performance

This section describes how the performance of the system is measured. All measurements are taken from the real world simulator.

3.3.1 Metrics

The performance of the system is measured based on the following four factors. Some of these factors are similar to the optimisation objectives, but are scored over longer periods of time.

- Throughput: The amount of cars that pass through the intersection during a simulation.
- Mean evacuation time (MET): The MET of all the cars. Evacuation time here means the time it takes the car to leave the IM controlled zone completely. By running a simulation ignoring collisions with the maximum speed set on each vehicle, the optimal value for MET was determined to be $13.12s$.
- Total kinetic energy lost. The same as the optimisation objective, just summed over the whole time period.
- Amount of collisions. The amount of collisions that occurred during the simulation. There is no distinction between types of collisions.

3.3.2 Collisions

When cars collide, it is impossible to keep running the simulation and keep recording valid metrics. Three ways of dealing with collisions were considered. First, an approach similar to [7] where the collided cars would simply stop, and it would be up to the rest of the vehicles to route around them, was considered. This was not a fitting solution, as one of the premises of the proposed IM is that vehicles move along predefined predictable paths. If cars are to be stopped in the middle of the intersection, and cars may not route around them, the rest of the simulation would simply break down and the only usable result would be the time it takes before a collision occurs.

Secondly, it was considered to simply remove cars that collide. Removing cars is positive in the sense that they no longer count towards the measured MET, and can be ignored for calculations of total kinetic energy lost. However, the resulting performance metrics would be skewed very favourably when problematic cars are simply allowed to vanish.

Finally, it was decided to handle cars that have collided as still being active cars in the intersection, but ignore the physics between cars that have collided. The cars are allowed to pass through each other in order to clear them from the intersection. The EP will ignore the collision occurring between the colliding pair for the following time steps, but still count other cars colliding with the ones that are collided. This approach was used because the goal is to measure the number of collisions that occur, and not the EPs ability to separate cars that already occupy the same space. The performance scores are still affected by this approach, but they should only be affected in the positive sense. Therefore, observing both a rise in MET and collisions leaves the increase in MET open to other interpretations.

3.4 Selecting the parameters

Table 3.3 gives a summary of the parameters used in the experiments. The system was tested multiple times, with the spawn rate set to 0.5, t_{main} and t_{sim} to 1.5s and given 25000 evaluations, with different values for $P_{mutation}$, $P_{crossover}$, population size and the objective weightings in order to find values that worked well. The values represented in table 3.3 were selected as they were found not to cause collisions, and produce acceptable values for MET and total kinetic energy lost.

The amount of $\frac{evaluations}{second}$ was set based on two criteria. First, like for other parameters the IM performed reasonably well in terms of performance metrics for a medium amount of cars. Secondly, running the IM takes a long time. Therefore, the amount of evaluations had to be limited to a reasonable amount in terms of computation time to make the amount of experiments in chapter 4 feasible.

Table 3.3: A summary of the parameters used in the experiments.

Parameter	**Value**	**Notes**
$P_{mutation}$	0.1	The probability that the speed of one vehicle is mutated.
$P_{crossover}$	0.9	The probability of crossover.
Population size	100	The population size used for the MOEA.
t_{main}	1.5s	The amount of time the IM deploys each solution. Different values are explored in section 4.1.
t_{sim}	1.5s	The amount of time the EP evaluates a solution beyond t_{main}. Different values are explored in section 4.1.
Speed limit	$25\mathrm{m/s}$	The speed limit in the intersection.
Vehicle mass	1000kg	The mass of each vehicle. This is the same for each vehicle. Used when calculating kinetic energy.
$\frac{evaluations}{second}$	50 000	The amount of evaluations available per second of t_{main}.
Objective weighting	0.25	The importance of each objective. This work used same weight for all objectives.

Chapter 4

Results and Discussion

This chapter will present the experiments conducted in this thesis. Each experiment follows the methodology presented in chapter 3, and is aimed to give insight into the research questions introduced in chapter 1. The results will be discussed for each experiment.

Section 4.1 explores the use of different time step sizes and finds reasonable time step values. These values are then used to examine the effects of independent time step modelling over continuous time in section 4.2. In order to understand the effects of the chosen optimisation objectives section 4.3 explores the behaviour of the IM when only optimising for different subsets of said objectives. Section 4.4 and section 4.5 explore the effects and behaviour of the objectives. Finally, section 4.6 sums up and discusses the overall observations in relation to the goal and research questions presented in the introductory chapter (chapter 1).

4.1 Experiment 1: Looking at time steps

Goal

This experiment aimed to give more information about how solving independent time segments affects the overall behaviour of the system. As stated in research question 1, we have investigated the effects of dividing the intersection management problem into smaller time steps. This section looks at the effects of varying t_{main} and t_{sim}. A thorough explanation of these variables is available in section 3.1.1.

Methodology

As explained in chapter 3, we have found solutions for individual time segments with the MOEA. By using the results of these smaller time segments when routing vehicles through the intersection we have looked at the overall behaviour of the IM. In this experiment we looked at how the performance metrics are affected by varying the size of t_{sim} and t_{main}, with focus on the number of collisions in the intersection. This experiment also provided results showing how the chosen values for t_{main} and t_{sim} performs in regards to the performance metrics described in section 3.3.1.

The number of evaluations in the MOEA for each second(e_v, described in section 3.1.4) was constant for all the experiments. This value was set to 50 000 evaluations and the population size to 100, i.e. the MOEA evolves 500 generations with 100 individuals per second of t_{main}. Since t_{main} varies for each experiment, we have calculated the evaluations per time step based on Equation 3.3.

The two parts of this experiments differed in which of the time variables that were investigated. The first part focused on t_{main} and the second used the results of part 1 to investigate different values of t_{sim}.

The intersection states in these experiments were the same for each of the time steps. We have created 15 different scenarios (scenario creation is explained in section 3.1.3) populated with different amounts of cars. 5 with low traffic (10-11 cars), 5 with medium traffic (20-22 cars) and 5 with high traffic (40-43 cars). Three example scenarios are presented in appendix A.

No cars were spawned during the simulation, to make the different runs as similar as possible, ensuring a fair comparison of time step values. An example starting state is available in figure 4.1. The MET values reported in this section were sometimes below the stated minimum of $13.12s$. This is because the MET here was measured from the start of the scenario. This means that some vehicles have already traversed a significant portion of the intersection when their MET counter starts. MET can therefore here only be used to measure differences between time step values, and not as an indicator of overall performance. Section 4.2 explores overall performance.

Part 1

Goal

The intention of this experiment was to find in which range t_{main} gives the best performance with regards to the performance metrics defined in 3.3.1, with the main focus on minimising collisions in the intersection. This experiment should also find a reasonable t_{main} that could be used in the later experiments presented in this chapter. Finding a reasonable t_{main} is important because it allows for less uncertainty when determining reasons for different observed behaviours. Knowing how the IM behaves with different values for t_{main} would also give valuable insight towards research question 1.

Hypothesis

We expected that the lower time steps would give better control over the vehicles, and therefore less collisions. MET is expected to follow the same behaviour, where a lower t_{main} equals a lower MET. Since the IM will have more opportunities to

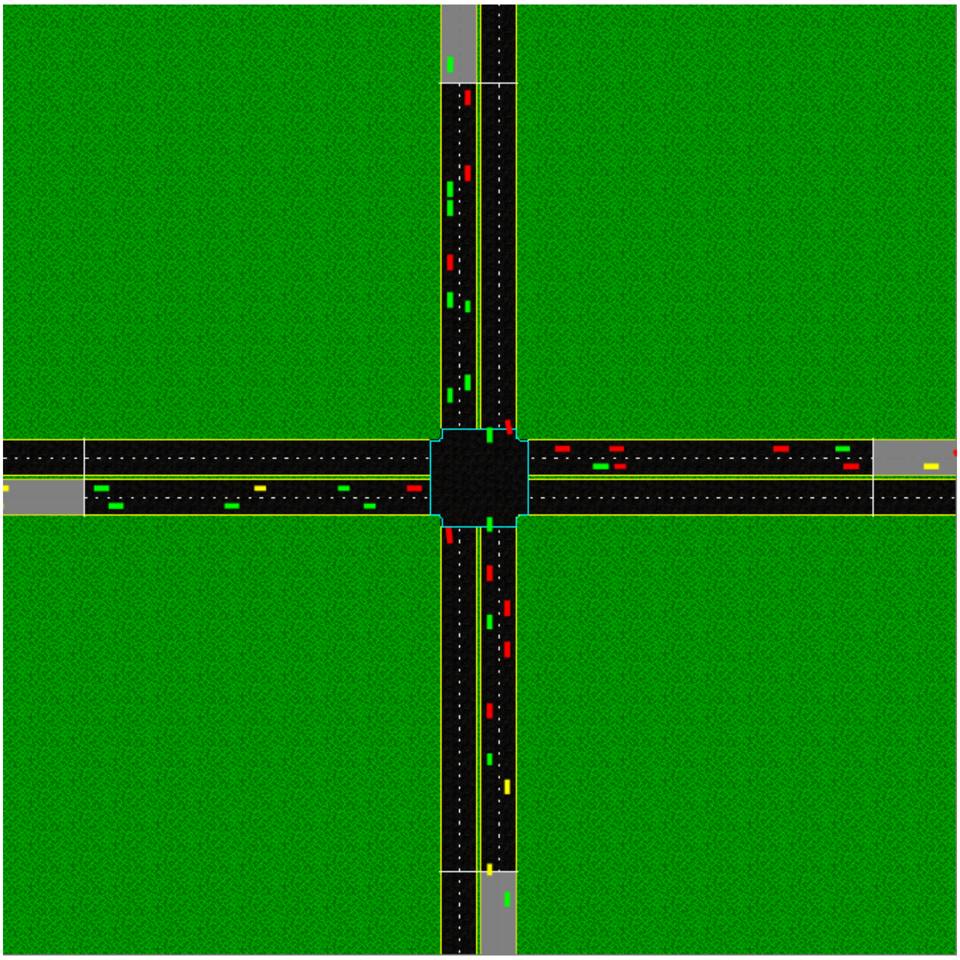

Figure 4.1: A sample starting point for a simulation in experiment 1.

change the speed of each vehicle, we expected that each vehicle will use more energy to increase the overall score of the chosen individual solution at the expense of total kinetic energy lost.

Given that we have set an upper boundary for the evaluations, we expected to find that some values of t_{main}, especially the smaller ones, did not get enough evaluations to avoid collisions.

Methodology

In this experiment we tested different values for t_{main}. $t_{evaluate}$ was calculated from equation 4.1. The specific values for this experiment are showed in table 4.1. The results were generated by running the IM on each scenario 5 times (a total of 75 runs per t_{main}), and computing an average.

$$t_{evaluate} = t_{main} + 2.0 \tag{4.1}$$

As can be seen from equation 4.1 and table 4.1 t_{sim} is here set to 2.0 seconds. Some preliminary testing indicated that a t_{sim} value of 2.0s gives the internal simulator enough of a margin to avoid collisions. As this experiment did not focus on the value of $t_{evaluate}$ and t_{sim} we wanted to make sure the value set did not interfere with the IMs ability to avoid collisions.

Table 4.1: The time step values, in seconds, used in Experiment 1 Part 1. Each column represents the values used for one configuration.

t_{main}	0.1	0.2	0.5	1.0	1.5	2.0	5.0	10.0	15.0
t_{sim}	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
$t_{evaluate}$	2.1	2.2	2.5	3.0	3.5	4.0	7.0	12.0	17.0

Results

The results are displayed in three graphs. One for the total number of collisions and one for each of the performance metrics total kinetic energy lost and MET, respectively. The graphs display the average of each of these three values per time step. Considering the number of runs performed, the average value for each time step is based on 75 scenarios. The results for throughput are not displayed, as every vehicle was evacuated in all scenarios.

As can be seen in figure 4.3, the average amount of collisions dropped around 0.5s, stayed low until 2.0s and then rose slightly. The difference between averages when $t_{main} > 2.0$s are not likely to be significant. However, there was a clear trend of collisions for small values of t_{main}.

Manually inspecting a high traffic scenario with collisions for $t_{main} = 0.1s$, showed that the IM constantly changed what vehicles it prioritised for intersection traversal. This lead to a deadlock, shown in figure 4.2. This deadlock resulted in a collision.

The MET was at its lowest when t_{main} was 2.0s, as can be seen in figure 4.4. However, the difference was negligible from that of 1.5s and 1.0s. The MET increased as t_{main} increased past 5.0s.

Figure 4.5 shows how varying t_{main} affected the total kinetic energy lost. The total kinetic energy lost rose when the time step size decreased, this is again most likely because the IM has more control in lower time steps. The more often the IM can update the speed for each vehicle, the higher the chance of braking becomes. It is interesting to note that at around time step size 1.5s there was a change in the difference in total kinetic energy lost between time steps. Given a time step size of 1.5s, the gain from using a larger time step size was a lot less for total kinetic energy lost compared to below 1.5s. Because the gain in terms of energy was low when increasing t_{main} further beyond 1.5s, this value is seen as a good compromise in relation to other objectives.

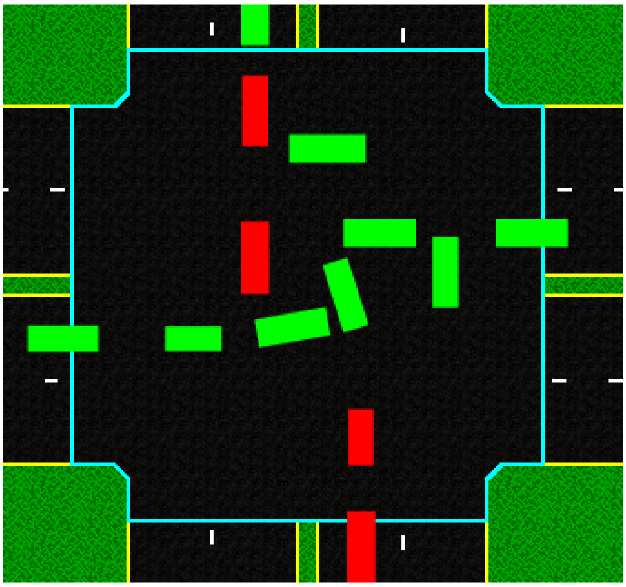

Figure 4.2: A deadlock from running a high-traffic scenario with $t_{main} = 0.1$. The vehicles can not move out of the inner intersection without passing through each other.

Because all vehicles being evacuated was used as the stopping condition for these experiments all runs produced the maximum throughput.

Part 2

Goal

The goal of this experiment was to optimise the value of t_{sim} with regards to the performance metrics. In addition, the results should give insight to which combinations

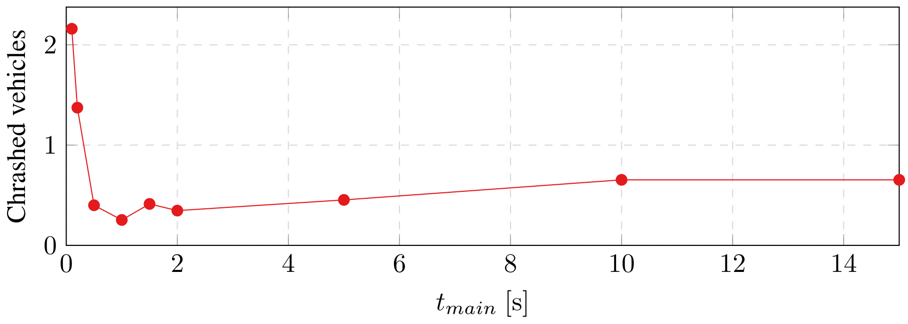

Figure 4.3: A plot showing how the number of collisions in the intersection changes based on different values of t_{main}.

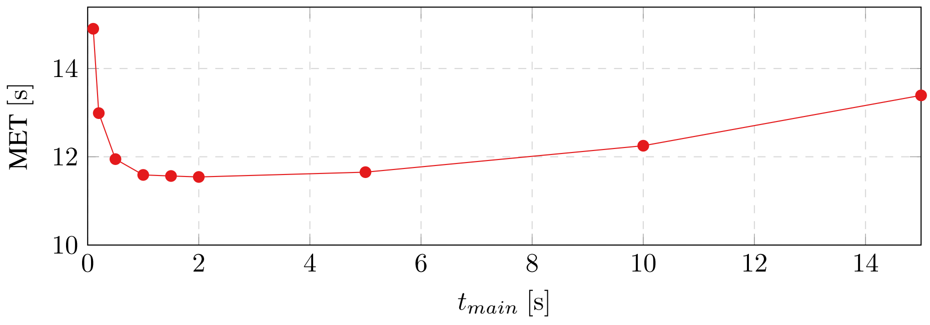

Figure 4.4: A plot showing how MET changes based on different values of t_{main}.

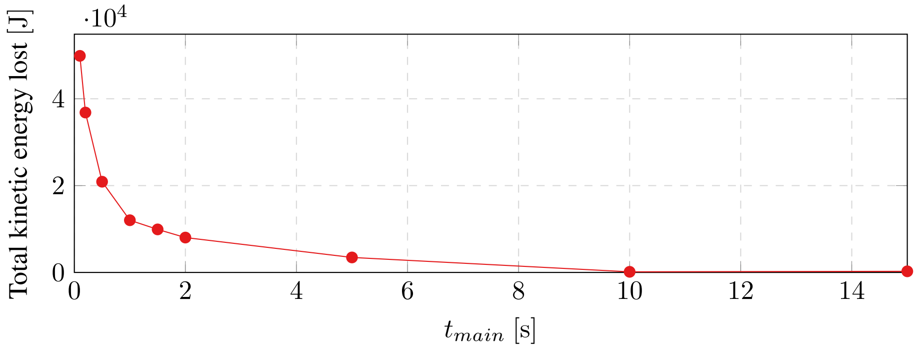

Figure 4.5: A plot showing how total kinetic energy lost changes based on different values of t_{main}.

of t_{main} and $t_{evaluate}$ result in an appropriate behaviour in the IM. When examining time step values the constraint for collisions was prioritised over the other performance metrics. This was necessary because collisions in the intersection might positively affect the other objectives by breaking the rules of our physical world (section 3.3.2 explains how the simulator deals with collisions).

Hypothesis

We expected that the lower values of t_{sim} will result in more collisions in the intersection. This was because we expected that the EP would not be able to avoid unsolvable states. The higher values might be able to avoid collisions, but at the expense of the objectives. Therefore, we expected to find a range of time step sizes where there are few collisions while still maintaining comparatively good objective scores.

Methodology

Using the same method as for testing values for t_{main}, we tested different values of t_{sim} coupled with the most promising values for t_{main}. Because t_{sim} was set to a previously decided value, and there is no proof that different t_{main} values did not have different optimal values for t_{sim}, it was decided to use the time step values between 0.2 seconds and 1.5 seconds for the next experiment. 0.2s was included in order to show whether its performance could be improved by finding a better t_{sim}. 0.1s was excluded because the reasons for its poor performance are closely related to the reasons for the poor performance of 0.2s. If 0.2s then continued to perform badly in the next experiment, it can be assumed that 0.1s also would. 2.0s was excluded because it was very similar to 1.5s, a difference not considered significant.

The values used for t_{sim} can be seen in table 4.2. The values for $t_{evaluate}$ were calculated using equation 3.1 by inserting the values 0.2s, 0.5s, 1.0s and 1.5s for t_{main}. Each scenario was evaluated once for every configuration of t_{sim} and t_{main}.

Table 4.2: The time step values, in seconds, used in Experiment 1 Part 2.

t_{sim}	0.1	0.2	0.5	1.0	1.5	2.0	5.0	10.0	15.0

Results

The results for this experiment are presented in three figures (figure 4.6, 4.7 and 4.8), where each figure contains one of the performance metrics for four different t_{main} values.

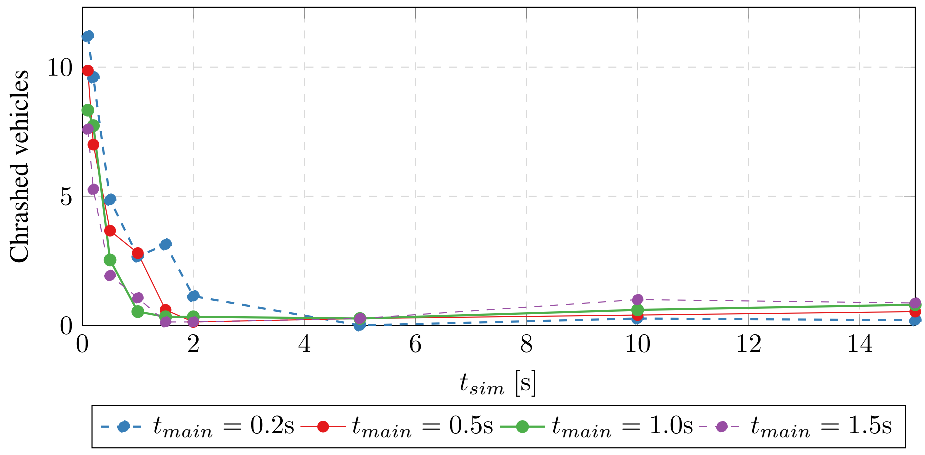

Figure 4.6: Four plots showing how the number of collisions in the intersection changed based on different values of t_{sim}. The four plots differ in which t_{main} is used for the simulations.

The throughput in this experiment is the same as in part 1. All vehicles got out of the intersection and because of this there was not any information when comparing the throughput of different t_{sim} values.

From figure 4.6 it was clear that when t_{sim} was low, the vehicles collided. This is as expected because the evaluation of the speed vector does not consider what happens after t_{main} has passed. As such, the solutions chosen in each step created

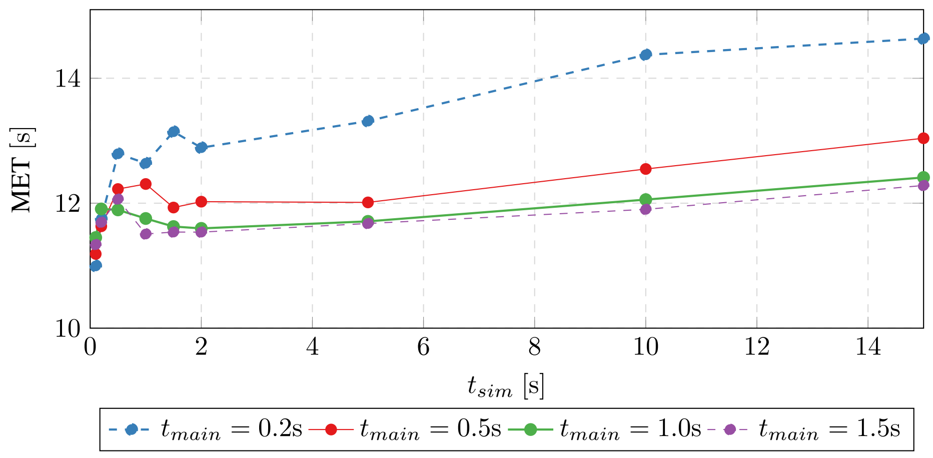

Figure 4.7: Four plots showing how MET changed based on different values of t_{sim}. The four plots differ in which t_{main} is used for the simulations.

unsolvable states in the next step. The number of collisions decreased until t_{sim} equalled either 1.5 or 2.0 seconds, depending on which of the graphs we look at.

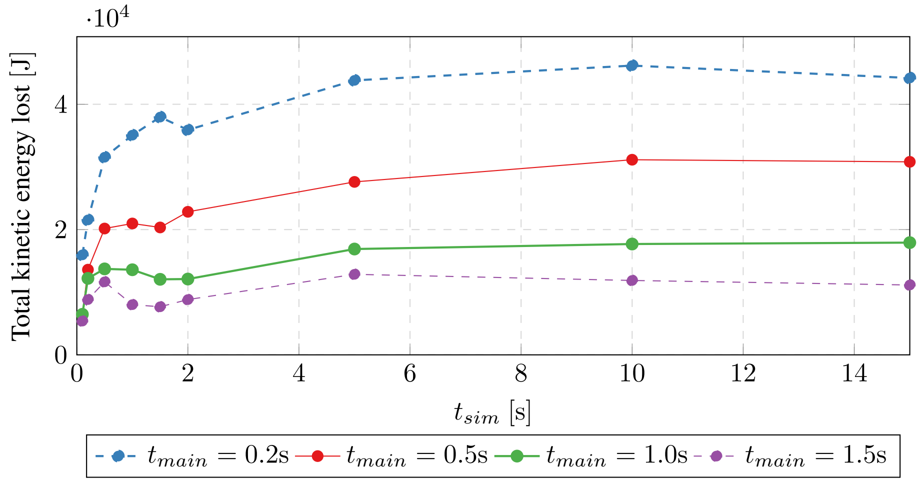

Figure 4.8: Four plots showing how total kinetic energy lost changed based on different values of t_{sim}. The four plots differ in which t_{main} is used for the simulations.

The MET had a slight increase for all t_{main} values after a certain point. The values before this point are not considered because there were a lot of collisions. Based on MET the t_{sim} should be as low as possible, as long as collisions can be avoided.

The total kinetic energy lost behaved similarly to MET in that the value increased with a higher t_{sim}, but the most interesting part about these graphs is that the 1.5 seconds t_{main} graph was consistently lower than other values of t_{main} for every

t_{sim}.

Based on these observations we found that the value 1.5s for t_{main} would perform reasonably. Considering the values for the number of collisions and MET again, with this specific graph in mind, we found that a value of 1.5s for t_{sim} in combination with 1.5s for t_{main} should give the IM a good balance between security and efficiency.

4.2 Experiment 2: Continuous traffic

Goal

The goal of this experiment was to better understand how dividing the intersection management problem into smaller time steps behaves based on the amount of cars in the intersection.

Hypothesis

We expected that the IM would perform worse as the amount of cars increased. This should result in a higher MET and loss of kinetic energy as the spawn rates increase. It is known that the system fails to avoid collisions in some states with a high amount of vehicles. It is much more likely to get those states as the amount of cars increase. Therefore increasing the spawn rate should also increase the number of observed collisions. In addition, as the amount of cars increases, it is expected to pass the limit where it is physically impossible to deal with all the cars without creating a queue.

Methodology

This experiment compared the performance metric scores of the IM (these scores are explained in 3.3) for different spawn rates. The real world simulator ran in a continuous mode. This involved spawning cars probabilistically with a spawn rate (spawn rates are explained in section 3.1.3). Every spawn rate between 0 and 1 was tested in increments of 0.1. No scenarios were used in this experiment. Each simulation was ran for a total of 300s and did not terminate before the time ran out. Each spawn rate was tested five times.

As the previous experiment found reasonable values for t_{main} and t_{sim} these were set to the following values.

- $t_{main} = 1.5\text{s}$
- $t_{sim} = 1.5\text{s}$
- $t_{evaluate} = 3.0\text{s}$

Results

The results are presented in their entirety in table 4.3. In addition two graphs have been included to illustrate the non-linear relationships between spawn rates and total kinetic energy lost, and MET. The table displays throughput, here meaning the total amount of cars that have evacuated the intersection in the given 300s. MET, collisions, throughput and total kinetic energy lost are averages over the 5 runs. Collisions (%) is the percentage of the runs that collided, i.e. a value of 100 meaning that all 5 runs had vehicles colliding in them. A collision is available in appendix C.

Table 4.3: IM performance metrics for different levels of traffic in experiment 2. MET, Throughput and Total Kinetic Energy Lost all increased as the spawn rate increased. The vehicles started to collide when the spawn rate was above 0.6, shown by Collisions and Collisions(%).

Spawn Rate	**Runs**	**MET** (s)	**Collisions**	**Collisions** (%)	**Throughput**	**Total kinetic energy lost** (J)
0.1	5	13.37	0.00	0.00	236.20	4539.93
0.2	5	13.37	0.00	0.00	417.60	16923.33
0.3	5	13.44	0.00	0.00	578.60	54323.80
0.4	5	13.46	0.00	0.00	712.60	126729.28
0.5	5	13.64	0.00	0.00	820.40	343614.86
0.6	5	13.89	0.40	20.00	926.80	768865.80
0.7	5	14.17	0.80	20.00	1025.80	1641376.12
0.8	5	17.59	68.80	100.00	1042.60	4810951.97
0.9	5	25.08	189.20	100.00	943.00	9604215.88
1.0	5	28.08	204.00	100.00	943.60	11658489.02

No collisions occurred in the intersection before the spawn rate was 0.6, where the collisions in both 0.6 and 0.7 can be traced to a single run for each with 2 and 4 collisions respectively. This is in line with the observations from the previous experiments where collisions could only be observed in scenarios with high amounts of traffic. The exact mechanism that caused cars to collide is hard to pinpoint, but manual observation of the runs did not witness of states where collisions were unavoidable, but rather occurring in complex scenarios with many cars.

The measured throughput for lower spawn rates is not very interesting, but simply showed that increasing the spawn rate does in fact increase the amount of cars that had to be routed through the intersection in the given time. The non-linear relationship between throughput and spawn rate is explained in 3.1.3. For higher spawn rates throughput stopped increasing; it went slightly down. This occurred at the same spawn rates that produced queueing and congestion in the inner intersection. As expected, when the queue of vehicles reached back to the spawning zones, no new vehicles would spawn. This was also a contributing factor to the decreasing amount of throughput observed.

When looking at the graph comparing MET to spawn rate it is apparent that as long as the spawn rates were low enough, vehicles could be routed through the intersection with-

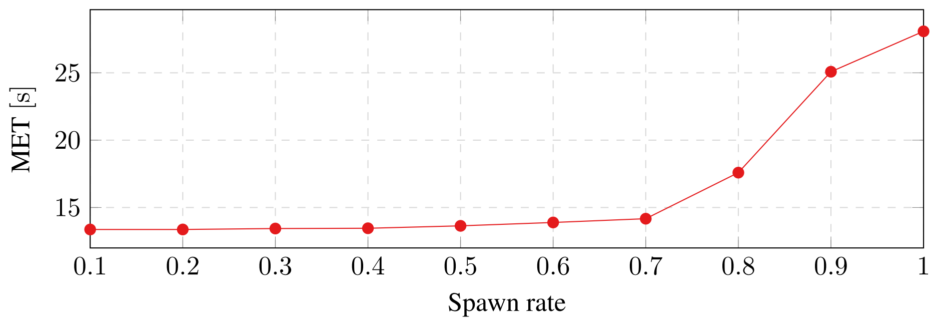

Figure 4.9: A plot showing how MET varies in regards to spawn rate.

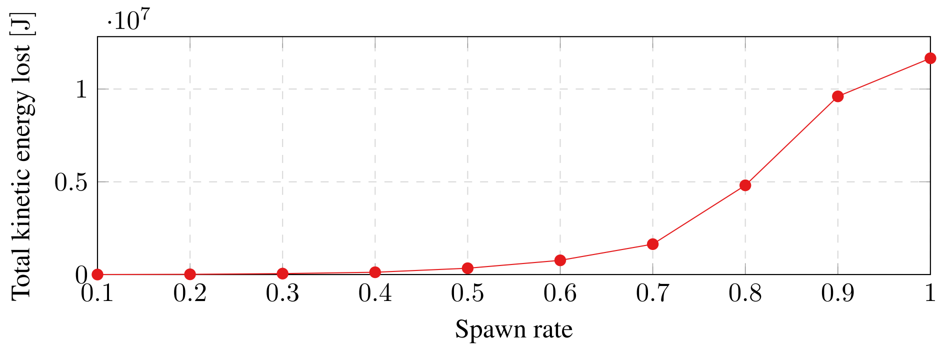

Figure 4.10: A plot showing how total kinetic energy lost varies in regards to spawn rate.

out much delay. A higher MET showed that the IM had reduced the speed of some cars. Unfortunately, as mentioned in 3.3.2, measured MET is not correct when collisions occur. However, it should only be skewed in the negative direction, and the difference is still positive as spawn rate rises. We could therefore conclude that the IM was able to slow vehicles significantly to allow the passage of new cars. Manual observation of the intersection also supported this. Looking at figure 4.11 it is apparent that the IM was slowing the whole group of cars arriving from south(here represented in red) in order to speed up another group of cars arriving from west(speeding cars are coloured green).

4.3 Experiment 3: Scalability

Goal

The goal of this experiment was to investigate how solving individual states of the intersection scales in regards to the amount of traffic when considering collision avoidance. This was in order to help answer research question 2, and to give some insight as to why we could see collisions for high traffic levels in section 4.2.

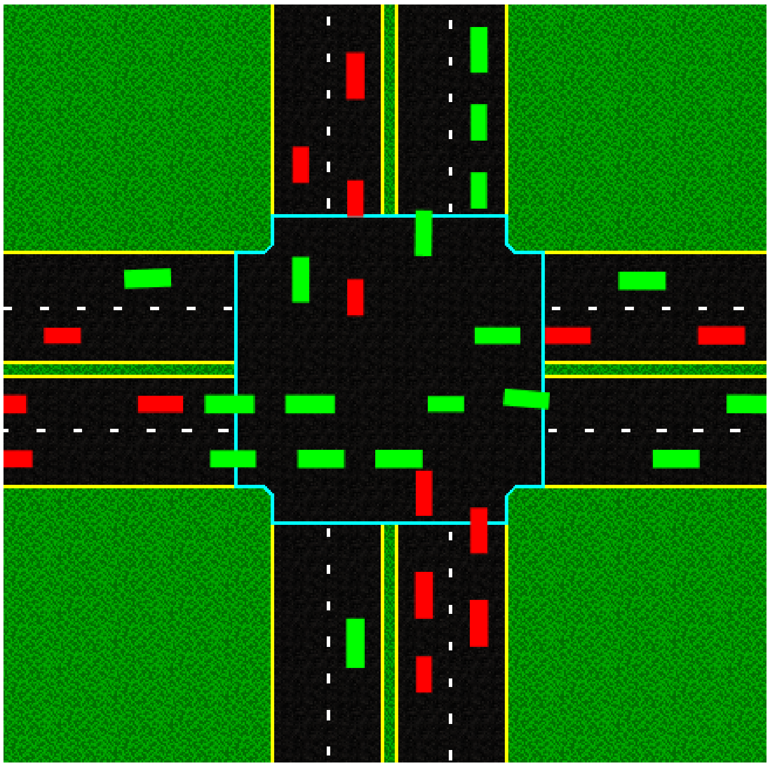

Figure 4.11: Demonstration of how the IM decides to slow down the vehicles arriving from south so that the vehicles arriving from the west can pass. Cars slowing down or stopped are displayed in red. Cars speeding up are shown in green.

Hypothesis

When more vehicles are present in the intersection the EP should, on average, need more evaluations to find a solution that involves no collisions. Knowing that the search space of the EP increases by a dimension when one more vehicle is introduced, it should be expected that the amount of evaluations needed would increase faster than linearly.

Methodology

This experiment was similar to experiment 2 in parameters. By running the IM over $300s$ for each of the spawn rates in experiment 2, we recorded the amount of cars in the intersection and the number of evaluations needed before a solution without collisions was found. If the EP was unable to find a solution without a collision, nothing was recorded. In other words, only states that could be solved in the given amount of evaluations were considered. Because the evaluations of individuals were done in parallel the granularity of the results were constrained to within one generation; 100 evaluations.

The amount of recorded states containing a given amount of cars varied widely. In order to avoid drawing conclusions from statistical outliers, data points were ignored unless there existed at least ten measurements for the given amount of cars.

Results

The result from this experiment can be seen in figure 4.12. It was clear that the EP found a solution within the first generation when a small amount of cars (< 35) were present. However, once more than 35 cars are present the amount of evaluations needed rose rapidly.

When considering the amount of evaluations needed to avoid collisions in terms of spawn rate, as displayed in figure 4.16, it was apparent that the amount of evaluations needed rose sharply when the spawn rate was greater than 0.6.

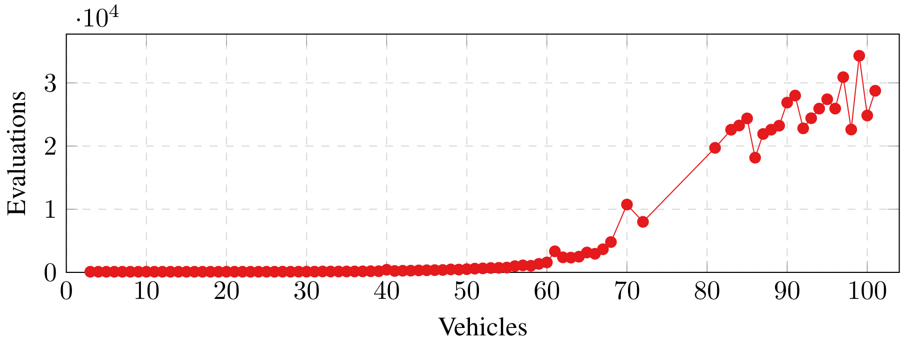

Figure 4.12: A plot showing the number of evaluations needed to avoid collisions based on the number of vehicles in the intersection.

4.4 Experiment 4: Objectives

Goal

The goal of this experiment was to investigate the effects of each objective on the overall performance of the system. This was done by removing each objective from the evaluation in four separate runs. This directly related to research question 3. The objectives are described in detail in section 3.2.1.

Hypothesis

When removing each objective, we expected the following behaviour:

- **Removing total kinetic energy lost**

  Minimising total kinetic energy lost should, on average, reduce the amount of kinetic energy lost over the course of a run. No matter what spawn rate is used. Therefore it was expected that not including this objective would consistently produce results that wasted more energy.

- **Removing starvation**

  The starvation objective was implemented to avoid single lane starvation. The behaviour of the continuous IM in section 4.2 showed no signs of starvation. It is hard to say exactly why, but it should be noted that maximising the distance travelled also keeps vehicles from standing still. Therefore we could not predict whether starvation would be observed when removing this objective, but the results would still point to whether this objective is important for the function of the IM or not.

- **Removing tiny throughput**

  Evacuating vehicles from the inner intersection as fast as possible should be important to avoid congestion and opening conflicting lanes of traffic more. Therefore we expected that disregarding this objective would lead to more queueing, higher MET, more vehicles in the intersection, more complex states and more collisions. However, this objective should only be important when the amount of cars is high, as maximising the distance travelled would evacuate a lot of cars in itself.

- **Removing distance travelled**

  The main thought behind maximising distance travelled was to make sure the vehicles travelled as far/fast as possible in every time step. Removing this objective should have an adverse effect on MET.

Methodology

In order to test the effect of each objective, the IM was tested with the absence of each objective. The performance, in terms of the overall performance metrics defined in section 3.3, could then be compared across different configurations of active objectives. For this experiment, the investigation in regards to MET and total kinetic energy lost was constrained to spawn rates below 0.7, as the physics surrounding collisions make comparisons harder when they occur.

Results

The results are presented based on performance metrics. Figure 4.13 shows the relationship between the different objective configurations and MET. Results with a spawn rate above 0.7 have been omitted as they include collisions. The graph shows a clear trend: not optimising for distance travelled produced a significantly higher MET. As long as distance travelled was included there was no significant change from the baseline acquired in section 4.2.

Interestingly, ignoring distance travelled was also the one that stood out when measuring the total kinetic energy lost (see figure 4.14). Removing the objective specifically designed for minimising this loss had an almost negligible effect, even scoring slightly better for values above 0.3.

No statistically significant results were found when comparing for collisions. Nor did the results show any deviation from the baseline when removing the tiny throughput or starvation objectives.

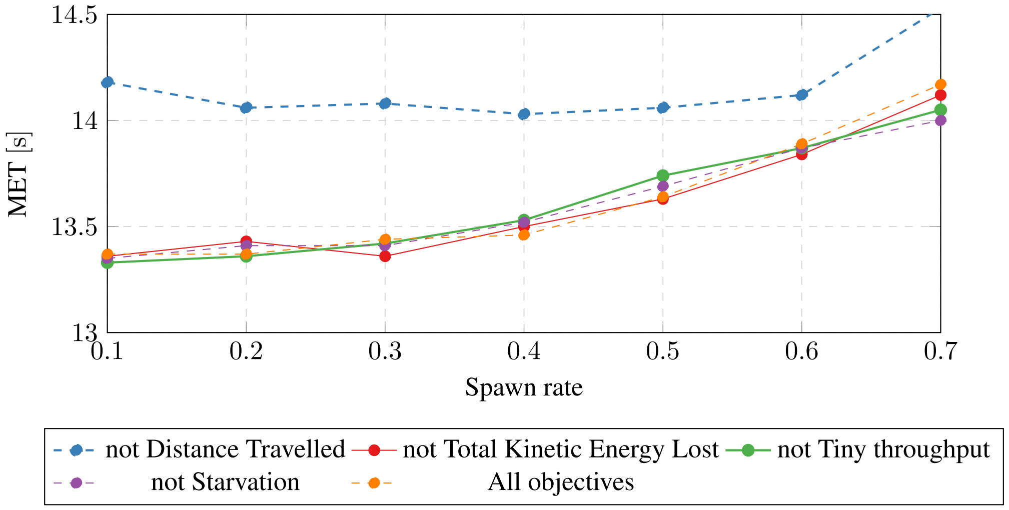

Figure 4.13: Figure showing how MET was affected by removing each of the objectives in the MOEA at different spawn rates. There are four plots, each showing the MET value when one of the objectives were not evaluated. The last plot shows the default behaviour when all objectives were evaluated.

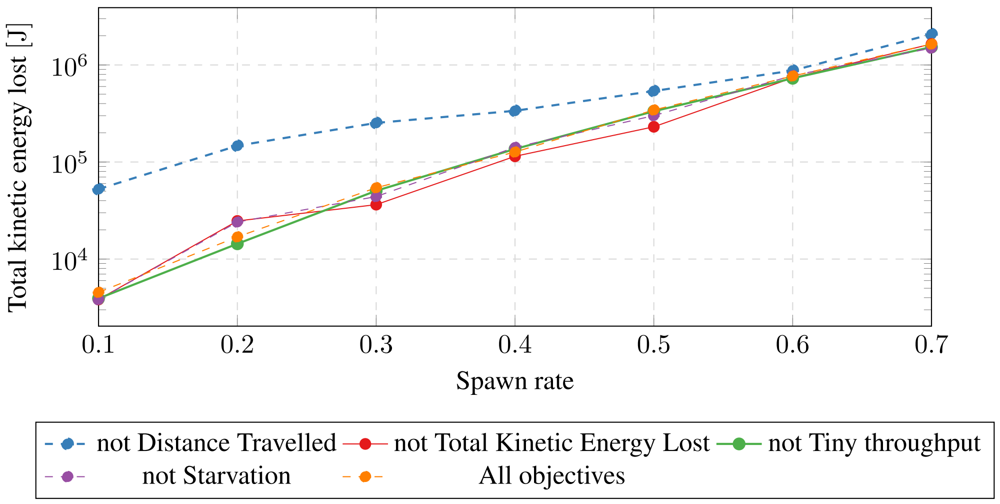

Figure 4.14: Figure showing how total kinetic energy lost was affected by removing each of the objectives in the MOEA at different spawn rates. There are four plots, each showing the total kinetic energy lost value when one of the objectives were not evaluated. The last plot shows the default behaviour when all objectives were evaluated.

4.5 Experiment 5: Conflicting objectives

Goal

The goal of this experiment was to better understand how the optimisation objectives (explained in section 3.2.1) conflict in both complex and non-complex scenarios.

Hypothesis

We expected that the EP would be able to find close to optimal solutions when provided with a simple scenario, where there are few vehicles. When the scenario becomes more complex, it was expected that the EP would struggle to find one dominating solution. This would result in a Pareto-set where each solution was considered equally good, without any further specification of which objectives are more important.

Methodology

This experiment looked at results from the EP when provided with different scenarios. The scenarios used should reflect the intersection both when there are few cars and when the intersection is crowded. This should ensure that the results show how the chosen objectives conflict in the varying situations the IM encounter when run continuously.

The solutions found in the EP served as examples of how the objectives may conflict, as there was no guarantee that the EP would find the same results when provided with the same scenario several times. This experiment would still be interesting to better understand how the EP makes compromises on one objective to improve another.

The experiment ran the EP on three different scenarios (figures available in appendix B). All scenarios were created by running the setup from experiment 2, differing only on spawn rate. The first scenario was generated with a spawn rate of 0.4, had few cars in the inner intersection and avoiding collisions was easy. The second scenario was generated with a spawn rate of 0.8, had a lot of vehicles in the intersection and the EP, with the settings from experiment 2, would struggle to avoid collisions. The third scenario was generated with a spawn rate of 0.9, had a lot of cars arriving in the intersection and the problem of avoiding collisions was similar to the second scenario. The spawn rate only mattered when generating the scenarios, as no vehicles would spawn while this experiment ran, since the EP only calculated one solution.

Results

The results by running the EP on a simple scenario is presented in table 4.4. Given that the optimal solution for distance travelled and tiny throughput is -1.0 and 0.0 for starvation and total kinetic energy lost, it is clear that all objectives were either optimal or close to optimal. As the EP did not have to struggle to avoid collisions by braking several vehicles, the result depended only on the MOEA being able to evolve a speed vector where each value was the speed limit. This problem is similar to the well known one-max problem, and is a lot simpler than the more complex scenarios in this experiment.

Table 4.5 presents the Pareto-set found by running the EP on the complex scenario generated with a spawn rate of 0.8. It should be noted that while these values seem a lot worse

Table 4.4: Table showing the Pareto-set found by the EP when run on a simple scenario generated with a spawn rate of 0.4.

Distance travelled	**Total kinetic energy lost**	**Starvation**	**Tiny throughput**
-0.9998	9.0464×10^{-5}	0.000	-1.000

than in the first scenario, it may still be close to optimal when considering that the EP should avoid collisions.

Table 4.5: Table showing the Pareto-set found by the EP when run on a complex scenario generated with a spawn rate of 0.8.

Distance travelled	**Total kinetic energy lost**	**Starvation**	**Tiny throughput**
-0.8935	0.0478	0.0000	-0.7500
-0.8962	0.0488	0.0000	-0.7500
-0.8948	0.0483	0.0000	-0.7500
-0.8916	0.0464	0.0000	-0.7500
-0.8895	0.0442	0.0120	-0.7500

An interesting value to note in the table is the last individual. Both starvation and distance travelled were worse than its peers, but total kinetic energy lost was improved. One explanation for this result is that the last solution chose to starve one car in order to not having to brake another. The reason that distance travelled is lower in that solution, might be because the starved car had the opportunity to travel further than the favoured car if it were not starved.

The Pareto-set found when running the EP on the complex scenario generated with a spawn rate of 0.9 is presented in table 4.6. Since it was hard to avoid collisions in this scenario, it is likely that the reason for a bigger Pareto-set was that the optimisation of the objectives had less generations to evolve and the search space was larger than in the other scenarios.

There are two possible reasons for why there are several non-dominated solutions in the two complex scenarios. Either the EP has to compromise on one objective to improve another, i.e. the objectives are conflicting and when choosing a solution we have to decide what objectives are more important, or, because of the complexity in the scenario, the EP struggles to find the single dominating solution while still avoiding the collision constraint with the given amount of evaluations.

Table 4.6: Table showing the Pareto-set found by the EP when run on a complex scenario generated with a spawn rate of 0.9.

Distance travelled	Total kinetic energy lost	Starvation	Tiny throughput
-0.7942	0.1816	0.0000	-0.6471
-0.7855	0.1799	0.0000	-0.6471
-0.7702	0.1714	0.0148	-0.6471
-0.7775	0.1762	0.0148	-0.6471
-0.7850	0.1713	0.0281	-0.6471
-0.7695	0.1609	0.0428	-0.6471
-0.7790	0.1804	0.0000	-0.7059
-0.7849	0.1826	0.0000	-0.7059
-0.7859	0.1904	0.0000	-0.7059
-0.7770	0.1772	0.0134	-0.7059
-0.7755	0.1771	0.0151	-0.7059
-0.7644	0.1663	0.0281	-0.7059
-0.7656	0.1690	0.0281	-0.7059
-0.7714	0.1758	0.0281	-0.7059

4.6 Discussion

This section discusses the findings of this chapter in relation to the overall research goals set in the introduction.

4.6.1 Independent time step modelling

When splitting the large and continuous problem of intersection management into smaller time steps, it is clear that the two time variables t_{main} and t_{sim} had a large impact on the performance of the IM. Small values for both variables caused a large amount of collisions, but for different reasons. A small t_{sim} has the obvious drawback that the IM does not evaluate far enough into the future to avoid collisions in the next time step. Looking at the vehicle behaviour from the results of section 4.1 when varying t_{sim} it is clear that a lot of the collisions at smaller values were caused by this effect. An example scenario is presented in figure 4.15. Here it is clear that the circled car will not have time to stop before crashing into the parked car. The collision of vehicles affects the other performance metrics, but should only affect them positively. This means that MET and total kinetic energy lost should both decrease as opposed to the solution where a collision is avoided. The reasoning behind this is available in section 3.3.

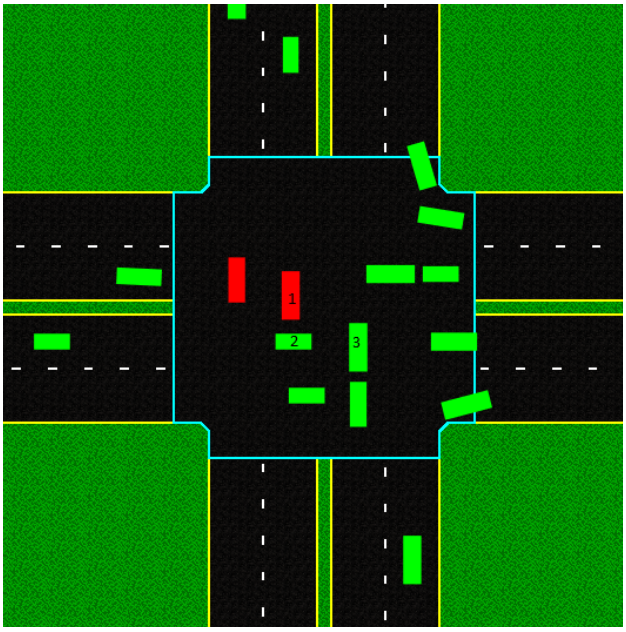

Figure 4.15: Unsolvable state; vehicle 2 and 3 will crash because vehicle 2 is moving too fast to stop for vehicle 3 within the given $t_{sim} = 0.2s$. Vehicle 1 is also on a collision course with vehicle 2.

Smaller values of t_{main} also produces an increased amount of collisions, but the reason is not as obvious. Because the IM makes no effort to avoid vehicles from conflicting trajectories occupying the intersection at the same time, the setup can be subject to *deadlocks*, here meaning a state where collisions are avoidable in the next $t_{evaluate}$, but evacuating all the cars from the inner intersection has become impossible. Figure 4.2 shows an example deadlock from the $t_{main} = 0.1$ tests. As the cars move slowly into this deadlock the amount of solutions in the search space that do not involve a collision slowly decreases to the point where the cars are locked very close to each other and any acceleration of any car will result in a collision. As a result the EP then, for an infinite amount of time, has to come up with a solution that involves holding all vehicles still. It then seems reasonable that, as the EP is limited by both computation time and the converging nature of MOEAs, it eventually fails to find a solution that creates no collisions, resulting in vehicles crashing. This problem is exacerbated by smaller values of t_{main} because the cars move a very small amount each time, and will therefore more evenly distribute which of the cars is favoured to move. As t_{main} grows larger the IM will favour a smaller subset of the cars for a longer amount of time, largely avoiding deadlocks.

It was decided to keep the amount of evaluations per second of real time consistent between different configurations of the IM; as t_{main} becomes smaller the available time for the EP to find a solution decreases. This is explained in section 3.1.4. Therefore, some of the loss of performance when lowering t_{main} also has to be attributed to the lower amount of evaluations available to the EP.

There are also drawbacks to using larger values for t_{sim} and t_{main}. When using larger values for $t_{evaluate}$ (remember that $t_{evaluate} = t_{main} + t_{sim}$) the IM has to find a solution that is applicable for a very long time. Therefore, once complex situations arise, it can be hard to find a solution that has high enough speeds to get a reasonable MET where the high speeds can also be used for a long amount of time. Figure 4.4 shows a clear relationship between high values of t_{main} and a rise in MET.

Considering total kinetic energy lost, it is apparent that the IM will waste more energy the smaller values are used for t_{main}. This can be seen by the strictly decreasing graph in figure 4.5. This is most likely closely related to the reasoning behind the high amount of collisions for a small t_{main}. The less amount of time the IM deploys each solution the more often the speed of each vehicle is likely to change within one scenario. The exact reason why the EP does not keep favouring speeding up the same vehicles and braking others across multiple time steps is difficult to pinpoint. While the objective for minimising total kinetic energy lost should keep favouring the same cars (it is more expensive to brake a fast car than a slow one), the EP is limited by computation time and the nature of MOEAs. Therefore it does not always produce perfect results, which should result in higher loss of energy on smaller values of t_{main}. Interestingly, varying t_{sim} shows a less significant inverse relationship to what was found with t_{main}. Figure 4.8 shows a clear, but small, rise in total kinetic energy lost as t_{sim} rises. This is most likely caused by the decrease in the available amount of solutions that do not violate the constraints as $t_{evaluate}$ grows larger, which in turn leaves the EP with less choices for solutions that improve total kinetic energy lost.

Perhaps the most important observation from the testing of different time step values is the fact that there is no set of values that performs best in all scenarios. Therefore, for the later usage of the IM, one must find a compromise between the available objective scores. In this project the values of $t_{main} = 1.5s$ and $t_{sim} = 1.5s$ were chosen. This was because they displayed the most promising results in terms of collision avoidance, while maintaining what was considered acceptable values in other objective scores[1].

Examining the behaviour of the IM over a continuous time period showed that there definitely exists some amount of cars where the EP is unable to solve the problem (fast enough); causing collisions. According to table 4.3 it is clear that there is no hard limit, but rather an increasing probability of collisions as the amount of vehicles increases. There are many physical scenarios that lead to collisions, but a high amount of cars seems to be a common occurrence in states that lead to a collision. The relationship between the amount of cars present in the intersection and the complexity of each state is explored further in section 4.6.2.

The IM shows an ability to form queues over time. Not necessary orderly queues like the ones seen in normal traffic, but it clearly displays an ability to slow multiple vehicles

[1]Acceptable here meaning in relation to other time step values, not necessarily acceptable in general.

travelling the same trajectory for a longer amount of time. This also translates across different time steps. Figure 4.11 shows that the cars arriving from the south are braking in unison for the eastbound vehicles.

Reading the MET values from table 4.3 and comparing them to the best case MET of $13.12s$ (found in 3.3.1) it is apparent that as long as the amount of vehicles is manageable the IM demonstrates an ability to route vehicles through the intersection with an MET close to optimal. The difference between the optimal value and the one demonstrated by the IM is caused by two things. Firstly, because the IM tries to avoid collisions, when two cars travelling at the speed limit are going to crash one has to be slowed. Secondly, because the cars are out of the scope of the IM once they move out of the intersection, there is no incentive, in terms of distance travelled, to hold the full speed until the end of the intersection in the last t_{main}. The distance travelled will be the same. This problem should be mitigated by optimising for total kinetic energy lost, but an explanation for why that sometimes fails is available in 4.6.3.

There exists a theoretical limit to how many cars can be routed through the intersection in a given amount of time. Such a limit must exist, as vehicles cannot occupy the same space at the same time without colliding, and there is a limit to how fast each vehicle can traverse the inner intersection. Finding that exact limit for this system is impossible given a realistic amount of time, as it would involve finding the optimal solution for every time step dependent on each other. This limit is likely a largely contributing reason to why the MET rises sharply when the spawn rate is above 0.7.

Perhaps the biggest drawback, but also the most interesting part, of solving one state at a time is that there is no direct relation between the objectives being optimised for and the performance metrics used to measure how the system behaves. The difficulties shown at smaller values of t_{main} bear witness to the difficulties of solving single states independently. While a single state can be optimised very well in relation to its objectives, it is apparent from the emergent deadlocks and complicated states that this does not always translate into a good solution when considering the continuous problem.

Single lane starvation is a problem that several of the related works in chapter 2 dealt with. Problematic starvation was never observed for this system. It happened that cars would stop completely, but never for too long. The spawn rate used in these experiments was symmetric, meaning that vehicles arrived from all directions with the same frequency. Asymmetric spawn rates could have been used to attempt to cause scenarios where starvation was more likely.

4.6.2 Scaling

The results shown in figure 4.12 clearly demonstrated that the EP uses more evaluations to find a solution without collisions as the amount of vehicles in the intersection increases. There are two main reasons to expect a result similar to this. We imagine the non constraint violating part of the search space as the full set of paths that can be travelled by all the cars in the next $t_{evaluate}$ without causing a collision. Introducing another car anywhere in the intersection will make the amount of valid paths of travel for the already present vehicles decrease or stay equal. Therefore, the amount of valid solutions in the search space decreases or stays equal. In addition, introducing one more vehicle also adds a dimension to the search space, drastically increasing the size of the search space as a whole. These effects together makes each state harder to solve for the EP when increasing the amount of vehicles.

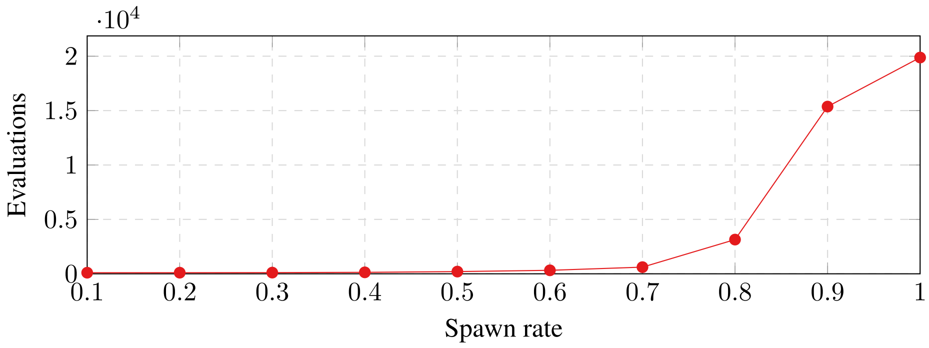

Figure 4.16: A plot showing the number of evaluations needed to avoid collisions based on spawn rates.

Looking at the same data, just ordered by spawn rate (figure 4.16), tells a very similar story. Higher spawn rates means more cars, which in turn should increase the average complexity of each state. This is in line with the previous observation that higher spawn rates cause more collisions. In addition, the queueing behaviour discussed in 4.6.1 rapidly increases the amount of cars present in the intersection. This further exacerbates the exponential increase in complexity we would expect based just on the knowledge that more vehicles leads to increased complexity. The higher amount of evaluations needed simply to avoid violating the constraint of collisions also negatively impacts the amount of evaluations available to the MOEA to better the score for other objectives. As a result, the IM will more often fail to evacuate vehicles as fast as possible, further increasing the amount of vehicles in the intersection. This is supported by the observation that the rise in evaluations needed clearly correlates with the worse performance metrics observed in figures 4.10 and 4.9, based on spawn rate.

If the IM proposed in this work is to be applicable for real world use it has to able to accommodate a large amount of vehicles while avoiding collisions. As it stands, the current IM struggles to find a valid solution for a large amount of vehicles, within the computational constraints used in these experiments. Therefore, the current IM can not be deployed in the real world, without some restriction on the amount of cars entering the intersection. In addition, CSGs and FGs, presented in chapter 2, have been used to effectively combat complexity issues in intersections.

4.6.3 The objectives

Maximising the distance travelled in each time step has a large effect on the overall performance of the IM. Looking at the results for MET (figure 4.13) and total kinetic energy lost (figure 4.14) it is clear that the IM performs considerably worse when this objective is ignored. This effect is most noticeable at lower spawn rates. At the same time, some of the more adverse behaviour, such as deadlocks and gridlocks, present at higher spawn-rates can be attributed to this objective. When the IM is holding off some cars to allow the passage of others it will move those slightly forward and into the inner intersection. There is no incentive to make sure the passing lanes are open to traffic in the next time step.

As can be seen in figure 4.14, minimising the total kinetic energy lost in each time step does not attribute to a lower overall loss of kinetic energy. For spawn rates lower than 0.7 there is no significant trend that shows that removing this objective has any effect at all. This has two immediate explanations. First, the weighting used by the DM to choose a solution from the Pareto-set only favours optimising for minimising total wasted kinetic energy when the difference is large compared to the worst case scenario. The score for this objective when comparing it to others is:

$$\alpha_{ek}\frac{E}{E_{worst}} \tag{4.2}$$

,where α_{ek} is the weight for this objective, E is the kinetic energy lost in this solution and E_{worst} is the kinetic energy lost when evaluating a solution that slows every car as much as possible. This value then lies somewhere in the range of $[0, \alpha_{ek}]$. If E_{worst} is a high number the difference between two solutions becomes very small. Therefore, the effects of this objective gets smaller as the amount of stoppable energy in the intersection gets higher. Second, optimising for energy loss in single time steps can have an adverse effect when looking at a continuous time scale. If two cars are heading toward the intersection at the speed limit on conflicting trajectories one has to be slowed to allow passage of the other. The closer the cars are to each other the harder one has to brake. Increasing the size of $t_{evaluate}$ allows the vehicles to see this collision occur earlier, resulting in a smaller change in speed and a smaller amount of energy lost.

No significant changes were observed when not optimising for tiny throughput and starvation. Some small trends were observed for the amount of collisions and how often collisions occurred, but no statistically significant relationship could be established. This could be because no such relationship exists, or because the results are counted over longer periods of continuous simulation. As seen in section 4.5 the effects of the objectives, and how they conflict amongst themselves, will vary greatly depending on the specific state that the IM is considering. Looking at the combined data from many such states, and their respective solutions, instead of single states can lead to obfuscating the true relationship between the objectives and the performance metrics.

When looking at the results from section 4.5 it is clear that when there are few vehicles in the intersection the objectives do not conflict much. This makes sense, as when no vehicles are on a colliding path the collision constraint can not be violated, and the objectives themselves do not conflict. The best way to minimise the loss of kinetic energy, travel the furthest distance, get as many cars through the inner intersection and avoid starvation is to maximise the speed of each vehicle.

It was also shown that the objectives may conflict when the amount of vehicles in the intersection is high, or the state is complex. The data in table 4.6 shows that the EP finds a more diverse Pareto-set for a high-complexity intersection state. However, no conclusion can be made about whether an optimal solution without conflicts exists, or if conflicts can occur in every such state. Answering those questions would involve searching through the whole search space (after deciding on some discrete step-values as the genetic encoding in this project uses continuous variables), a process which is unfeasible for a large amount of vehicles.

4.6.4 Real world application

In order to use this system in the real world autonomous vehicles, and the intersections surrounding them, need the following supporting infrastructure.

Generating states

As described in chapter 3 the IM is dependent on generating a correct state of the current world in the intersection. This means that it needs to know the exact position of all vehicles, their respective speeds and all other physical stateful attributes. The states should also contain information regarding the environment. Therefore there is also a need for sensors in the intersection or on the vehicles to collect the relevant environmental data.

Predicting behaviour

In order to trust that a deployed solution will perform as expected the behaviour of the vehicles has to be predictable. One of the limitations of this thesis is the lack of realism in the simulator. A lot of things have been simplified in order to ensure predictability, such as the vehicles having strictly linear acceleration and deceleration profiles and not having to slow down for sharp turns. However, as long as the IM, given a set target speed, can accurately predict the position of a car in the future the realism of the mechanism that allows it to do that only has to match the world the IM is acting upon. In this project that reality was a simulator. The requirement of vehicles being able to predict their own future position can be found in other real time autonomous intersection management applications, such as FCFS [8].

Communication

Because the IM architecture in this project is based on a central IM with total control, loss of communication can be disastrous. The safety is in part guaranteed by the IMs ability to accurately, and timely, get the information it needs from the vehicles to generate a scenario. It also has to be able to relay the target speeds back to the vehicles on time. Therefore, unless some workaround not discussed in this thesis is available, perfect communication is a prerequisite for implementing this solution in the real world. Vehicle ad hoc networks [20] could be used instead of the V2I infrastructure chosen in this project. This would then include choosing a leader amongst the vehicles that is responsible for acting as the IM.

Chapter 5

Conclusion

This thesis has evaluated the field of intersection management. After an evaluation of the current fields of study a hybrid approach (explained in chapter 3) of related methods has been suggested, implemented and evaluated. This chapter serves as a conclusion of the whole thesis.

The background chapter (chapter 2) gave a review of the current state of intersection management. We decided that it would be interesting and useful to investigate the properties of an IM that was created from joining two promising parts of the field. In addition this chapter explained the relevant theory behind MOEAs for this thesis.

Chapter 3 describes the overall methodology of the experiments. As a part of that methodology the proposed IM was described in detail.

In order to answer the research questions the system was tested with several experiments. These experiments, including their specific methodology and results, were described in chapter 4. In addition, that chapter provided a discussion of the findings.

5.1 Goal evaluation

The goal of this project was to ” *investigate the real time use of a Multiobjective Evolutionary Algorithm to manage traffic through an intersection.*”

In order to achieve this we created an IM, divided into two logical parts. One that could control the vehicles in the intersection by modulating their speed, here known as the IM. The other part, the EP, could analyse a state of the intersection and, using an MOEA, find a speed for each vehicle in the given state. A simulator was implemented in order to simulate the real world that the IM would act upon, and to run simulations for evaluating objective scores in the EP. The project showed that an MOEA could be used in real time to control the intersection. However, exploring the performance of the IM showed that there are limitations to this kind of intersection management.

The research questions, posed in section 1.2, defined the specific parts of the system that this thesis wanted to explore. The system was evaluated under the performance metrics defined in section 3.3.1.

Research question 1: *"What are the effects of using independent time step modelling to determine actions for the system, when looking at effects at an infinite time horizon?"*

The performance of the system was shown to be very dependent on the time step parameters. Using smaller values for t_{main} (the variable that determines how long each solution is deployed) lowered the amount of time each solution is used. When this variable is very low the IM is constantly deploying new solutions. Frequently reevaluating what speed the cars should hold allows the IM to behave more erratically, frequently changing what vehicles should be favoured for intersection traversal. Smaller values ($t_{main} \leq 0.2$s) significantly increased both the MET and the total kinetic energy lost. The IM was observed not consistently prioritising the movement of the same vehicles, leading to more vehicles being stuck in the inner intersection, leading to deadlocks. Using higher values for t_{main} stabilised the behaviour of the IM. However, deploying a solution for too long increased the amount of time a solution has to be valid for, in turn increasing MET.

When varying t_{sim} (the variable that determines the amount of time each solution is evaluated beyond t_{main}) it was shown that collisions could be avoided for medium amounts of traffic, as long as t_{sim} was large enough. Using too small values for t_{sim} was shown to cause a large amount of collisions because the IM would not have enough time to stop cars in the next state. As was shown for t_{main}, large values of t_{sim} also caused a slight increase in MET.

When using the chosen values for t_{main} and t_{sim}, the IM was consistently able to avoid collisions for spawn rates below 0.6. In addition, the MET was close to the optimal values calculated when ignoring collisions. Observing the behaviour of the vehicles clearly showed that the IM kept a close to maximum speed for the vehicles, only significantly slowing them down to avoid collisions in the inner intersection. When the amount of cars was increased, and the inner intersection became congested, the IM displayed behaviour where it would slow a group of cars in order to allow the passage of another group. This closely resembled the explicit behaviour found in related works. Single lane starvation was never observed.

Research question 2: *"How does the complexity of solving independent time steps scale with the amount of cars present in the intersection?"*

In order to understand why the IM behaved differently when presented with different amounts of cars we wanted to investigate how the complexity of each state grows as more cars are introduced. As the amount of cars in the intersection grew, so did the search space that the EP had to traverse.

It was shown that the average amount of evaluations needed by the MOEA, before a solution without collisions was present in the population, rises faster than linearly. At the most, the average was close to half the amount of available evaluations, when only considering scenarios where the EP was able to avoid collisions. This indicated that the rising complexity of each state was one of the contributing reasons why the IM was not always able to avoid collisions. Because collision avoidance is an important part of intersection management, the rising complexity is a limiting factor for real world usage of the proposed IM.

Research question 3: *"How do the chosen objective functions contribute to the effectiveness of the intersection manager?"*

The contribution of the chosen optimisation objectives was measured by comparing the performance of the IM with, and without, optimising for each objective. It was shown that, for smaller spawn rates where collisions do not occur, optimising for distance travelled was very important. Not optimising for distance travelled caused a large upward shift from the MET obtained when optimising for all objectives. No other objectives had any effect here.

It was hypothesised that not optimising for the total kinetic energy lost would have a similar effect on the performance metric pertaining to the total kinetic energy lost. However, no such effect was observed. This was most likely caused by two factors. First, there is no guarantee that optimising for the loss of kinetic energy in every time step translates to a good performance score. Simply because avoiding slightly slowing down a vehicle may lead to having to slow it down a lot more in a subsequent time step. Second, the DM used in this project is a very simplistic one; using only a weighted sum of the objective scores for each Pareto-optimal solution to determine a winner.

The conducted experiments did not show any deviation from the norm when looking at the starvation and the tiny throughput objectives. Those objectives were designed to be most effective when the amount of traffic is high and there is a large amount of congestion in the inner intersection. Under those circumstances there is also a high probability of collisions, at which point the measured MET and total kinetic energy lost are not good for making comparisons.

When looking at how objectives conflict in differing scenarios it was shown that when there are few vehicles in the intersection the objectives do not often conflict. In a scenario with low amounts of traffic the MOEA was shown to find a single good solution. However, the EP was shown to find several Pareto-optimal solutions for two complex scenarios involving many vehicles in the intersection.

5.2 Contributions

The current research in intersection management, as presented in chapter 2, has found many different ways of managing intersections. Some explicit methods that support continuous control of the intersection, and some that use optimisation techniques to optimise single scenarios. This thesis suggests, implements and tests an IM that finds new ground in the field by combining those methods.

It was found that the IM proposed in this project was able to efficiently route vehicles through the intersection safely, with only small deviations from the measured optimal MET for small to medium amounts of vehicles. However, the IM proposed was also shown to be sensitive to both its own internal parameters and the complexity of the intersection states it had to solve. This thesis found the effects of the most important parameters, and identified the primary reasons why collisions occur at higher levels of traffic.

5.3 Further study

This section contains our suggestions for further research.

Avoiding congestion in the inner intersection

It is apparent from the results in chapter 4 that congestion of the inner intersection is a big contributing factor to the problems of the IM approach described in this thesis. It would be interesting to evaluate how the IM performs when more direct methods are used for avoiding congestion.

In order to reduce the congestion a method inspired by related work could be applied. Yan et al. [25] used the notion of Compatible Stream Groups to avoid collisions in the intersection. Compatible Stream Groups are covered in section 2.2. This could be implemented by forcing the IM to only consider vehicles in trajectories that do not conflict. However, determining which Compatible Stream Groups should be active in a given state is not an easy problem.

A more indirect approach is to add another optimisation objective that seeks to avoid congestion by limiting the amount of cars present in the inner intersection. This would be closely related to the current objective of tiny throughput; evacuating cars from the inner intersection as fast as possible, but focus on keeping them out. Ideally, this could also lead to more specific queueing behaviour. An investigation of this objective would also involve finding a good compromise when selecting a solution, as some cars have to occupy the inner intersection in order to be evacuated from the intersection as a whole.

Deadlocks (described and observed in section 4.6.1) are not only a result of congestion in the inner intersection, but also a confounding factor. When the inner intersection is blocked by a deadlock it follows that, as new vehicles move into the inner intersection, they are stuck, making the problem worse. In order to avoid this form of congestion deadlocks have to be avoided. More explicit methods for deadlock detection would be interesting.

Complexity reduction

Because it was shown that the complexity of each state is one of the reasons why collisions occur in the current system we think it would be interesting to see how it behaves when efforts are made to reduce this complexity. This could also give more insight toward research question 2. Yan et al. [25] introduced the concept of Fundamental mini-groups. They are groups of adjacent vehicles that follow the same trajectory. This allows the intersection manager to treat the group of cars as a single entity. The same concept could be applied here, reducing the search space significantly. Reducing the amount of cars in the intersection by some external mechanism should also be explored. Such as the traffic lights limiting the amount of cars merging into a highway, commonly found in the USA.

Asymmetric spawn rates

One of the limitations of the research conducted in this thesis is that we only examined spawn rates symmetrical between directions. This may have been a contributing factor to why single lane starvation was never observed over large periods of time. For instance, if the amount of cars from the east was significantly larger than what came from the north it is plausible that the EP would favour the eastern cars and stop the northern ones.

Bibliography

[1] AIM, 2016. The aim4 simulator v1.0.3. `http://www.cs.utexas.edu/~aim/`, accessed: 2016-01-20.

[2] AIMSUN, 2016. `http://www.aimsun.com/wp/`, accessed: 2016-05-12.

[3] CORSIM, 2016. `http://mctrans.ce.ufl.edu/mct/`, accessed: 2016-05-12.

[4] David, H., 2015. Moea framework user guide. version 2.6. `http://www.moeaframework.org/`, accessed: 2015-12-6.

[5] Deb, K., 2001. Multi-objective optimization using evolutionary algorithms. Vol. 16. John Wiley & Sons.

[6] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., Apr. 2002. A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation 6 (2), 182–197.

[7] Dresner, K., Stone, P., 2004. Multiagent traffic management: A reservation-based intersection control mechanism. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2. IEEE Computer Society, pp. 530–537.

[8] Dresner, K., Stone, P., 2008. A multiagent approach to autonomous intersection management. Journal of artificial intelligence research, 591–656.

[9] Ferreira, J. C., Fonseca, C. M., Gaspar-Cunha, A., 2007. Methodology to select solutions from the pareto-optimal set: a comparative study. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation. ACM, pp. 789–796.

[10] Ferreira, M., Fernandes, R., Conceição, H., Viriyasitavat, W., Tonguz, O. K., 2010. Self-organized traffic control. In: Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking. ACM, pp. 85–90.

[11] Floreano, D., Mattiussi, C. M., 2008. Bio-inspired artificial intelligence theories, methods, and technologies. MIT Press, Cambridge, Mass.
URL `http://www.amazon.de/dp/0262062712`

[12] Holland, J., 1975. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press.
URL `https://books.google.no/books?id=JE5RAAAAMAAJ`

[13] Hunt, P., Robertson, D., Bretherton, R., Winton, R., 1981. Scoot-a traffic responsive method of coordinating signals. Tech. rep.

[14] Jiang, D., Delgrossi, L., 2008. Ieee 802.11 p: Towards an international standard for wireless access in vehicular environments. In: Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE. IEEE, pp. 2036–2040.

[15] Miettinen, K., 2012. Nonlinear multiobjective optimization. Vol. 12. Springer Science & Business Media.

[16] Schaffer, J. D., 1985. Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the 1st International Conference on Genetic Algorithms, Pittsburgh, PA, USA, July 1985. pp. 93–100.

[17] Sommer, C., Hagenauer, F., Dressler, F., 2014. A networking perspective on self-organizing intersection management. In: Internet of Things (WF-IoT), 2014 IEEE World Forum on. IEEE, pp. 230–234.

[18] TO, H. R., Barker, M. M., 2001. White paper european transport policy for 2010: time to decide.

[19] VISSIM, 2016. `http://vision-traffic.ptvgroup.com/en-uk/home/`, accessed: 2016-05-12.

[20] Willke, T. L., Tientrakool, P., Maxemchuk, N. F., 2009. A survey of inter-vehicle communication protocols and their applications. Communications Surveys & Tutorials, IEEE 11 (2), 3–20.

[21] Wu, J., Abbas-Turki, A., El Moudni, A., 2009. Discrete methods for urban intersection traffic controlling. In: Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th. IEEE, pp. 1–5.

[22] Wuthishuwong, C., Traechtler, A., Bruns, T., 2015. Safe trajectory planning for autonomous intersection management by using vehicle to infrastructure communication. EURASIP Journal on Wireless Communications and Networking 2015 (1), 1–12.

[23] Yan, F., Dridi, M., El Moudni, A., 2009. A branch and bound algorithm for new traffic signal control system of an isolated intersection. In: Computers & Industrial Engineering, 2009. CIE 2009. International Conference on. IEEE, pp. 999–1004.

[24] Yan, F., Dridi, M., El-Moudni, A., 2012. New vehicle sequencing algorithms with vehicular infrastructure integration for an isolated intersection. Telecommunication Systems 50 (4), 325–337.

[25] Yan, F., Dridi, M., El Moudni, A., 2013. An autonomous vehicle sequencing problem at intersections: A genetic algorithm approach. International Journal of Applied Mathematics and Computer Science 23 (1), 183–200.

[26] Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., Zhang, Q., 2011. Multiobjective evolutionary algorithms: A survey of the state of the art. Swarm and Evolutionary Computation 1 (1), 32–49.

Appendix A

Experiment 1 scenarios

Three sample scenarios: one low-traffic, one medium-traffic and one high-traffic. These were used in section 4.1. Figures start on the next page.

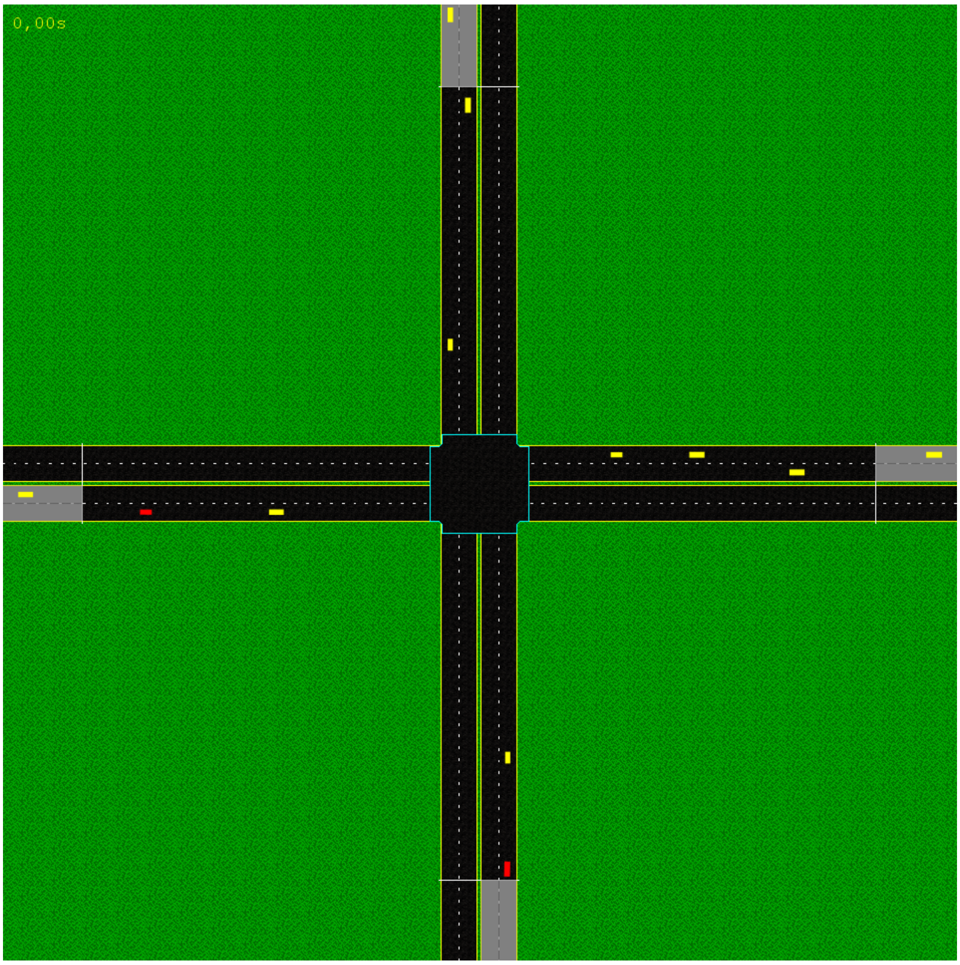

Figure A.1: A sample low-traffic scenario.

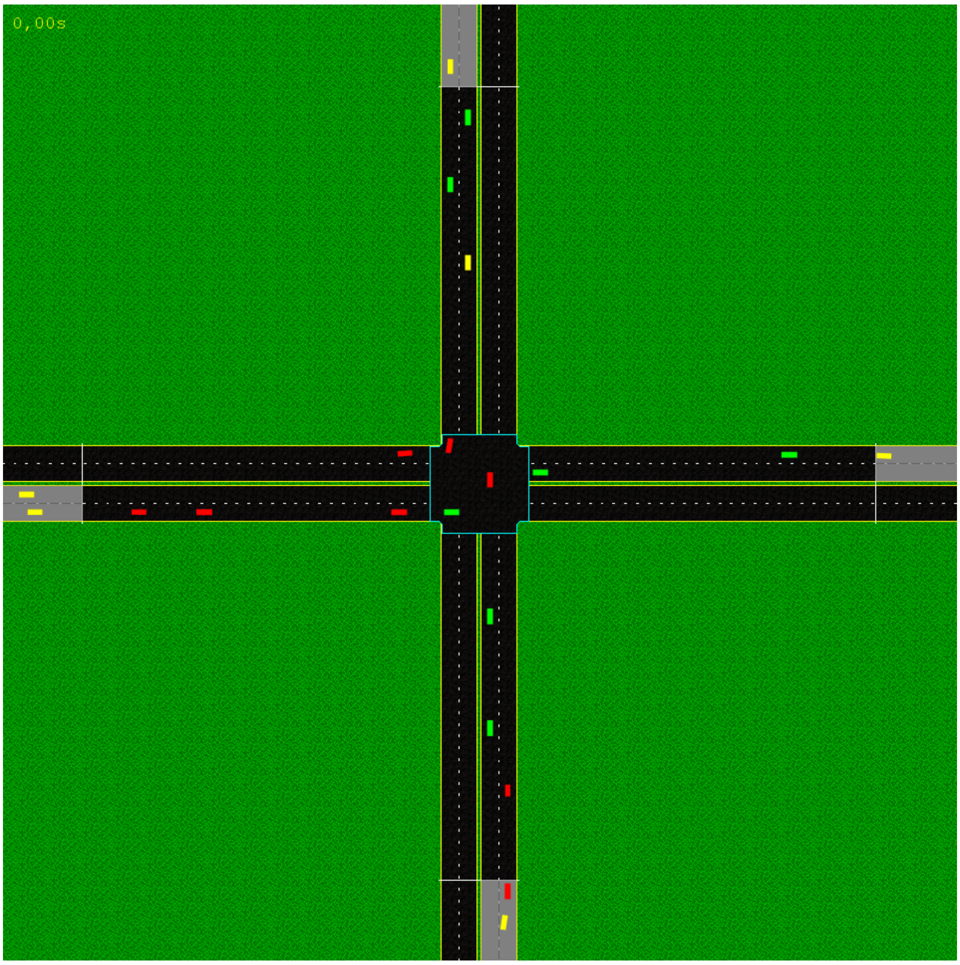

Figure A.2: A sample medium-traffic scenario.

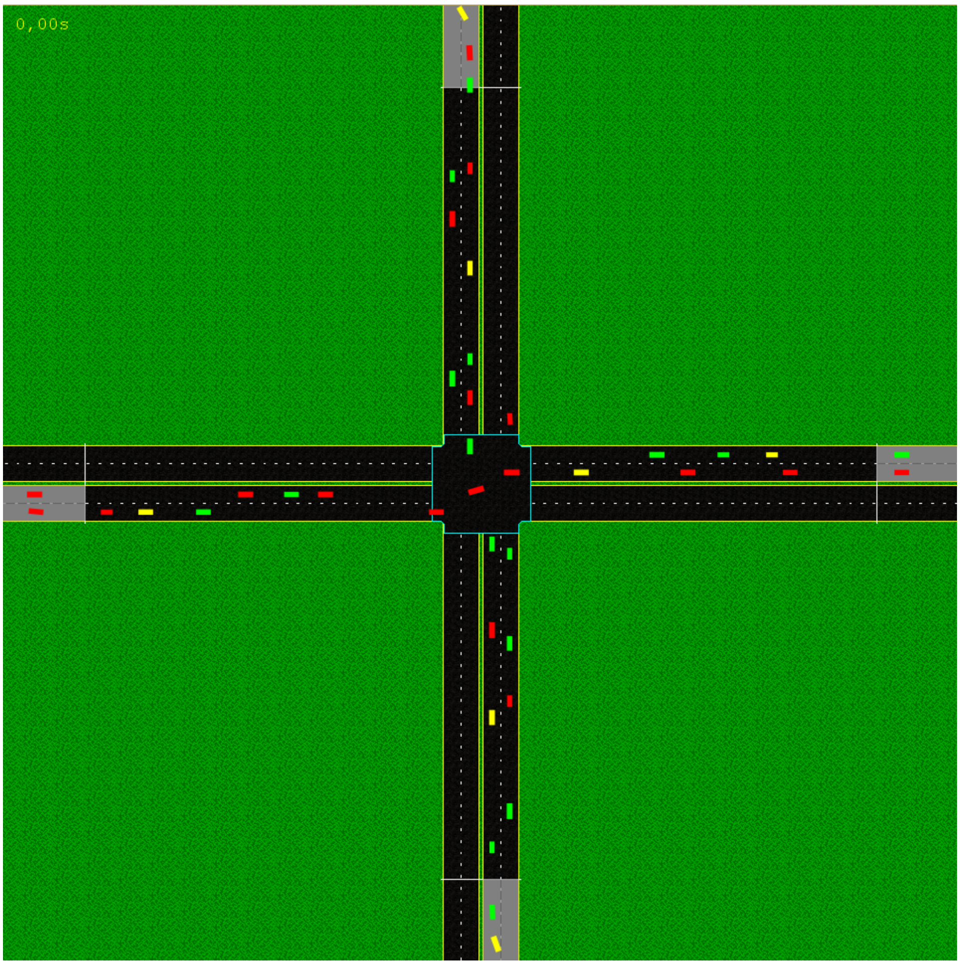

Figure A.3: A sample high-traffic scenario.

Appendix B

Experiment 5 scenarios

This appendix contains the scenarios that were used in section 4.5. Figures start on the next page.

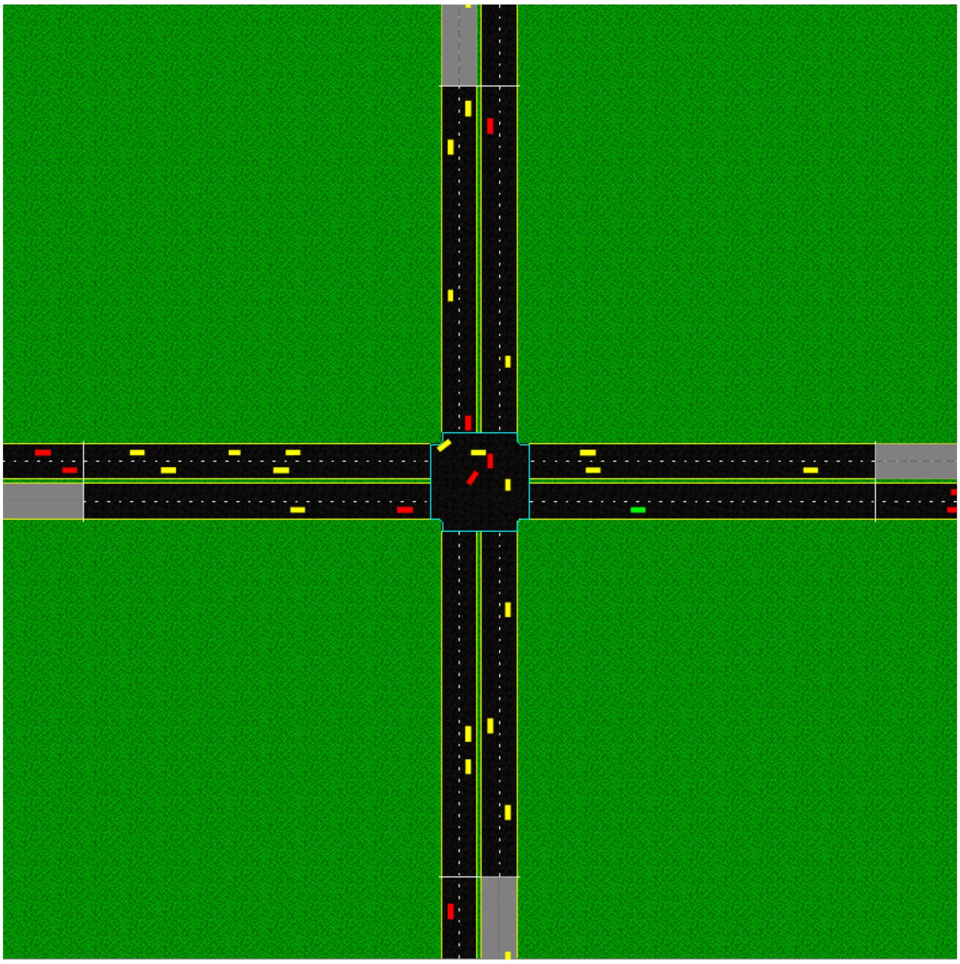

Figure B.1: The figure shows the simple scenario used in experiment 5. The scenario was generated with a spawn rate of 0.4. As there are few vehicles in the intersection and the trajectories are non-conflicting it is simple to evolve a solution without collisions where all objectives are close to optimal.

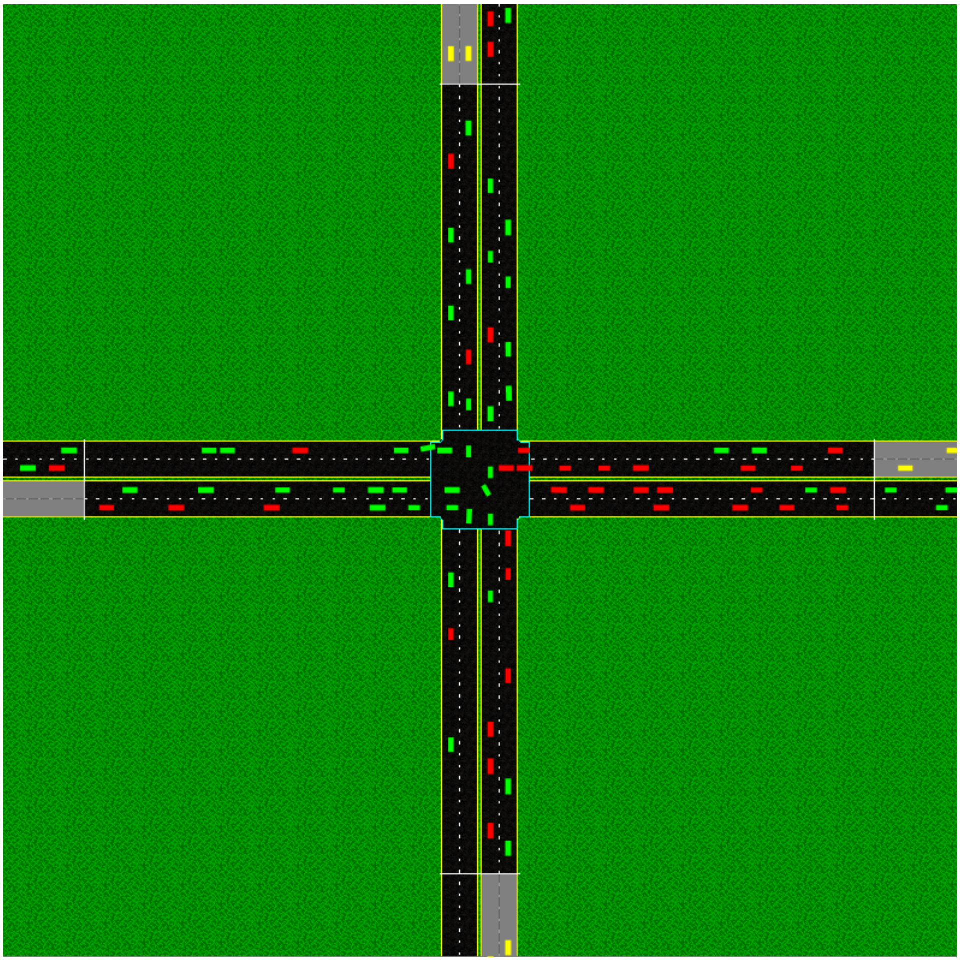

Figure B.2: The figure shows the first complex scenario used in experiment 5. The scenario was generated with a spawn rate of 0.8. As there are a lot of vehicles, especially in the inner intersection, the objectives are expected to conflict.

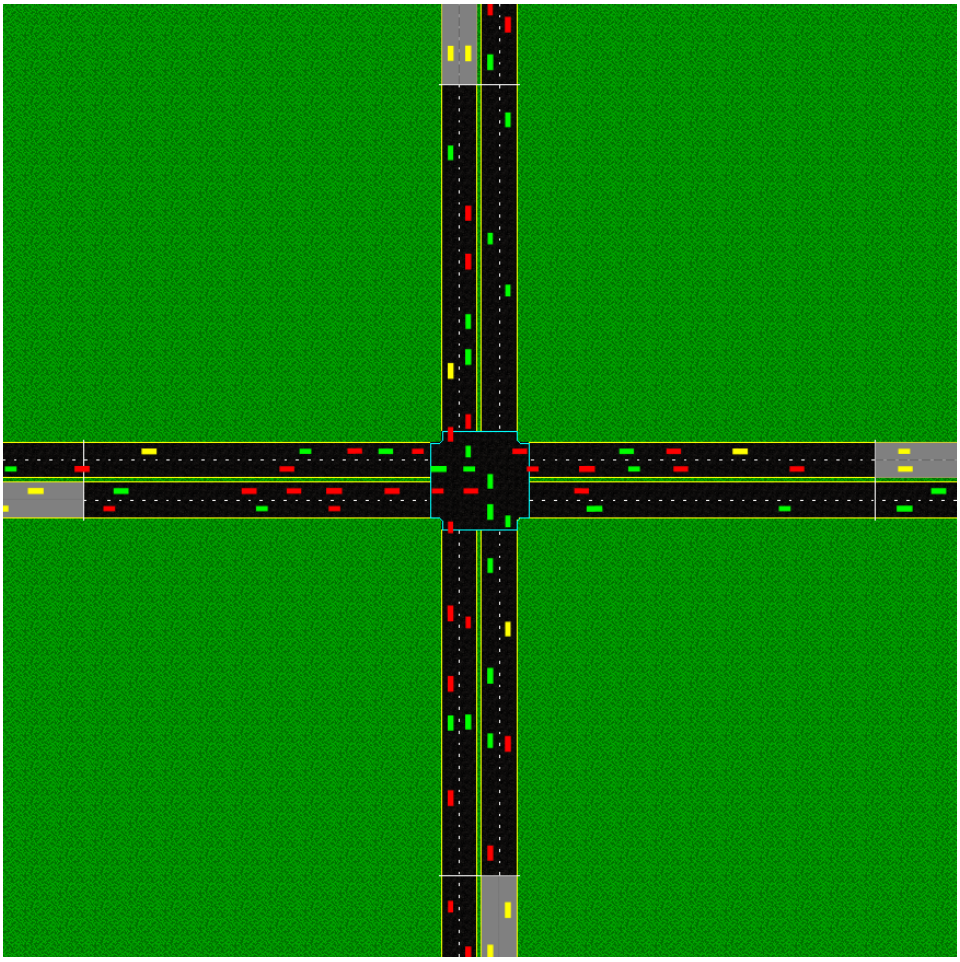

Figure B.3: The figure shows the first complex scenario used in experiment 5. The scenario was generated with a spawn rate of 0.9. As there are a lot of vehicles arriving in the intersection the objectives are expected to conflict.

Appendix C

A collision

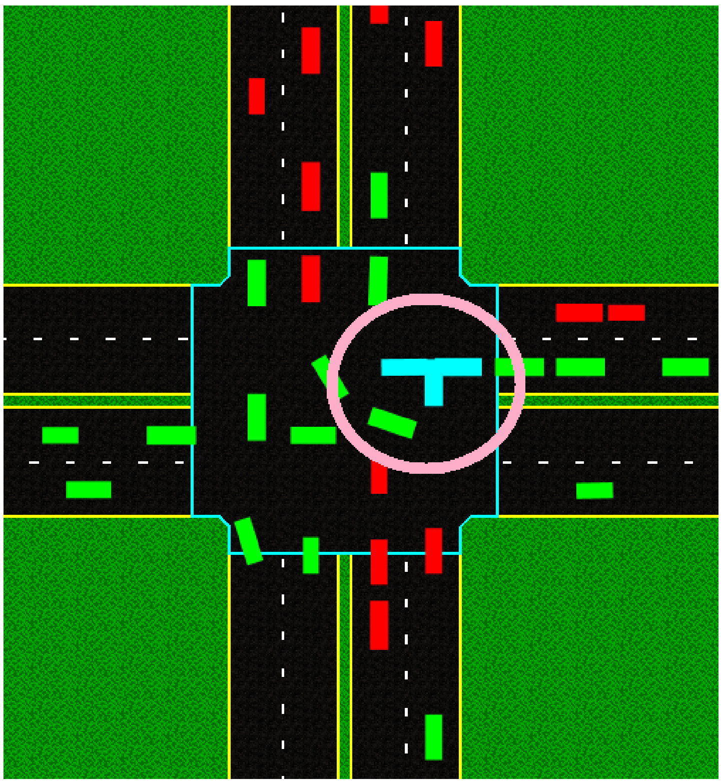

Figure C.1: A collisions occurring in the inner intersection (circled). The figure is taken from one run, with a spawn rate of 0.8, from the experiment in 4.2.

Appendix D

Running the software

Instructions for building or running are available in the *readme.txt* file. The system has only been tested on Windows 10, but should work on other operating systems. It requires Java 8 to run.