Dynamic Traffic Control of the Hovenring

Nomura, Satoshi

Award date:
2022

Department of Mechanical Engineering

Dynamics and Control Group

Dynamic Traffic Control of the Hovenring

4WC00 - Bachelor End Project
3rd and 4th Quartile 2021-2022

Report Number: DC 2022.068

Satoshi Nomura
1428152

Project Supervisor:
Dr.ir. A.A.J. Lefeber

Eindhoven, July 10, 2022

ABSTRACT

The main objective of this project is to develop a dynamic controller for the traffic in Hovenring. The controller is designed to be capable of serving traffic in a steady state while adapting to the change in the traffic intensity. This allows the traffic to be served with maximum efficiently with a single controller rather than having to switch between multiple controllers. The optimal cycle for a given situation of traffic intensity situation is created using the previous work [2]. Each intensity data of the day is used to create the optimal cycle and by identifying the differences between each optimal cycles, the dynamic controller was designed. For a given intensity level, the dynamic controller adapts by changing the service length for a given direction. It is capable of adjusting to any intensity level if the utilization of certain directions are below a certain value. The performance of the dynamic controller was measured by the average waiting time and was tested by comparing to the optimal cycle for the same intensity level. The created dynamic controller enables the intersection to be served at any time of the day with any intensity level, if the utilization conditions are met, eliminating the need to design numerous number of optimal controllers and switching between them to meet the service requirement.

Contents

List of symbols

Symbol	**Definition**	**Symbol Unit**
v	Velocity	m/s
s	Distance	m
a	Acceleration	m/s^2
λ	Arrival rate	cars/s
μ	Service rate	cars/s
$\min_{\text{green}}$	Minimum green time	s
T	Period	s
σ	Setup time	s
X	Buffer of all directions	cars
x	Buffer of one direction	cars

Chapter 1

Introduction

Traffic controllers are one of the most familiar controllers people encounter in their daily lives. A badly designed traffic controller can result in not only frustration but also can result in accidents and endangers the public. Every intersection is different as it is made to fit the location and designing an optimal controller is a challenge. In order to create the best use experience, optimization of intersection control has been a big topic. An optimised controller should minimise the waiting times and avoiding spillbacks without sacrificing the safety of the users.

Time based traffic controller is the conventional method to control a intersection. An optimal cycle is created based on the intersection data and used to serve the intersection. However, the way traffic reacts to a given controller is dependent on multiple factors that are constantly changing, such as the weather, drivers, and traffic intensity. With a slight change in one of the factors, the time based controller will no longer be able to operate with the highest efficiency. To combat this, multiple optimal controllers can be designed and switched depending on the situation. However, designing a controller suited for each and every situation is not only time consuming but is almost impossible.

Dynamic controller enables the adaptation to different situation to occur automatically. It will not only eliminate the need to create multiple controllers but also be able to keep serving the intersection with high efficiency every small change in the situation. In this project, one of the most important varying factor, the changing traffic intensity throughout the day is investigated. Using tooling developed in earlier work [2] the optimal periodic behaviour for a given traffic intensity is determined. From these optimal cycles, the controller is converted to a dynamic controller enabling it to adapt to different times of the day based on the work of [1]. Finally the controller is simplified and adapted in order to suite the changes in traffic intensity throughout the entire day. The created controller is tested and validated in order to assess its performance compared to the optimal periodic behaviour.

In this project, the Hovenring, a landmark intersection situated between Eindhoven and Veldhoven is investigated. The Hovenring is a major intersection with multilane traffic entering the intersection from all 4 sides. Moreover, it has completely separated the motor vehicles from the pedestrian and cyclists posing a unique setup of solely car oriented intersection. This project aims to create a dynamic traffic controller that is capable of adapting to the changes in the traffic intensity is created with the goal of reducing the average waiting time for each driver as much as possible while maintaining the safety of the drivers.

Chapter 2

Intersection Analysis

In order to design a controller suited for the Hovenring intersection, analysis of the intersection is crucial. It is necessary to calculate and outline the key variables such as conflicting directions and time required to clear the intersection. In this chapter, the key values are calculated through the use of data of inflow rate of cars for each direction provided by the city of Eindhoven and measurements made on Google maps.

2.1 Numbering the directions

The first step in this analysis was to identify and number the directions of the in flowing traffic. The numbering was done following the standard system of numbering which can be found in the Appendix A.2. The numbered direction along with the corresponding traffic paths can be seen in the Figure 2.1.

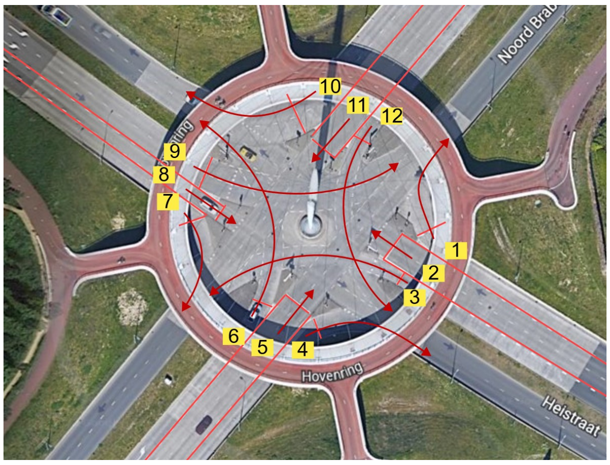


Figure 2.1: Annotated Hovenring Intersection

2.2 Identifying conflicting directions

One of the most important features of the traffic controller is to avoid accidents. These accidents occur between cars that are travelling in a conflicting directions. By not serving the set of conflicting directions simultaneously, accidents can be avoided. Furthermore, when switching from one of the conflicting directions to the other, it is necessary to make sure that the car that was being served has passed the point of possible collision before the car served by the second direction arrives at the same point. The necessary time interval between these two service is known as clearance times or setup time σ and in this report it is referred to the latter. In this step, the possible combination of conflicting directions are identified and the setup time required to avoid an accident is calculated.

Using the map of the Hovenring, all of the possible combination of conflicts were identified. These were mapped in a plot as shown in Figure 2.2 where nodes represent the different directions and the connecting lines connect the possible collision combination between two directions. As seen by the arrows, each combinations conflicts in transitions in both directions and there are in total 28 conflicting pairs resulting in 56 conflicting transitions in service.

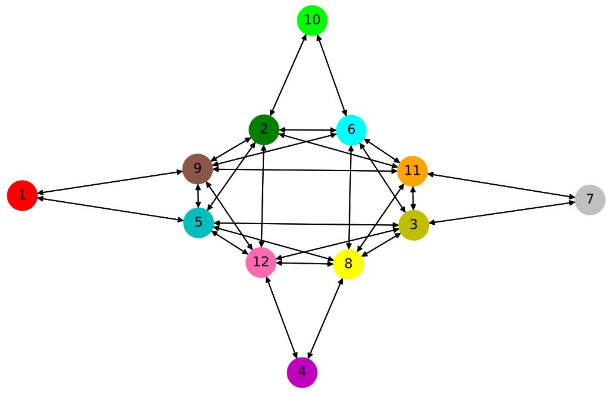


Figure 2.2: Conflict Graph Hovenring

2.3 Conflict matrix

After finding the combinations of the conflicting directions, the points of intersection between the two path were marked on the map and the distance from the stop line to the point of collision were measured for both direction. It was assumed that the acceleration of the car is estimated as $4m/s$ and the velocity of the car to be $40km/h$. Using (2.1) the required setup time for transition from direction i to j were calculated. This formula subtracts the time it takes for the car from direction j to reach the point of conflict from the time it takes for the car in the direction i to clear the intersection.

$$\sigma_{\mathrm{ij}} = \frac{s_{\mathrm{i}}}{v_{\mathrm{i}}} - \frac{1}{2} \cdot \left(\sqrt{\frac{2 \cdot s_{\mathrm{j}}}{a_{\mathrm{j}}}} + \frac{s_{\mathrm{j}}}{v_{\mathrm{j}}} \right) + 1 \tag{2.1}$$

The time for direction j has been calculated as the average time between the car that accelerates from stationary state to the intersection point and the car approaching at speed to the intersection point as the car was assumed to have slowed down approaching the intersection but it may not be completely stationary when the light turns green. Therefore, the average time between the completely stationary and approaching at high speed was assumed to be a good estimate. In addition to this, additional 1 second was added and the time for direction i was rounded up and j was rounded down for extra safety margins. After the calculations, if the negative value were obtained, the setup time for that conflict pair was assumed to be 0 seconds. Due to the extra margins the error that may exist in the measurements or the assumptions are accounted for and therefore the controller designed according to this calculation is very unlikely to cause any accidents.

The calculated results has been formed into a matrix as shown in the Table 2.1. The position on the matrix represent the pair of conflicting directions and the values represent the required setup time. The conflict only occurs in the combination that has been coloured in green. The number of conflicts on the left hand side of the matrix must be equal to that of the right hand side as the conflict exists in switching in either order. Typically the sum of setup times on the left hand side and the right hand side is equal. However, for the Hovenring it is not exactly identical. This difference is possibly due to the rounding that has been done in the process of calculation.

Table 2.1: Conflict matrix of Hovenring

From \ To	1	2	3	4	5	6	7	8	9	10	11	12
1	▨				0				0			
2		▨			0	2			2	5	3	2
3			▨		2	3	5	3			2	1
4				▨				0				0
5	4	4	2		▨			0	2			1
6		2	0			▨		2	3	5	3	
7			0				▨				0	
8			1	5	3	2		▨			0	3
9	5	3			3	1			▨		2	3
10		0				0				▨		
11		0	2			1	4	4	2		▨	
12		2	3	5	3			2	0			▨

In this chapter, the Hovenring intersection was analysed and the traffic paths and the possible collision locations were identified. The required setup time for transition of service between two directions were calculated which is a vital information in the creation of the traffic controller. The calculated setup time along with some information about the intersection are fed into the code from the past work [2] in the next step to create the optimal cycles.

Chapter 3

Optimised Cycle

The analysis of the intersection provided the information necessary to shift towards the next step. In this chapter, using the conflict matrix created, the optimal cycle for every times of the day are created which is then used as a template to the steps of developing the dynamic controller.

3.1 Creating the Hourly Optimal Cycle

In order to develop and assess the performance of a controller, capable of serving the entire day with fluctuating inflow of traffic, it is necessary to have an optimal cycle for different traffic inflow to use as a development base and to compare the performance to. A code developed by S.T.G. Fleuren [2], along with the hourly traffic intensity data provided by the city of Eindhoven (Appendix A.1), and the created conflict matrix were used in order to create optimal traffic cycle for the intersection for each hour of the day. Furthermore, the minimum green time was assumed to be 4 seconds, requiring the green light to be at least 4 seconds long. The resulting cycle for the different times of the day were analysed as explained in the following.

3.2 Analysis of the optimal cycle

The code produces a list of directions and their corresponding start and stop time of serving. The results were utilised to outline multiple characteristics of the optimal cycle, which determines the formation of the final controller. Furthermore, the results were represented in either green or red light state for each given direction. The amber or yellow light was considered as a part of green light period and the combination of green and amber light periods is known as the effective green time. In this report, the term green light time is used to represent the effective green time. A plot was outputted by the code which was then annotated to visualize and facilitate the analysis done as shown in the Figure 3.1.

3.2.1 Relation between the served directions

The analysis of served directions were done next. This step aims to analyse and identify the pattern in the order of the served traffic directions and the relationship between each directions served. The conflict matrix created in subsection 2.3 was used in this analysis. The conflicting directions cannot be served simultaneously and when switching from one to the other it is required to have a set up time. These factors act as a constraint when optimal cycles were produced. However, not all of the conflicting combinations were utilized as constraints. The ones that had been used has been identified as active constraints. The active constraints determines the order and relates the service of multiple directions.

In order to identify the active constraints, the time interval between the services of a pair of conflicting directions were calculated using the end of service time for one direction and the start of service time of the other. The value is then compared to the setup time stated in the conflict matrix and if they are similar, then the conflicting pair can be considered as an active constraint. The identified active constraints can be represented on the plot of the period as shown by the purple and blue lines on the Figure 3.1. By following the lines of active constraints and green light period of different directions, it should be able to make a loop around the cycle. Furthermore, every start and end of green period should be connected to an active constraint. In another words, starting from service of any direction, following the active constraints and the green light period should make a loop back to the starting point.

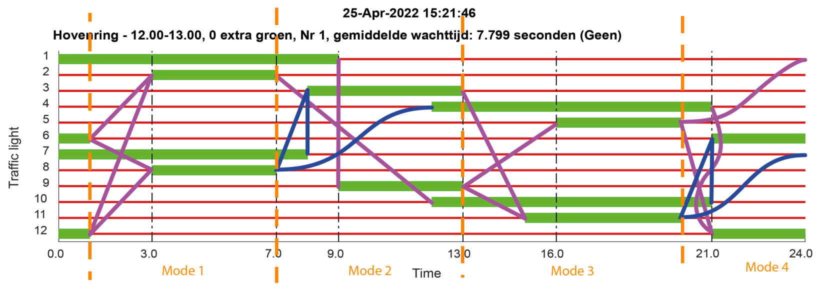


Figure 3.1: Hovenring optimal cycle between 12h and 13h

The created plots such as the Figure 3.1, confirms that all of the starts and ends of the green period are connected to an active constraint. It is also possible to identify two sets of independent constraints system present at certain points of the cycle. Two colours, purple and blue lines are used to visualise these two paths. Due to these two sets of simultaneous transitions being independent from each other, at certain time of the day, the transition of services shifts independent to each other. An example of this shifting can be seen in the Figure 3.2.

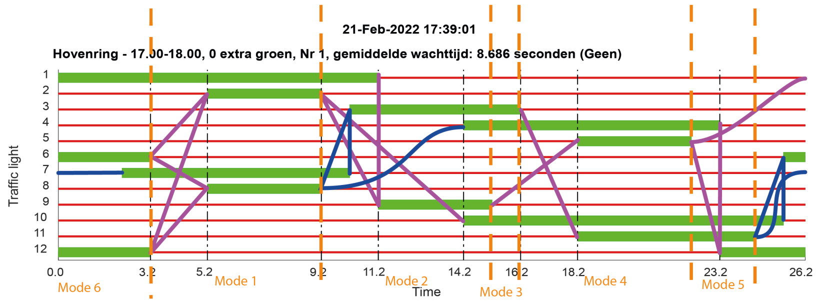


Figure 3.2: Hovenring optimal cycle between 17h and 18h

Investigating all of the different time slots it was identified that most of them had the same number of active constraints. However due to some shifts mentioned above, there were certain places where the active constraints changed depending on the time of the day. Furthermore, the period of the cycle also varied between 24 seconds and 26 seconds. Even though those cycles have slight differences in the active constraints, the overall pattern of the paths stayed the same connecting the same directions.

3.2.2 Identifying modes

Now that the relation between the served direction is found, the next step is to identify modes. Modes are period of time when multiple transitions occur, followed by service of those directions. In other words, modes are time intervals between groups of transitions that occur simultaneously. It forms a building block that make up the entire traffic cycle. Within a cycle, multiple modes exist and the size and complexity of the intersection determines the total number of modes required. By identifying the modes during different times of the day, it is possible to identify similarities and discover opportunities to simplify the traffic controller. In the example plots (Figure 3.1 and Figure 3.2), the modes were represented by orange lines.

The modes were easily located when the transition of service direction occurs simultaneously. For example for the direction 6 and 12 transitioning to directions 2 and 8 always initiated at the same time. However, this is not always the case and in some cases, not all of the transitions may occur at the same time. In this situation the active constraint at that point; representing the transition, may share a starting or ending node with another constraint. A group is created by the occurring transitions sharing the nodes and the earliest transitions starting time of that group was taken to be the point mode transition. By dividing the entire cycle using this rule, the modes are created as shown on the Figure 3.1 and Figure 3.2 in orange.

3.2.3 Switching conditions

The length of the mode is determined by the length of service for a given direction. Each mode has a key direction that determines the length of the mode. In order to determine the conditions for the switching to occur, it is necessary to confirm that the ideal cycle is operating under a clearing policy (each direction served until the buffer is empty).

In order to confirm this, a graph of the evolution of buffer throughout the cycle was created (Figure 3.3) with the assumption that the cycle operates with the clearing policy. When operating, the buffer is required to be stable. From the graph it was observed that the buffer count at the start and end of the period is identical indicating that the cycle is stable and therefore it can be confirmed that the optimal cycle operates with the clearing policy.

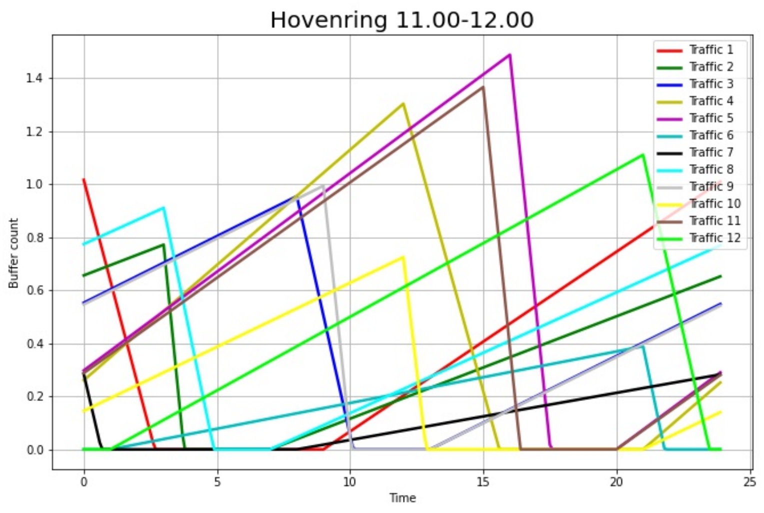


Figure 3.3: Hovenring Buffer evolution (12h and 13h)

As the traffic controller operates under a clearing policy, the buffer has to be emptied for the serviced direction before the traffic light switches to red. However, when the traffic intensity is low, this results in the traffic light switching immediately after starting service. This is avoided by the minimum green time set during the formation of the optimal cycle. So there are two possibilities in order to determine the duration of the service.

- The duration will be equal to the time it takes to empty the buffer, if it takes longer than the minimum green time
- The duration will be equal to the minimum green time, if the buffer empties before the minimum green time is reached.

The key directions for each mode can be determined by checking the length of green period for each direction that is turning red at the end of the mode. If the length is equal to the minimum green time, then that direction can be identified as the key direction. Detecting the other condition for the triggering of transmission is more difficult. In order to detect the correct key direction, the buffer just before the end of the mode has to be analysed. A direction that empties shortly before the switching must be identified. The clearing policy operates in a way where the traffic light for a given direction turns red shortly after the buffer empties. So, if all of the directions that is turning red at a given mode has duration longer than minimum green time, the direction with the least time of empty buffer at the end becomes the key direction.

With the above knowledge, key direction for a given mode can be identified and the switching condition for the key direction can be determined. The optimal cycle throughout the day can be analysed in this way and if the key direction for a given direction remains equal throughout the day, then and only then can the mode characteristic for that period be considered uniform through out the day.

3.2.4 Average waiting time

From the optimal time cycles, the average waiting times for each direction were also calculated. The average waiting time for the all of the direction for a given cycle was provided as an output along with the optimal cycle but the individual waiting time was not. The average time was computed using the knowledge of inflow of cars to each direction, rate of outflow of cars exiting the intersection and the service time provided by the optimal cycle. The average time for each directions were weight by the intensity and the average of the outcome has been confirmed to be identical to that outputted by the code. Example of the calculated average data can be found in the Table 3.1. As expected, the increase in traffic intensity results in longer waiting time as seen in the difference in waiting time between mid-day and around the afternoon rush hour.

The waiting time is a good criteria to be used to determine the quality of the controller as, controller with the shortest waiting time has the highest efficiency. The optimal cycle for a given traffic intensity should have the lowest average waiting time possible. The average waiting time of the optimal cycle is compared to that of the designed controller and it is possible to assess the quality of the designed controller.

Table 3.1: Average waiting time for a given direction

		Hour of the day	
		12.00-13.00	17.00-18.00
Service direction	1	5.644967	5.17816
	2	8.679274	10.34696
	3	8.647547	10.11834
	4	6.026059	7.419499
	5	9.179252	10.28393
	6	8.586766	10.06104
	7	5.508106	7.128322
	8	9.182003	10.82275
	9	8.980945	10.04919
	10	5.062901	4.904842
	11	8.184787	8.796361
	12	9.66963	10.00221
Mean waiting time		7.798597	8.685955
Mean waiting time (Code)		7.798665	8.685994

3.3 Optimised cycle overview

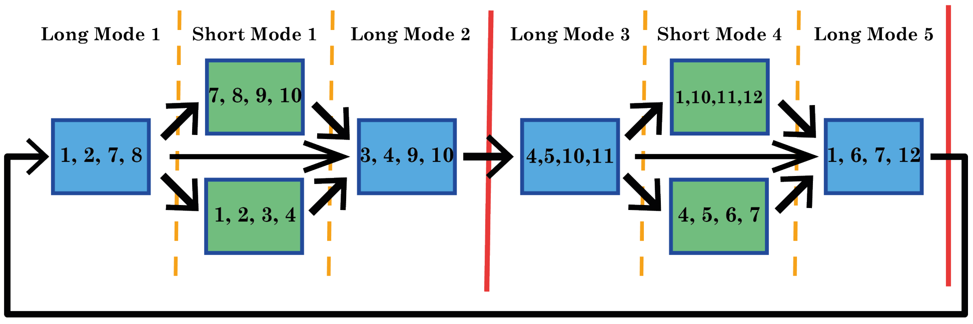


Figure 3.4: Hovenring Mode and direction mapping

The hourly optimal cycle were analysed in this chapter. Looking at all of the different times of the day, the modes and the served directions can be expressed as shown in the Figure 3.4. The mapping represents the possible directions served in each mode and the arrows represents the possible transition from one mode to another. Most of the time slots experienced only the 4 long modes bypassing all of the short modes. In some cases where the traffic intensity were higher, such as during the rush hours, a mode lasting a short time appeared. These short modes perform as a transition phase from one long mode to another.

The mapping also represents the number of changes in the served direction. Where there exist a solid red line between the mode transition, the long modes transitions into another long mode. During transition of mode occurring through the dotted orange lines, the short modes are involved and therefore it represents that the mode can also be skipped. In this intersection every mode services 4 directions and each mode may have either 2 or 4 transitioning directions depending on if the short mode is skipped or not.

There was one exception to the mapping represented. During the hour 17h-18h another additional mode appears in between long mode 2 and 3 (represented as mode 3 on Figure 3.2). This is an exception as in this optimal cycle, the service towards the direction 3 is prioritised over the service to the direction 11. This resulted in a transition from direction 3 to 11 to maximise the service to direction 3. This transition is independent to the state of the buffer and results in the optimal cycle at the hour 17h-18h to not fit that of the concluded mapping.

The conclusions and findings obtained from the optimal cycle is used in the following chapter to develop the dynamic controller for the Hovenring.

Chapter 4

Defining the Policy

The analysis of the optimal cycle has revealed some of the characteristics of the Hovering intersection. In this chapter a controller policy for the Hovering intersection with the capability of controlling the fluctuating traffic inflow, is developed utilizing the findings. The defined policy is compared to each optimal cycles and its performance is analysed. The policy aims to simplify and minimise the number of different controllers required in order to adapt to the changing situation throughout the day as possible without sacrificing the performance.

4.1 Simplification of the policy of the optimal cycle

The first step in defining the policy is to investigate the possibilities of simplification in the policy of the optimised cycle. Simplified policy is then be used as a foundation for defining a universal policy.

The findings from subsection 3.3 identified 6 possible modes, each serving different sets of direction. Furthermore, an exception to the mapping was also stated. First simplification to the policy is to ignore this exception. As it was only present in one hour through out the entire day it is assumed that the occurrence is less likely to have a significant effect on the overall performance of the controller.

The next simplification is to disregard the short modes. There were mainly two types of modes in the stated policy, the long modes and the short modes. As the short modes were often times less than 1 second, neglecting it is expected to only results in a minor impact on the performance.

The short modes were the product of shifts in the ending timing of the direction in the long modes. There were two pairs of served direction that transitioned during these short modes and the two pairs interchanged in the shifting order. Following this, the two pairs transition is assumed to occur at once when the service of both directions exceeds the minimum green time and the buffer has been emptied. This should be the optimal solution as it results in a reduction of waiting time in the ending service direction and increase the one or two direction served in the next mode. The net additional waiting time should be insignificant and the increase in the overall average waiting time at the intersection is only minor.

4.2 Policy definition

Following the assumption a policy was created.

Mode 1:
The direction 1 and 7 are already being served from the prior mode. The mode starts with the setup time prior to starting the services for directions 2 and 8. At the time when the direction 2 and 8 are both empty the switching condition is met. Service to directions 2 and 8 are stopped and the mode moves on to the second mode while keeping the service towards direction 1 and 7. In the second mode, when the service for directions 1 and 7 are stopped, the buffer contents in the direction has to empty. In order to confirm this situation, additional requirement is set. In addition to both buffers for direction 2 and 8 being empty, the buffer for directions 1 and 7 has to have contain less than the amount clear-able in the mode 2.

Serves directions 1,2,7,8

- Setup duration
  - $\sigma_{6,2} = \sigma_{12,8}$ while serving direction 1 and 7 at maximum rate or arrival rate.
- Serve x_2 and x_8 at maximum rate for:
  - If $x_1 < \sigma_{2,9} \cdot (\mu_1 - \lambda_1)$ and $x_7 < \sigma_{8,3} \cdot (\mu_7 - \lambda_7)$:

    Serve for $\min_{\text{green}}$ if $\min_{\text{green}} > \frac{x_2+\lambda_2\cdot\sigma_{6,2}}{\mu_2-\lambda_2}$ or $\frac{x_8+\lambda_8\cdot\sigma_{12,8}}{\mu_8-\lambda_8}$

    Serve for $\frac{x_2+\lambda_2\cdot\sigma_{6,2}}{\mu_2-\lambda_2}$ if $\frac{x_2+\lambda_2\cdot\sigma_{6,2}}{\mu_2-\lambda_2} > \min_{\text{green}}$ or $\frac{x_8+\lambda_8\cdot\sigma_{12,8}}{\mu_8-\lambda_8}$

    Serve for $\frac{x_8+\lambda_8\cdot\sigma_{12,8}}{\mu_8-\lambda_8}$ if $\frac{x_8+\lambda_8\cdot\sigma_{12,8}}{\mu_8-\lambda_8} > \min_{\text{green}}$ or $\frac{x_2+\lambda_2\cdot\sigma_{6,2}}{\mu_2-\lambda_2}$
  - If $x_1 > \sigma_{2,9} \cdot (\mu_1 - \lambda_1)$ when $x_2 = 0$ and $x_8 = 0$ and $x_7 < \sigma_{8,3} \cdot (\mu_7 - \lambda_7)$:

    Serve for $\frac{x_1-\sigma_{2,9}\cdot(\mu_1-\lambda_1)}{\mu_1-\lambda_1}$
  - If $x_7 < \sigma_{8,3} \cdot (\mu_7 - \lambda_7)$ when $x_2 = 0$ and $x_8 = 0$ and $x_1 < \sigma_{2,9} \cdot (\mu_1 - \lambda_1)$:

    Serve for $\frac{x_7-\sigma_{8,3}\cdot(\mu_7-\lambda_7)}{\mu_7-\lambda_7}$
- Switch to Mode 2

Mode 2:
In this mode, the mode initiate with the setup time for directions 3 and 9. At the start of the mode, direction 1 and 7 are still being served. Due to the additional requirement in mode 1, the directions are capable of being emptied within the setup time for direction 3 and 9. When the directions being service, the service to directions 1 and 7 are stopped and service for directions 3 and 9 are initiated. Later in the mode, after the setup time, directions 4 and 10 are started. These directions are served continuously into the next mode. The mode 2 transitions to mode 3 when buffers in both directions 3 and 9 are emptied.

Served directions 1,3,4,7,9,10
Start serves directions 3,4,9,10

- Setup duration
  - $\sigma_{8,4}$ and $\sigma_{2,10}$ followed by serving x_4 and x_{10} at maximum rate or arrival rate
  - $\sigma_{2,9}$ while serving direction 1 at maximum rate or at arrival rate
  - $\sigma_{8,3}$ while serving direction 7 at maximum rate or at arrival rate
- Serve x_3 and x_9
  - If $\min_{\text{green}} > \frac{x_9+\lambda_9\cdot\sigma_{2,9}}{\mu_9-\lambda_9}$ or $\left(\frac{x_3+\lambda_3\cdot\sigma_{8,3}}{\mu_3-\lambda_3} + \sigma_{8,3} - \sigma_{2,9}\right)$

    Serve x_9 for $\min_{\text{green}}$

    Serve x_3 for $\min_{\text{green}} - \sigma_{8,3} + \sigma_{2,9}$
  - If $\frac{x_9+\lambda_9\cdot\sigma_{2,9}}{\mu_9-\lambda_9} > \min_{\text{green}}$ or $\left(\frac{x_3+\lambda_3\cdot\sigma_{8,3}}{\mu_3-\lambda_3} + \sigma_{8,3} - \sigma_{2,9}\right)$

    Serve x_9 for $\frac{x_9+\lambda_9\cdot\sigma_{2,9}}{\mu_9-\lambda_9}$

    Serve x_3 for $\frac{x_9+\lambda_9\cdot\sigma_{2,9}}{\mu_9-\lambda_9} - \sigma_{8,3} + \sigma_{2,9}$
  - If $\frac{x_3+\lambda_3\cdot\sigma_{8,3}}{\mu_3-\lambda_3} > \left(\min_{\text{green}} \text{ or } \frac{x_9+\lambda_9\cdot\sigma_{2,9}}{\mu_9-\lambda_9}\right) - \sigma_{8,3} + \sigma_{2,9}$

    Serve x_9 for $\frac{x_3+\lambda_3\cdot\sigma_{8,3}}{\mu_3-\lambda_3} + \sigma_{8,3} - \sigma_{2,9}$

    Serve x_3 for $\frac{x_3+\lambda_3\cdot\sigma_{8,3}}{\mu_3-\lambda_3}$
- Switch to Mode 3

Mode 3:
Prior to this mode, the directions 4 and 10 already begins their service. The mode starts serving direction 5 and 11 until empty after the setup time. Directions 5 and 11 becomes the key in switching to mode 4. Additional constraints similar to the one in mode 1 also is applied to direction 4 and 10 as the buffer must be emptied in the following mode.

Serves directions 4,5,10,11

- Setup duration
  - $\sigma_{3,11}$ and $\sigma_{9,5}$ while serving direction 4 and 10 at maximum rate or arrival rate
- Serve x_5 and x_{11}
  - If $x_4 < \sigma_{11,6} \cdot (\mu_4 - \lambda_4)$ and $x_{10} < \sigma_{5,12} \cdot (\mu_{10} - \lambda_{10})$
    - If $\min_{\text{green}} > \frac{x_5+\lambda_5 \cdot \sigma_{9,5}}{\mu_5-\lambda_5}$ or $\left(\frac{x_{11}+\lambda_{11} \cdot \sigma_{3,11}}{\mu_{11}-\lambda_{11}} + \sigma_{3,11} - \sigma_{9,5}\right)$

      Serve x_9 for $\min_{\text{green}}$

      Serve $x_1 1$ for $\min_{\text{green}} - \sigma_{3,11} + \sigma_{9,5}$
    - $\frac{x_5+\lambda_5 \cdot \sigma_{9,5}}{\mu_5-\lambda_5} > \min_{\text{green}}$ or $\left(\frac{x_{11}+\lambda_{11} \cdot \sigma_{3,11}}{\mu_{11}-\lambda_{11}} + \sigma_{3,11} - \sigma_{9,5}\right)$

      Serve x_9 for $\frac{x_5+\lambda_5 \cdot \sigma_{9,5}}{\mu_5-\lambda_5}$

      Serve $x_1 1$ for $\frac{x_5+\lambda_5 \cdot \sigma_{9,5}}{\mu_5-\lambda_5} - \sigma_{3,11} + \sigma_{9,5}$
    - $\frac{x_{11}+\lambda_{11} \cdot \sigma_{3,11}}{\mu_{11}-\lambda_{11}} > \left(\min_{\text{green}} \text{or} \frac{x_5+\lambda_5 \cdot \sigma_{9,5}}{\mu_5-\lambda_5}\right) - \sigma_{3,11} + \sigma_{9,5}$

      Serve x_9 for $\frac{x_{11}+\lambda_{11} \cdot \sigma_{3,11}}{\mu_{11}-\lambda_{11}} + \sigma_{3,11} - \sigma_{9,5}$

      Serve $x_1 1$ for $\frac{x_{11}+\lambda_{11} \cdot \sigma_{3,11}}{\mu_{11}-\lambda_{11}}$
  - If $x_{10} < \sigma_{5,12} \cdot (\mu_{10} - \lambda_{10})$ when $x_9 = 0$ and $x_{11} = 0$ and $x_4 < \sigma_{11,6} \cdot (\mu_4 - \lambda_4)$:

    Serve for $\frac{x_{10}-\sigma_{5,12} \cdot (\mu_{10}-\lambda_{10})}{\mu_{10}-\lambda_{10}}$
  - If $x_4 < \sigma_{11,6} \cdot (\mu_4 - \lambda_4)$ when $x_9 = 0$ and $x_{11} = 0$ and $x_{10} < \sigma_{5,12} \cdot (\mu_{10} - \lambda_{10})$:

    Serve for $\frac{x_4-\sigma_{11,6} \cdot (\mu_4-\lambda_4)}{\mu_4-\lambda_4}$
- Switch to Mode 4

Mode 4:
The mode start with direction 4 and 10 already in service. The service is terminated after the setup time for directions 6 and 12 and is replaced by the service towards these directions. During the mode, direction 1 and 7 initiates service and the mode transitions back to mode 1 when the directions 6 and 12 are both empty.

Serves directions 1,6,7,12

- Setup duration
  - $\sigma_{5,1}$ and $\sigma_{11,7}$ followed by serving x_1 and x_7 at maximum rate or arrival rate
  - $\sigma_{5,12} = \sigma_{11,6}$, while serving direction 4 and 10 at maximum rate or at arrival rate
- Serve x_6 and x_{12}
  - If $\min_{\text{green}} > \frac{x_6+\lambda_6 \cdot \sigma_{11,6}}{\mu_6-\lambda_6}$ or $\frac{x_{12}+\lambda_{12} \cdot \sigma_{4,12}}{\mu_{12}-\lambda_{12}}$

    Serve x_6 for $\min_{\text{green}}$

    Serve $x_1 2$ for $\min_{\text{green}}$
  - If $\frac{x_6+\lambda_6 \cdot \sigma_{11,6}}{\mu_6-\lambda_6} > \min_{\text{green}}$ or $\frac{x_{12}+\lambda_{12} \cdot \sigma_{4,12}}{\mu_{12}-\lambda_{12}}$

Serve x_6 for $\frac{x_6+\lambda_6\cdot\sigma_{11,6}}{\mu_6-\lambda_6}$

Serve x_12 for $\frac{x_6+\lambda_6\cdot\sigma_{11,6}}{\mu_6-\lambda_6}$

– If $\frac{x_{12}+\lambda_{12}\cdot\sigma_{4,12}}{\mu_{12}-\lambda_{12}} > \min_{\text{green}}$ or $\frac{x_6+\lambda_6\cdot\sigma_{11,6}}{\mu_6-\lambda_6}$

Serve x_6 for $\frac{x_{12}+\lambda_{12}\cdot\sigma_{4,12}}{\mu_{12}-\lambda_{12}}$

Serve x_12 for $\frac{x_{12}+\lambda_{12}\cdot\sigma_{4,12}}{\mu_{12}-\lambda_{12}}$

- Switch to Mode 1

In this chapter the policy for the dynamic traffic controller was defined from the optimal hourly cycles. Key aspects of the optimal cycles were identified and utilised in the dynamic controller. Number of assumptions were made during this process and in the next chapter the created policy is analysed and vilified to operate with similar performance to the optimal cycle.

Chapter 5

Analysis of the Policy

In the chapters above, the policy for the dynamic controller of the Hovenring was defined according to the findings from the analysis of the optimal fixed time controller. In this chapter the policy created is analysed and validated. The defined policy requires to be capable of serving every direction under a stable condition, meaning that the buffer count at the end of the cycle should be equal or smaller than the buffer count at the start of the cycle. By the means of simulation and mathematical theory based on the paper by V. Feoktistova [1], the stability is confirmed and the limitations of the designed controller are outlined.

5.1 Simulation

With the purpose of confirming the assumptions made during the creation of the policy, a code was created according to the policy. Different conditions were coded using the If statements and loops to loop for a given time period. The loop was done so that at every time interval, the buffer contents for all the directions were calculated and recorded. Each directions were assigned either green or red situation and depending on the modes, the directions defined as green were changed. The buffer contents were monitored and when a switching condition defined by if statement was met, the mode were switched to the next mode. The code can be found in Appendix B.

It is assumed that the policy establishes stability after a certain cycles. Therefore, the created code simulates multiple periods and uses the last cycle of traffic to compute the average waiting time. For each of the different traffic intensity during the different times of the day, calculation for the average waiting times for individual directions were performed. Furthermore, using the method in subsubsection 3.2.4, the different directions were weighed and the overall average waiting times were calculated.

5.1.1 Simulation results

As expected, the simulation showed that the controller were stable under each of the provided intensity data for Hovenring. Furthermore, the outcome cycle closely resemble that of the optimal cycle. This was expected as the dynamic controller was designed based on the optimal cycles.

The simulation has successfully validated the controller stability and its capability to recreate a similar cycle to the optimal cycle. The stability can be spotted by the either ends of the buffer evolution graph shown in Figure 5.2. The dotted vertical lines represents where the cycle begins and ends. Looking at different directions denoted by the different colours, it is able to see that the buffer contents at the location of the first dotted line is identical to the second. Moreover the Figure 5.1 validates the capability of the dynamic controller to recreate the optimal cycle. The cycle represented here is identical to one of the solutions provided as an optimal cycle. One key difference is that the dynamic controller always creates the cycle to be exactly 24 seconds while the optimal cycle varied in between 24 to 26 seconds.

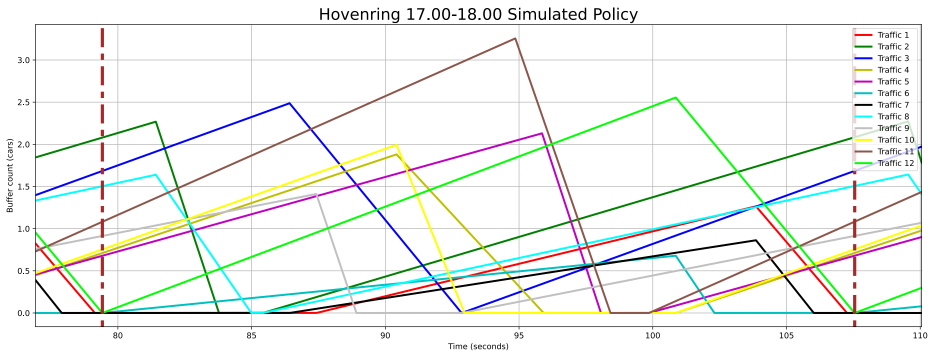


Figure 5.1: Hovenring simulated buffer evolution

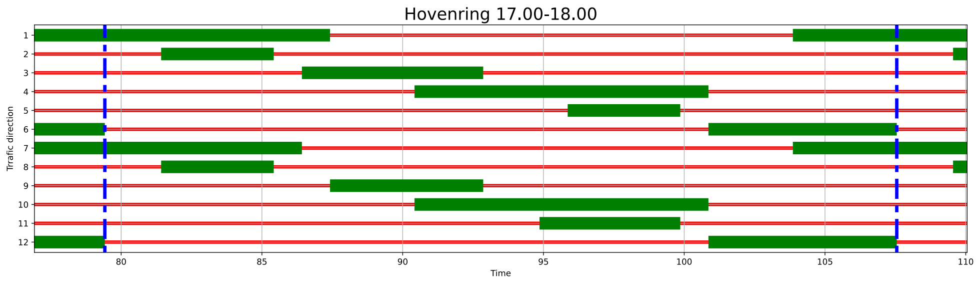


Figure 5.2: Hovenring simulated cycle

Another important simulation was to assess the controller's capability in adapting to changing traffic intensity. With this simulation the result was predicted to have a transition period where the buffer count is in between the two states. After the transition phase the buffer count is expected to become stable at the new intensity rate.

As shown in Figure 5.3, the controller is capable of adapting to changes in traffic intensity and after a cycle or two, it reaches a stable point. The simulation changed the traffic intensity from the hour of 17h to 18h to the hour of 18h to 19h at the time 150 seconds after the simulation started. This represents the real life situation when the traffic intensity shifts after the service period reaching stability. By proving that the stability point is reached automatically, the controller is confirmed to be performing the way it was designed to do.

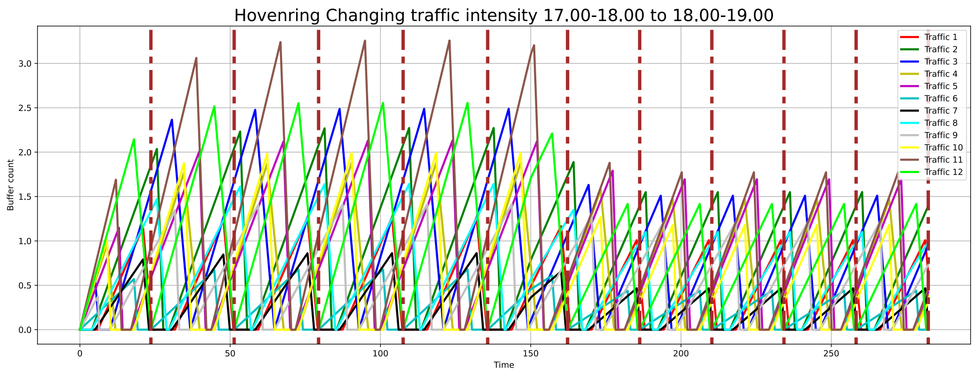


Figure 5.3: Hovenring simulated change in traffic intensity

Average waiting times are a good criteria to analyse the performance of the controller. Compared to the optimal cycle, simulated results performed very similar in terms of average waiting times. Some of the hours had exactly the same average waiting time as the optimal cycle. These directions, the service period for each directions were identical to the optimal cycle. This confirms the performance of the dynamic controller as it was able to recreate the optimal cycle with only the information of the traffic intensity. The time of the day with the largest increase in average waiting time was from 17h to 18h. The Table 5.1 compares the average waiting times of the optimal cycle to the simulated dynamic cycle. For a given direction, the Optimal column represents the average waiting time of the optimal cycle and the Simulated column represents the average waiting time of the dynamic controller. The difference is calculated by taking away the optimal value from the simulated value.

Though the overall difference in average waiting time is longer in the simulated dynamic cycle, there are certain direction where the dynamic controller results in a lower waiting time. These directions can be spotted by difference column being a negative value. The optimal cycle aims to reduce the overall waiting time, therefore the dynamic controller is some times able to outperform it. However due to these negative values, the dynamic controller is able to gain back some of the lost waiting times in other directions to result in the over all average waiting time to decrease. Looking at the results, overall the controller is performing very well. However, its is having to sacrifice performance of some directions over others and it cannot be said that the controller makes the best choice on which directions to prioritise. For some sets of traffic intensity there may be modes that should be prioritised over others to maximise the performance by reducing waiting time. A controller with a feature to determine the priorities of the modes may be possible to be created as it only relies on the traffic intensity data to make the decision. However it is out of scope for this project.

Table 5.1: Hovenring Average waiting time between 17h-18h

17.00-18.00			
Direction	Optimal	Simulated	Difference
1	5.17816	5.798564	0.620404
2	10.34696	11.36311	1.016146
3	10.11834	10.8432	0.724861
4	7.419499	7.291515	-0.12798
5	10.28393	11.29458	1.010641
6	10.06104	8.721351	-1.33969
7	7.128322	6.075878	-1.05244
8	10.82275	11.88148	1.058729
9	10.04919	9.756401	-0.29279
10	4.904842	6.362004	1.457162
11	8.796361	10.97346	2.177095
12	10.00221	10.71787	0.715665
Overall Average	8.685994	9.454078	0.768084

5.2 Mathematical analysis

The performance of the controller under the conditions provided has been confirmed to work. However it is apparent that the controller is not capable of dealing with any traffic intensity. There is a limiting point where the designed policy starts face difficulty dealing with the inflow of cars. With a high traffic intensity situation, the dynamic controller becomes unable to maintain stability. In this section, mathematical theory is applied in order to answer the question “Under what conditions will the controller be functional?”. The conditions of traffic intensity which the controller is capable of handling is identified by applying the theory by V. Feoktistova[1]. In this section, the calculation for mode 1, with direction 2 being the switching key direction is used as an example of the calculation.

5.2.1 Operator calculation

The calculation is done to each different modes as defined in the policy definition and then combined in the later stages to obtain the final results. The first step of the mathematical variation is to investigate the individual modes and to define the possible length of the mode. The length of the mode is determined by the service length of the direction that takes the longest to empty. This results in multiple possible length for each modes. For the example mode 1, possible mode lengths are as shown below. The (5.4) is the length that is used in the following example.

$$\mathrm{T}_1 = \sigma_{12,8 \text{ or } 6,2} + min_{\text{green}} \tag{5.2}$$

$$\mathrm{T}_1 = \sigma_{12,8} + \frac{x_8 + \lambda_8 \cdot \sigma_{12,8}}{\mu_8 - \lambda_8} \tag{5.3}$$

$$\mathrm{T}_1 = \sigma_{6,2} + \frac{x_2 + \lambda_2 \cdot \sigma_{6,2}}{\mu_2 - \lambda_2} \tag{5.4}$$

Following the defined length of the mode, the next step is to create a mode operator. Mode operator is a mapping for the buffer contents at the start to the end of the mode. The mapping forms a form of $Ax + b$. Consisting of a square matrix A, with size the number of service directions and it is multiplied by the existing buffer x at the beginning of the mode. The vector b also with inputs of the number of service direction. For the case of the Hovenring, the A is a 12×12 matrix and b is a 12×1 vector. The mapping of the mode 1 when the length of the mode is defined by (5.4) is as shown in the (5.6). The buffer mapping for a given direction that is not served during the mode forms the (5.5). Where i represents the direction and n represents the direction that determines the length of the mode. x_i is the buffer at the start of the mode and it is only multiplied to the first term making the first term to be represented by the A matrix and the second term to be inserted into the b vector.

$$\frac{\lambda_i}{\mu_n - \lambda_n} x_i + \frac{\lambda_i \cdot \sigma_n}{1 - \rho_n} \tag{5.5}$$

For the directions that are emptied at the end of the mode, the contents in A matrix and b vector are 0. Lastly, in some modes, there are directions that are served at the beginning but then finishes the service during the mode or directions that start serving but continues service through onto the next mode. These directions can all be expressed using the format

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{\lambda_3}{\mu_2-\lambda_2} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{\lambda_4}{\mu_2-\lambda_2} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{\lambda_5}{\mu_2-\lambda_2} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{\lambda_6}{\mu_2-\lambda_2} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{\lambda_9}{\lambda_9-\lambda_2} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & \frac{\lambda_{10}}{\mu_2-\lambda_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & \frac{\lambda_{11}}{\mu_2-\lambda_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & \frac{\lambda_{12}}{\mu_2-\lambda_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix} X_1 + \begin{bmatrix}
(\lambda_1 - \mu_1)\left(\sigma_{6,2} + \frac{x_2+\lambda_2\cdot\sigma_{6,2}}{\mu_2-\lambda_2}\right) \\
0 \\
\frac{\lambda_3\cdot\sigma_{6,2}}{1-\rho_2} \\
\frac{\lambda_4\cdot\sigma_{6,2}}{1-\rho_2} \\
\frac{\lambda_5\cdot\sigma_{6,2}}{1-\rho_2} \\
\frac{\lambda_6\cdot\sigma_{6,2}}{1-\rho_2} \\
(\lambda_7 - \mu_7)\left(\sigma_{6,2} + \frac{x_2+\lambda_2\cdot\sigma_{6,2}}{\mu_2-\lambda_2}\right) \\
0 \\
\frac{\lambda_9\cdot\sigma_{6,2}}{1-\rho_2} \\
\frac{\lambda_{10}\cdot\sigma_{6,2}}{1-\rho_2} \\
\frac{\lambda_{11}\cdot\sigma_{6,2}}{1-\rho_2} \\
\frac{\lambda_{12}\cdot\sigma_{6,2}}{1-\rho_2}
\end{bmatrix} \tag{5.6}$$

The formed mapping can be simplified further by removing the 0 terms and also simplifying the A matrix that is multiplied by buffer terms X that are empty at the beginning of the mode. In this case the direction 2 and 8 are emptied therefore can be removed due to the 0 values and the directions 6 and 12 are empty at the beginning so all the terms that are multiplied by x_6 and x_{12} in the buffer vector X_1 can be removed.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\lambda_3}{\mu_2-\lambda_2} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\lambda_4}{\mu_2-\lambda_2} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\lambda_5}{\mu_2-\lambda_2} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\lambda_6}{\mu_2-\lambda_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & \frac{\lambda_9}{\lambda_9-\lambda_2} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{\lambda_{10}}{\mu_2-\lambda_2} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & \frac{\lambda_{11}}{\mu_2-\lambda_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & \frac{\lambda_{12}}{\mu_2-\lambda_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} X_1 + \begin{bmatrix} (\lambda_1-\mu_1)\left(\sigma_{6,2}+\frac{x_2+\lambda_2\cdot\sigma_{6,2}}{\mu_2-\lambda_2}\right) \\ \frac{\lambda_3\cdot\sigma_{6,2}}{1-\rho_2} \\ \frac{\lambda_4\cdot\sigma_{6,2}}{1-\rho_2} \\ \frac{\lambda_5\cdot\sigma_{6,2}}{1-\rho_2} \\ \frac{\lambda_6\cdot\sigma_{6,2}}{1-\rho_2} \\ (\lambda_7-\mu_7)\left(\sigma_{6,2}+\frac{x_2+\lambda_2\cdot\sigma_{6,2}}{\mu_2-\lambda_2}\right) \\ \frac{\lambda_9\cdot\sigma_{6,2}}{1-\rho_2} \\ \frac{\lambda_{10}\cdot\sigma_{6,2}}{1-\rho_2} \\ \frac{\lambda_{11}\cdot\sigma_{6,2}}{1-\rho_2} \\ \frac{\lambda_{12}\cdot\sigma_{6,2}}{1-\rho_2} \end{bmatrix} \tag{5.7}$$

The simplified mode operator, is checked against the conditions defined by the paper [1], required in order to have a unique fixed point and attract all solutions. These conditions have to be met in order to have a stable traffic controller. The mode operator has to be:

1. Piecewise affine
   i.e. $T(x) = A_i x + b_i$ for $x \in \{P_i x \le q_i\}$
2. Continuous
   Meaning that there exist a operator for any situations, which can be confirmed already in the phase of defining the policy.
3. Monotone
   $A_i \ge 0$, All values in the A matrix has to be 0 or positive.
4. Strictly dominated
   $b_i > 0$, All values in the b vector has to be a positive value not including 0.
5. Have a fixed point Cannot be confirmed using the mode operator.

Looking at the example operator (5.7), all of the conditions can be confirmed to be met except the last one. At this point, if a operator is not able to meet some of the conditions, multiple operators can be combined into one mapping and the combined operator has to be confirmed to meet the conditions. The next step is to confirm that the controller has a fixed point.

5.2.2 Fixed point confirmation

In order to confirm that fix point, the calculation is started by assuming the controller is stable. Using the knowledge that the controller operates with the clearing policy, at end of each modes, there exists some buffer directions that are empty. Following the buffer around once, it is possible to define the buffer count in terms of different mode length. For the mode 1, buffer 2 and 8 are emptied at the end, therefore it is possible to formulate the formula for the buffer contents for these directions at the start of mode 1 in terms of mode length of mode 2, 3 and 4. The buffer contents at the start of mode 1 is shown in (5.8) and (5.9).

$$x_2 = \lambda_2 (T_2 + T_3 + T_4) \tag{5.8}$$

$$x_8 = \lambda_8 (T_2 + T_3 + T_4) \tag{5.9}$$

The formula for the buffer x can be substitute into the (5.4) creating an equation consisting of only the unknown mode lengths and the known inflow λ and outflow rates μ.

$$\mathrm{T}_1 = \sigma_{6,2} + \frac{\lambda_2 (T_2 + T_3 + T_4) + \lambda_2 \cdot \sigma_{6,2}}{\mu_2 - \lambda_2} \tag{5.10}$$

From the different possible choices of key directions for each mode, a combination can be created and in total four formulas similar to (5.10) can be created for each of the modes. For the Hovenring, it was required to create over 100 combinations due to the large number of possible combinations. Often times the inflow and outflow values can be replaced by the utilization, further simplifying the equation.

Finally using the created four formulas, the equations for the mode length T_n can be created. Since the mode length has to be a none negative value, a equation including the utilization which has to be larger than zero can be formed. Creating a formula for which the requirements for utilization can be calculated from.

5.2.3 Service Limitation

From the calculations, the limitation of the maximum inflow for the created controller were created. The requirements limit the maximum utilization for the combinations of directions of services.

- $\rho_{2 \text{ or } 8} + \rho_{3 \text{ or } 9} + \rho_{5 \text{ or } 11} + \rho_{6 \text{ or } 12} < 1$
  This conditions requires the services for the directions that are serviced only within a single mode to be limited in terms of utilization represented by the blue highlights on the Figure 5.4. There are 4 modes in total therefore if one of the combinations of the utilization combinations exceeds 1, the inflow of cars is larger than the maximum service rate and therefore results in backlash and pileup of buffer or even worse, a complete failure.

- $\rho_3 + \rho_7 + \rho_{11} < 1$
  $\rho_9 + \rho_1 + \rho_5 < 1$
  $\rho_2 + \rho_4 + \rho_6 < 1$
  $\rho_8 + \rho_{10} + \rho_{12} < 1$
  These condition requires the direction that is served across multiple modes to be combined with two of the directions served for a single mode. One of these sets are shown in the Figure 5.4 as orange highlights which represents the case $\rho_3 + \rho_7 + \rho_{11} < 1$.

- $\rho_{1 \text{ or } 7} + \rho_{4 \text{ or } 10} < 1$ This is a condition between the pair of two directions that are both served for the duration of multiple modes. A example is represented as the yellow highlight on the Figure 5.4.

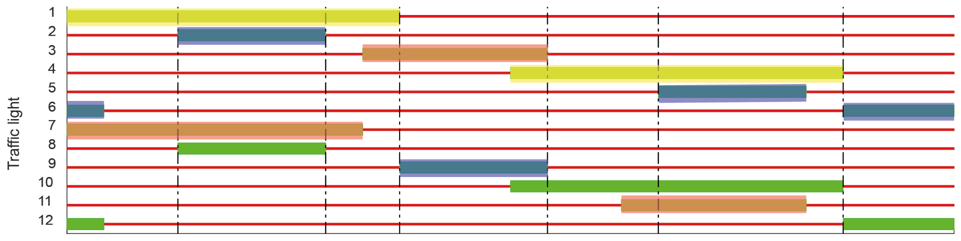


Figure 5.4: Hovenring simulated cycle

5.3 Analysis conclusion

The dynamic policy was analysed by both the simulation and the mathematical analysis. The simulation has proved that the controller is well within the operational limits for a given intensity data from different times of the day of the Hovenring intersection. The mathematical analysis has managed to outline the limitation in the maximum utilization for a number of combinations of directions. Checking the limitations, it was also confirmed that the traffic intensity data from Hovenring is well within this limit therefore validating the controller to be used in the intersection in order to control the twelve different directions.

Chapter 6

Conclusion

6.1 Conclusion

In conclusion, the aim to create a dynamic controller for The Hovenring, capable of adapting to different traffic intensity situation has been achieved successfully. The intersection was analysed with large margins of safety avoiding the possibility for the controller to cause any accidents. The optimized cycles for each of the different hours of the day were created and analysed to be used as the guidance in the process of creating the dynamic controller. Finally, the dynamic controller was developed and validated using two methods which both concluded that the controller is stable and therefore should operate with similar performance to the optimal fixed time controller. Furthermore, the conditions that the controller remains in stable operation has been met by the traffic intensity data of Hovenring. Therefore, the developed controller is ready to be deployed in real life.

6.2 Recommendation

The dynamic controller development for Hovenring has been successful. However this project was conducted under a very specific situation, working specifically for the Hovenring intersection especially after Chapter 4. Moreover the controller was created under the assumption that the traffic intensity for every hour is a known factor and is a perfect representation of the real world.

Under these assumptions, the created dynamic controller has the best performance. However, not many alternative options were considered in this project and there is still room for more investigation. For instance, there may be some performance to be gained by the introduction of another mode. This is not too likely as the introduction of an additional mode results in increase of cycle time which in turn may increase the average waiting time.

Another possible way to improve the performance is to make adjustment in the intersection layout in it self. Currently, there are multiple lanes for certain direction but not for others. By looking at the allocations additional lanes to the different directions, there may be some performance to be gained without having to make major adjustment to the overall intersection footprint. Though it may confuse the drivers, it may be possible to create adaptable lanes that changes in service direction depending on the time of the day to fully maximise the service rate of the intersection.

In order to apply this project to other intersections, the codes used for simulations and analysis can be adapted. However, it may be better for a new generic code to be created using the current code as a guidance as there are multiple points that are experimental with the code. In addition to this, by improving the simulations, it will be possible to simulate more complex variations in the traffic intensity such as situation with random varying inflow rate throughout the simulation run.

Faster and optimised code enable the possibility of creating a dynamic controller capable of adjusting to real life data feed from the sensors at the intersection. Current dynamic controller utilises the data gathered in advance and predicts the traffic intensity in the future. By having multiple sensors, it should be able to measure the traffic intensity and adapt instantly resulting in even less waiting time.

The mathematical validation was the section where most number of problems and difficulties were encountered in this project. This was due to the numerous possible combinations and the limitation in the MATLAB symbolic calculation skills, it was unable to fully automate the process. There may have been a method that was missed which may help in automating the calculation process but with the current setup it is not recommended for any intersection more complex than the Hovenring to be approached. For a complex intersection, running multiple simulations with different combinations of traffic intensity will probably be faster and more efficient than the mathematical approach.

Overall, there are still multiple possibilities for an improvement available and further research is crucial in improving the waiting times at the intersection. These research will not only improve the driver experience at an intersection but can may be applied to other situations with multi input from different direction with changing inflow rates.

Bibliography

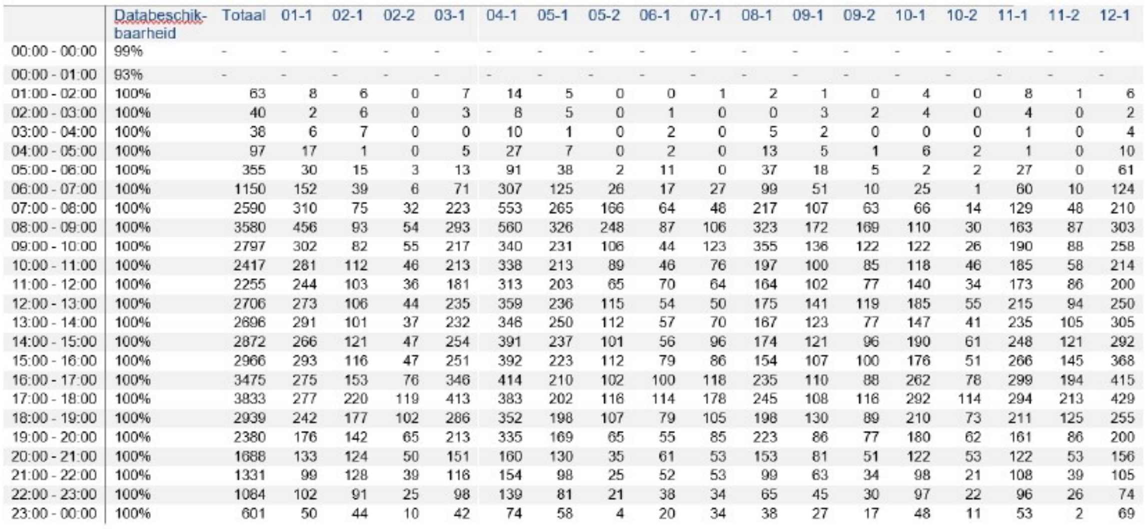
[1] Varvara Feoktistova et al. “Designs of optimal switching feedback decentralized control policies for fluid queueing networks”. In: *Mathematics of Control, Signals, and Systems* 24.4 (2012), pp. 477–503.

[2] STG Fleuren. “Optimizing pre-timed control at isolated intersections”. Master’s thesis. Eindhoven University of Technology, 2015.

Appendices

A Intersection Data

A.1 Traffic intensity data provided by city of Eindhoven



	Databeschik-baarheid	Totaal	01-1	02-1	02-2	03-1	04-1	05-1	05-2	06-1	07-1	08-1	09-1	09-2	10-1	10-2	11-1	11-2	12-1
00:00 - 00:00	99%	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
00:00 - 01:00	93%	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
01:00 - 02:00	100%	63	8	6	0	7	14	5	0	0	1	2	1	0	4	0	8	1	6
02:00 - 03:00	100%	40	2	6	0	3	8	5	0	1	0	0	3	2	4	0	4	0	2
03:00 - 04:00	100%	38	6	7	0	0	10	1	0	2	0	5	2	0	0	0	1	0	4
04:00 - 05:00	100%	97	17	1	0	5	27	7	0	2	0	13	5	1	6	2	1	0	10
05:00 - 06:00	100%	355	30	15	3	13	91	38	2	11	0	37	18	5	2	2	27	0	61
06:00 - 07:00	100%	1150	152	39	6	71	307	125	26	17	27	99	51	10	25	1	60	10	124
07:00 - 08:00	100%	2590	310	75	32	223	553	265	166	64	48	217	107	63	66	14	129	48	210
08:00 - 09:00	100%	3580	456	93	54	293	560	326	248	87	106	323	172	169	110	30	163	87	303
09:00 - 10:00	100%	2797	302	82	55	217	340	231	106	44	123	355	136	122	122	26	190	88	258
10:00 - 11:00	100%	2417	281	112	46	213	338	213	89	46	76	197	100	85	118	46	185	58	214
11:00 - 12:00	100%	2255	244	103	36	181	313	203	65	70	64	164	102	77	140	34	173	86	200
12:00 - 13:00	100%	2706	273	106	44	235	359	236	115	54	50	175	141	119	185	55	215	94	250
13:00 - 14:00	100%	2696	291	101	37	232	346	250	112	57	70	167	123	77	147	41	235	105	305
14:00 - 15:00	100%	2872	266	121	47	254	391	237	101	56	96	174	121	96	190	61	248	121	292
15:00 - 16:00	100%	2966	293	116	47	251	392	223	112	79	86	154	107	100	176	51	266	145	368
16:00 - 17:00	100%	3475	275	153	76	346	414	210	102	100	118	235	110	88	262	78	299	194	415
17:00 - 18:00	100%	3833	277	220	119	413	383	202	116	114	178	245	108	116	292	114	294	213	429
18:00 - 19:00	100%	2939	242	177	102	286	352	198	107	79	105	198	130	89	210	73	211	125	255
19:00 - 20:00	100%	2380	176	142	65	213	335	169	65	55	85	223	86	77	180	62	161	86	200
20:00 - 21:00	100%	1688	133	124	50	151	160	130	35	61	53	153	81	51	122	53	122	53	156
21:00 - 22:00	100%	1331	99	128	39	116	154	98	25	52	53	99	63	34	98	21	108	39	105
22:00 - 23:00	100%	1084	102	91	25	98	139	81	21	38	34	65	45	30	97	22	96	26	74
23:00 - 00:00	100%	601	50	44	10	42	74	58	4	20	34	38	27	17	48	11	53	2	69

Figure A.1: Hovenring Traffic Intensities

A.2 Numbering of Traffic Directions

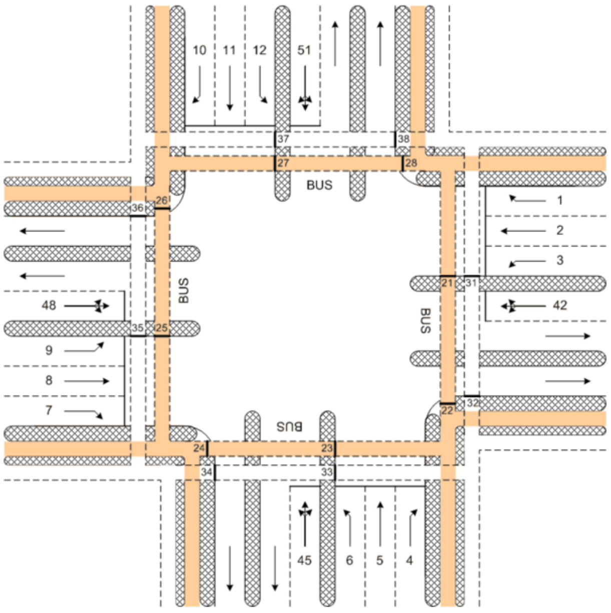


Figure A.2: Traffic direction Numbering Method

B Simulation Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# from matplotlib.figure import Figure
import sys   # used to output error message
from cycler import cycler   # used to control the colours of the plot
import networkx as nx # for the network analysis Graphb
import statistics


### Name of the intersection
intersection = 'Hovenring'


### Time range of the data
t_range_list = ['01.00-02.00', '02.00-03.00', '03.00-04.00', ...
    '04.00-05.00', '05.00-06.00',
                '06.00-07.00', '07.00-08.00', '08.00-09.00', ...
    '09.00-10.00', '10.00-11.00',
                '11.00-12.00', '12.00-13.00', '13.00-14.00', ...
    '14.00-15.00', '15.00-16.00',
                '16.00-17.00', '17.00-18.00', '18.00-19.00', ...
    '19.00-20.00', '20.00-21.00',
                '21.00-22.00', '22.00-23.00', '23.00-00.00' ]


# Name of the sheet the data is stored in
lightschedule_data = '0 extra groen, Nr 1 (Geen)'

############################################################################
# simulation start
data = {}
waitingfromdata = {}


for n_timerange in np.arange(0, len(t_range_list)):
    t_range = t_range_list[n_timerange]
    lights_data = pd.ExcelFile('./'+intersection+' - '+t_range+'.xlsm')



    # not necessary
    # # Period info (lists when traffic turns green and red for each ...
    direction)
    # df_tcont = pd.read_excel (lights_data, ...
    sheet_name=lightschedule_data, index_col=0, usecols=[0, 1, 2, 3])
    # # list of direction that is served (for the traffic )
    # traffic_distribution_direction = df_tcont.index     # dtype='object'


    # conflict matrix (list of where confricts may occur and the gap ...
    required to avoid it)
        # 1st column is loaded as a string as it will be set to index in ...
    the following steps
    df_conflict = pd.read_excel (lights_data, sheet_name="Conflicten", ...
    skiprows=3, dtype={0: str})
        # index is not correctly set therefore 1st the coloumn that will ...
    be used for index is named
    df_conflict.rename(columns={ df_conflict.columns[0]: "From\To" }, ...
    inplace = True)
        # The index is changed to the correct column
    df_conflict = df_conflict.set_index('From\To')


    # stoplight info. data about how many cars are incoming into
    df_stopinfo = pd.read_excel (lights_data, sheet_name="Stoplicht ...
    info", skiprows=0, index_col=0, usecols=[0, 1, 2,4])

    # converting the inflow info to cars per second instead of hours ...
    (adds two coloums)
    df_stopinfo['Intensiteit \n\n(pae/s)'] = df_stopinfo['Intensiteit ...
    \n\n(pae/u)']/3600
    df_stopinfo['Capaciteit  \n\n(pae/s)'] = df_stopinfo['Capaciteit  ...
    \n\n(pae/u)']/3600

    Min_green = df_stopinfo.to_dict()['Minimaal \nGroen\n(s)']

    # Making list of in and out flow rate
    OUT_flow_rate = df_stopinfo.to_dict()['Capaciteit  \n\n(pae/s)']
    IN_flow_rate = df_stopinfo.to_dict()['Intensiteit \n\n(pae/s)']


    # might be used
    # getting period and average waiting time for the given intersection
    cycleinfo = pd.read_excel (lights_data, ...
    sheet_name=lightschedule_data, index_col=0, usecols=[5,6])
    period = float(cycleinfo.loc['Periodetijd:'])
    Avg_waiting_t =  float(cycleinfo.loc['Gemiddelde wachttijd:'])


    # Creating the list of traffic directions in the intersection (in ...
    numbers) from stopinfo
    traffic_direction = df_stopinfo.index     # dtype='float64'


############################################################################

    t_lenght = 100 #second
    t_step = 0.01

    buffer = {}
    light_green = {}
    start_green = {}
    start_red = {}
    red_green_check={}

    # buffer[1] = IN_flow_rate
    # buffer

    def_green_start = -100
    def_red_start = 9999999

    # making the buffer for all directions
    for n_traffic in np.arange(0, len(traffic_direction)):
        buffer[traffic_direction[n_traffic]] = [0]

    for n_traffic in np.arange(0, len(traffic_direction)):
        light_green[traffic_direction[n_traffic]] = False

    for n_traffic in np.arange(0, len(traffic_direction)):
        start_green[traffic_direction[n_traffic]] = def_green_start

    for n_traffic in np.arange(0, len(traffic_direction)):
        start_red[traffic_direction[n_traffic]] = def_red_start

    graph_loc_red_green ={1: 12, 2: 11,  3: 10,  4: 9,  5: 8,  6: 7,  7: ...
6,  8: 5,  9: 4,  10: 3,  11: 2,  12: 1,}

    for n_traffic in np.arange(0, len(traffic_direction)):
        red_green_check[traffic_direction[n_traffic]] = [0]



    # setting up time to base the graph on. (y axis) set to interval of ...
0.01s for the lenght of simulation time.
    # t_lenght = 1000 #second
    # t_step = 0.01

    t_graph = np.arange(0, (t_lenght), t_step)

    t = np.arange(0, (t_lenght-t_step), t_step)
    Switch_condition = {}  # dictionary to insert the switching directions

    # computation function to facilitate this step
    # calculates the buffer for the direction with red
    def red_buffer(traffic_id, n):    # insert direction and t value.
        # the next value in the buffer is equal to the step t times by ...
the arrival rate
        buffer[traffic_id].append(buffer[traffic_id][n]+ ...
(t_step*IN_flow_rate[traffic_id]))

    # calculates the buffer for the direction with green
    def green_buffer(traffic_id, n):  # insert direction and t value.
        # The buffer count decreases by the in flow rate - out flow rate.
        # calculation calculates the new buffer value
        calculation = buffer[traffic_id][n]+ (t_step * ...
(IN_flow_rate[traffic_id]-OUT_flow_rate[traffic_id]))

        # Addition is the value that will be updated as the new value.
        # This makes sure that the buffer never becomes negative value
        addition = calculation if calculation > 0  else 0

        # appending the calculated buffer value and updating the list
        buffer[traffic_id].append(addition)

    def emp_mg(traffic_id):
        empty_and_mg = buffer[traffic_id][n_sim] == 0 and t[n_sim] - ...
start_green[traffic_id] ≥ Min_green[traffic_id]
        return empty_and_mg

    def switch_green(traffic_id):
        if light_green[traffic_id] != True:
            light_green[traffic_id] = True
            start_green[traffic_id] = t[n_sim]
            start_red[traffic_id] = def_red_start

    def switch_red(traffic_id):
        if light_green[traffic_id] != False:
            light_green[traffic_id] = False
            start_green[traffic_id] = def_green_start
            start_red[traffic_id] = t[n_sim]


```

```
sim_time = 0
sim_mode = 1
t_startmode = 0
start = True
cyclecount = 1
cycledata = {}


slowdown = [0]
for n_sim in np.arange(0, len(t)):   #slowdown:

    if start == True:
        switch_green(1)
        switch_green(2)
        switch_green(7)
        switch_green(8)
        start = False


# MODE 1
    if sim_mode == 1:

        if t[n_sim] - start_red[6] ≥ df_conflict.loc['6','2'] and ...
t[n_sim] - start_red[6] ≥ df_conflict.loc['6','8'] and          ...
t[n_sim] - start_red[12] ≥ df_conflict.loc['12','2'] and t[n_sim] - ...
start_red[12] ≥ df_conflict.loc['12','8']:
            switch_green(2)
            switch_green(8)





    # switching
        # case where the 1,2 switches to 9,10
        if emp_mg(1) and emp_mg(2):
            switch_red(2)

            t_startmode = t[n_sim]
            sim_mode = 2

        # case where the 7,8 switches to 3,4
        elif emp_mg(7) and emp_mg(8):
            switch_red(8)

            t_startmode = t[n_sim]
            sim_mode = 2

        # case where the 1,2,7,8 switches to 3,4,9,10
        elif emp_mg(1) and emp_mg(2) and emp_mg(7) and emp_mg(8):
            switch_red(2)
            switch_red(8)

            t_startmode = t[n_sim]
            sim_mode = 2

#         else: sim_mode == 1
```

```
    # MODE 2
        elif sim_mode == 2:
            if t[n_sim] - start_red[2] ≥ df_conflict.loc['2','9']:
                switch_red(1)
                switch_green(9)
            if t[n_sim] - start_red[8] ≥ df_conflict.loc['8','3']:
                switch_red(7)
                switch_green(3)

            if t[n_sim] - start_red[2] ≥ df_conflict.loc['2','10']:
                switch_green(10)

            if t[n_sim] - start_red[8] ≥ df_conflict.loc['8','4']:
                switch_green(4)

            #else: sim_mode == 2

            if emp_mg(7) and emp_mg(8):
                switch_red(8)

            if emp_mg(1) and emp_mg(2):
                switch_red(2)

        # switching to mode 3
            if emp_mg(3) and emp_mg(9):
                switch_red(3)
                switch_red(9)
                sim_mode = 3






    # MODE 3
        elif sim_mode == 3:
            if t[n_sim] - start_red[3] ≥ df_conflict.loc['3','11'] and ...
    t[n_sim] - start_red[9] ≥ df_conflict.loc['9','11'] :
                switch_green(11)
            if t[n_sim] - start_red[9] ≥ df_conflict.loc['9','5'] :
                switch_green(5)

            if t[n_sim] - start_red[2] ≥ df_conflict.loc['2','10']:
                switch_green(10)

            if t[n_sim] - start_red[8] ≥ df_conflict.loc['8','4']:
                switch_green(4)

        # switching
            if emp_mg(4) and emp_mg(10) and emp_mg(5) and emp_mg(11):
                switch_red(5)
                switch_red(11)
                sim_mode = 4

            elif emp_mg(10) and emp_mg(11):
                switch_red(11)
                sim_mode = 4

            elif emp_mg(4) and emp_mg(5):
                switch_red(5)
                sim_mode = 4




    # MODE 4
        elif sim_mode == 4:
            if emp_mg(10) and emp_mg(11):
                switch_red(11)
            if emp_mg(4) and emp_mg(5):
                switch_red(5)

            if t[n_sim] - start_red[5] ≥ df_conflict.loc['5','12'] :
                switch_red(4)
                switch_green(12)

            if t[n_sim] - start_red[11] ≥ df_conflict.loc['11','6'] :
                switch_red(10)
                switch_green(6)

            if t[n_sim] - start_red[5] ≥ df_conflict.loc['5','1'] :
                switch_green(1)

            if t[n_sim] - start_red[11] ≥ df_conflict.loc['11','7'] :
                switch_green(7)

        # Switching
            if emp_mg(6) and emp_mg(12):
                switch_red(12)
                switch_red(6)
                sim_mode = 1

                cyclecount = cyclecount + 1
                cycledata[cyclecount] = [cyclecount, t[n_sim], n_sim]



    # updating the next value
        green_direction = []
        red_direction = []
        for direction_sort in np.arange(0, len(traffic_direction)):
            if light_green[traffic_direction[direction_sort]] == True :
                green_direction.append(traffic_direction[direction_sort])
            elif light_green[traffic_direction[direction_sort]] == False:
                red_direction.append(traffic_direction[direction_sort])


        for n_green in np.arange(0, len(green_direction)):
            green_buffer(green_direction[n_green], n_sim)
            ...
    red_green_check[green_direction[n_green]].append(graph_loc_red_green[green_direction[n

        for n_red in np.arange(0, len(red_direction)):
            red_buffer(red_direction[n_red], n_sim)
            red_green_check[red_direction[n_red]].append(0)



############################################################################


    start_point = cyclecount-1

    p_Start = cycledata[start_point][2]
    p_end = cycledata[start_point+1][2]

    mean_buffer={}

    t_mean_wait_weighted= np.zeros(len(traffic_direction))
    arrive_rate = np.zeros(len(traffic_direction))
    t_mean_wait = np.zeros(len(traffic_direction))
    waitingfromdata[t_range]=float(cycleinfo.loc['Gemiddelde wachttijd:'])

    for n_buff_stat in np.arange(0, len(traffic_direction)):
        mean_buffer[traffic_direction[n_buff_stat]] = ...
    statistics.mean(buffer[traffic_direction[n_buff_stat]][p_Start:p_end])
        arrive_rate[n_buff_stat] = ...
    df_stopinfo.loc[traffic_direction[n_buff_stat], 'Intensiteit ...
    \n\n(pae/s)']

        t_mean_wait[n_buff_stat] = ...
    (mean_buffer[traffic_direction[n_buff_stat]])/arrive_rate[n_buff_stat]

        t_mean_wait_weighted[n_buff_stat] = ...
    t_mean_wait[n_buff_stat]*arrive_rate[n_buff_stat]


    mw_alldirection = sum(t_mean_wait_weighted)/sum(arrive_rate)

    data[(t_range)]  = np.append(t_mean_wait, mw_alldirection)


##########################################################################

avg_ind = traffic_direction.union(['Simulated Mean'])

average = pd.DataFrame(data, index =avg_ind)

#average.loc['Mean'] = average.mean()
average.loc['From optimization'] = waitingfromdata

average.loc['Difference'] = average.loc['Simulated Mean'] - ...
    average.loc['From optimization']

average.loc['cycle length (s)'] = cycledata[start_point+1][1] - ...
    cycledata[start_point][1]

average
```

C Simulation Results

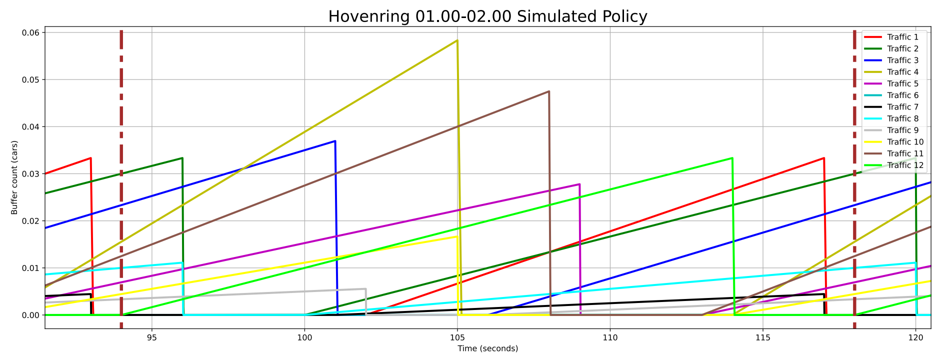


Figure C.1: Hovenring simulated buffer evolution

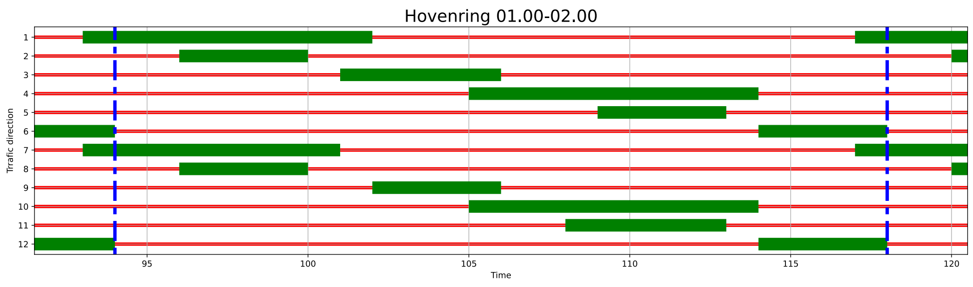


Figure C.2: Hovenring simulated cycle

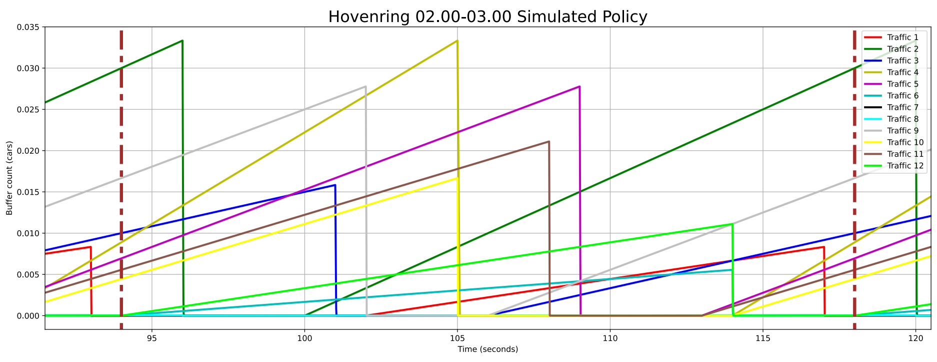


Figure C.3: Hovenring simulated buffer evolution

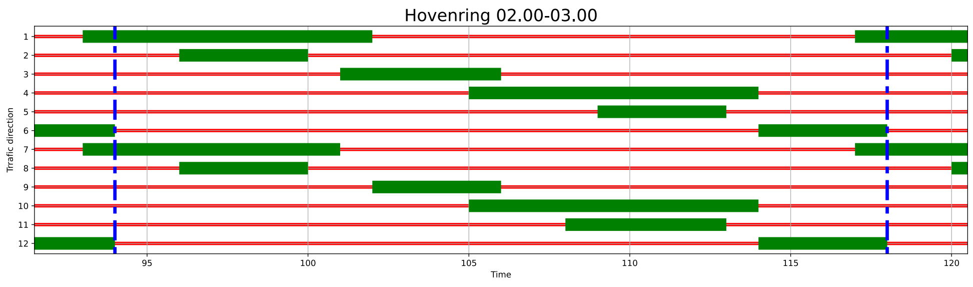


Figure C.4: Hovenring simulated cycle

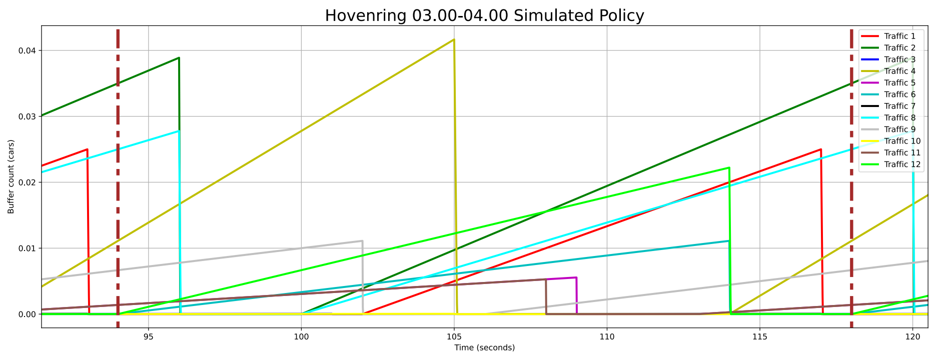


Figure C.5: Hovenring simulated buffer evolution

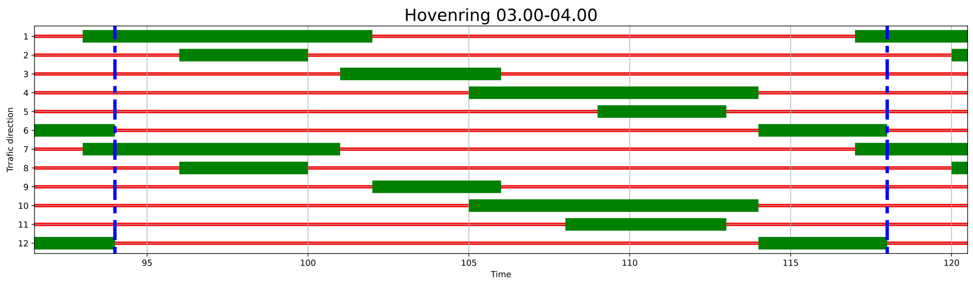


Figure C.6: Hovenring simulated cycle

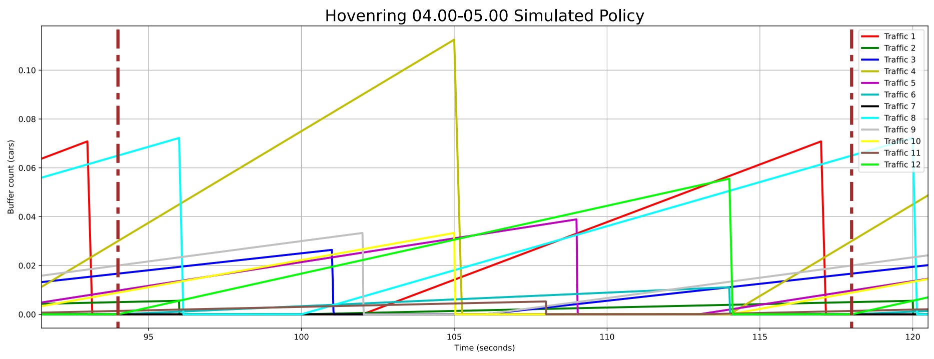


Figure C.7: Hovenring simulated buffer evolution

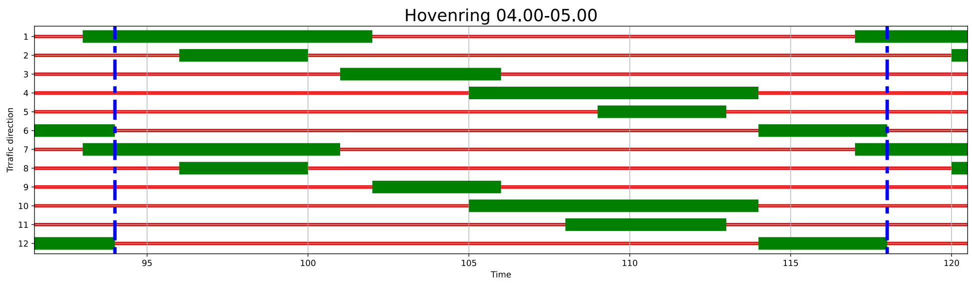


Figure C.8: Hovenring simulated cycle

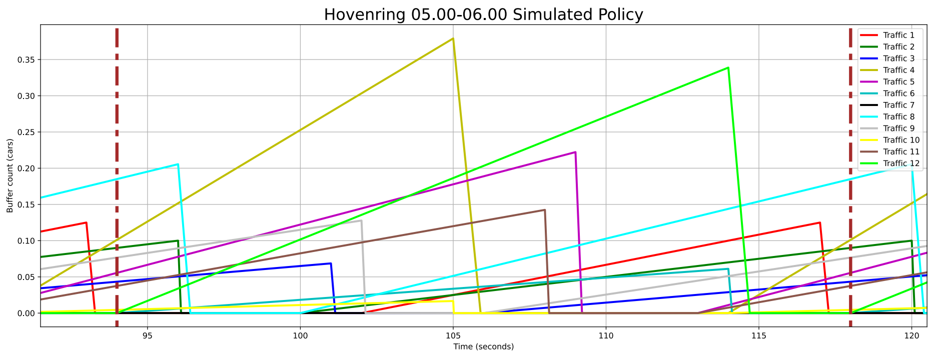


Figure C.9: Hovenring simulated buffer evolution

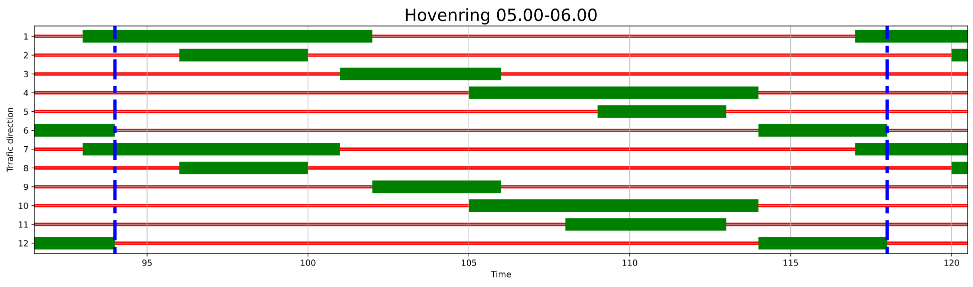


Figure C.10: Hovenring simulated cycle

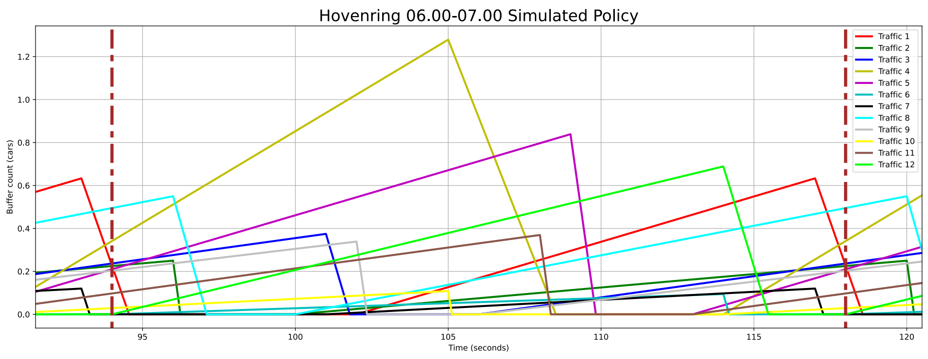


Figure C.11: Hovenring simulated buffer evolution

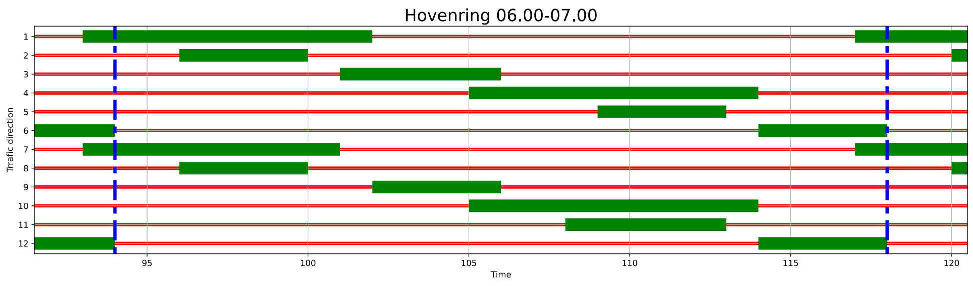


Figure C.12: Hovenring simulated cycle

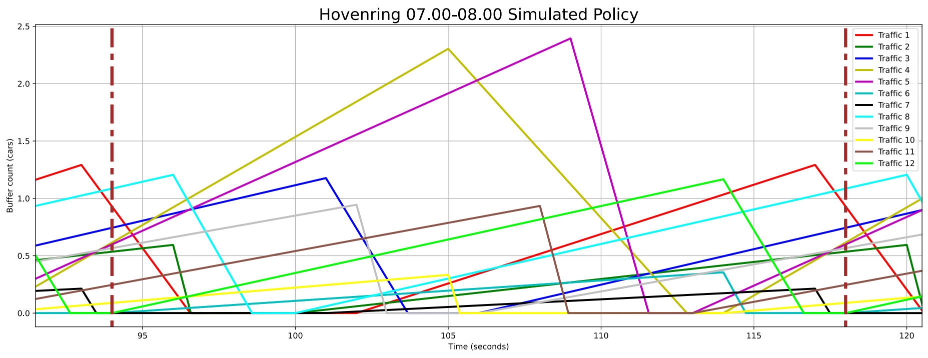


Figure C.13: Hovenring simulated buffer evolution

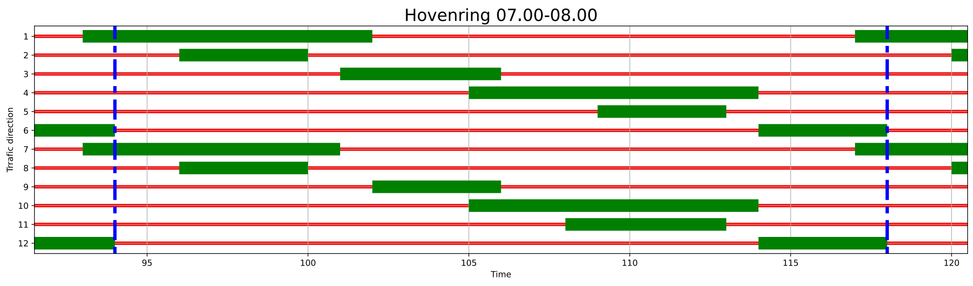


Figure C.14: Hovenring simulated cycle

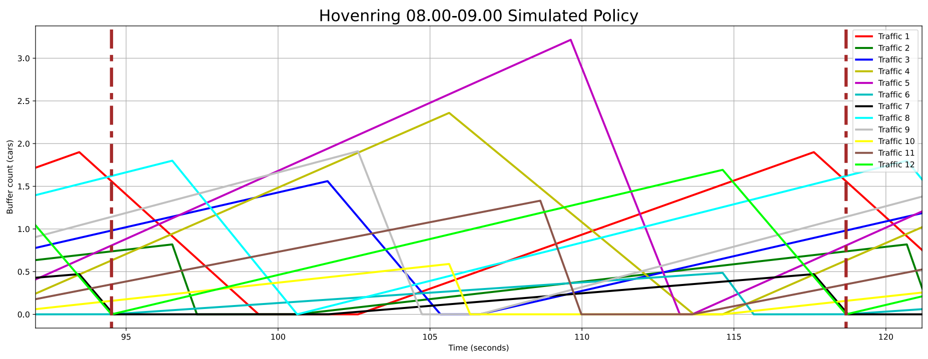


Figure C.15: Hovenring simulated buffer evolution

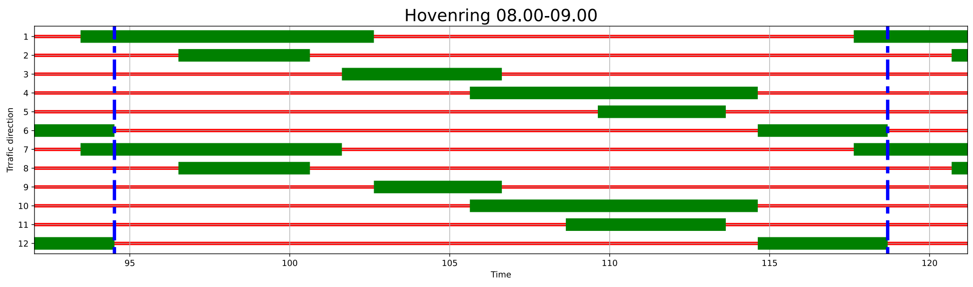


Figure C.16: Hovenring simulated cycle

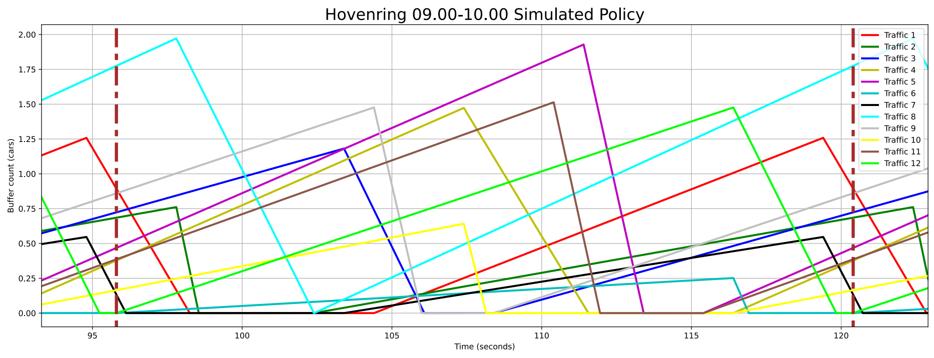


Figure C.17: Hovenring simulated buffer evolution

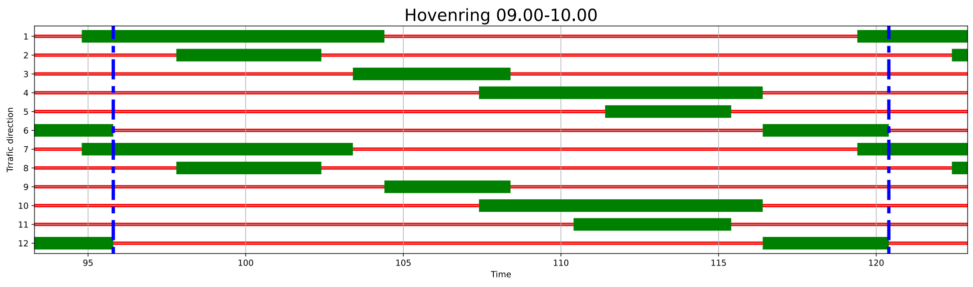


Figure C.18: Hovenring simulated cycle

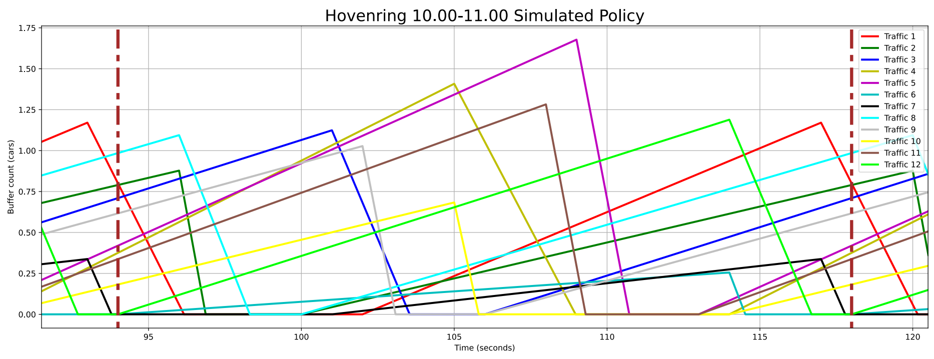


Figure C.19: Hovenring simulated buffer evolution

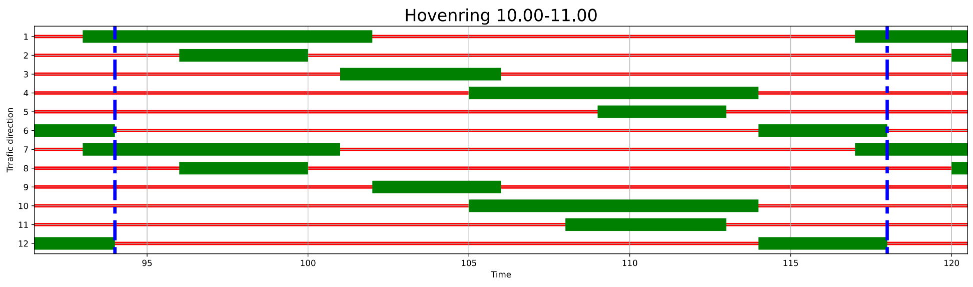


Figure C.20: Hovenring simulated cycle

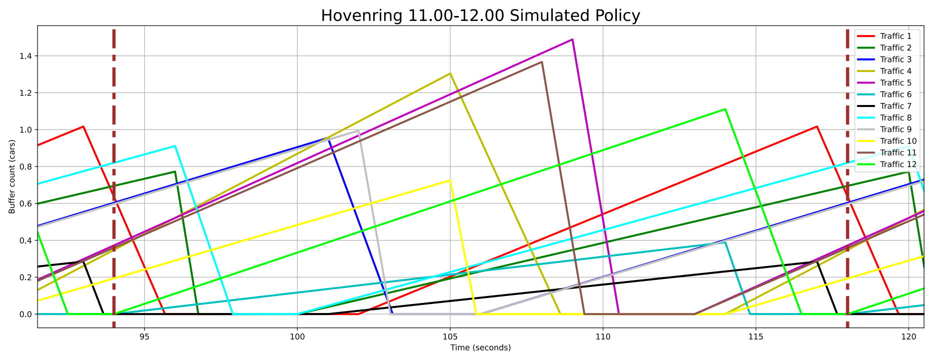


Figure C.21: Hovenring simulated buffer evolution

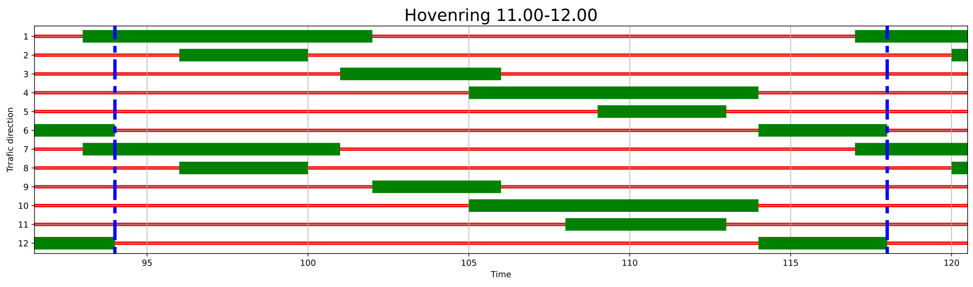


Figure C.22: Hovenring simulated cycle

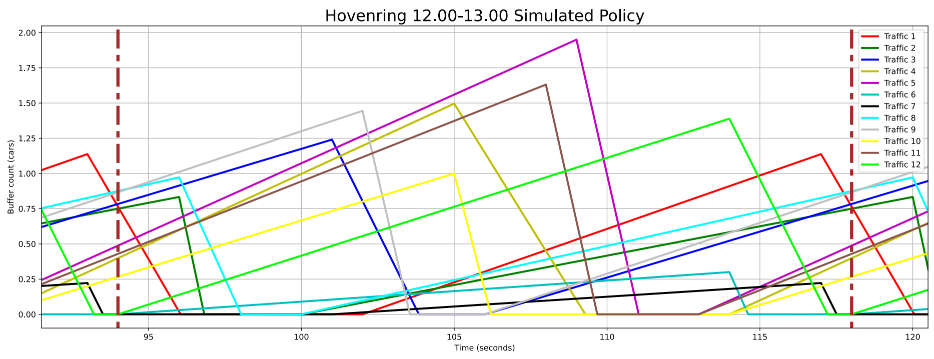


Figure C.23: Hovenring simulated buffer evolution

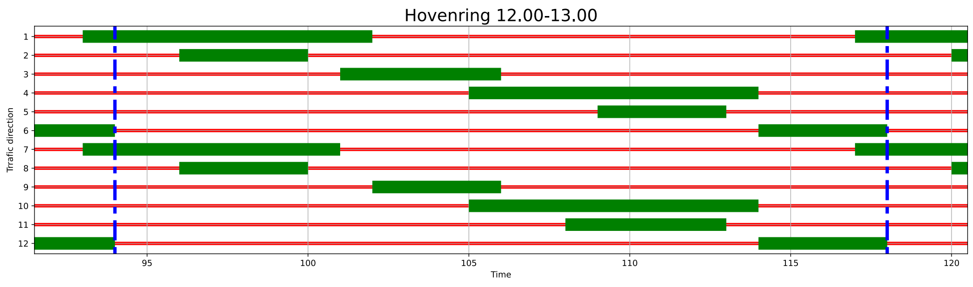


Figure C.24: Hovenring simulated cycle

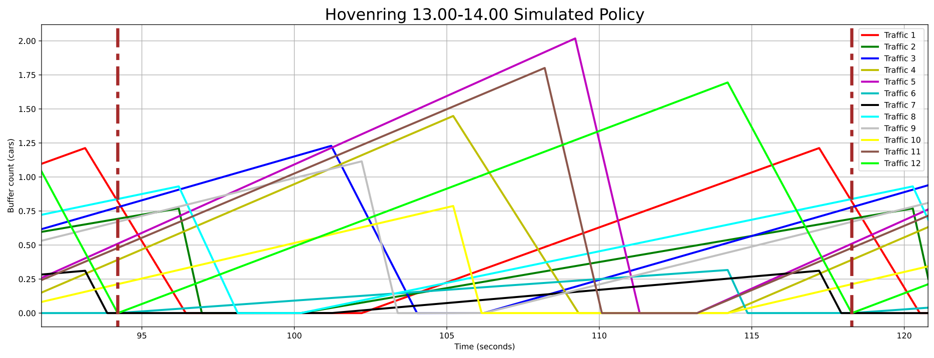


Figure C.25: Hovenring simulated buffer evolution

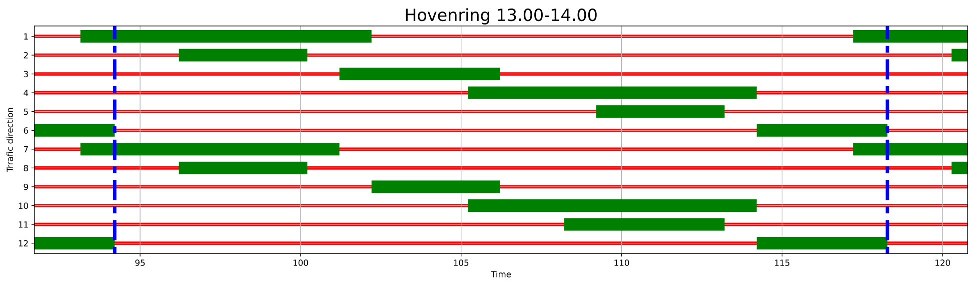


Figure C.26: Hovenring simulated cycle

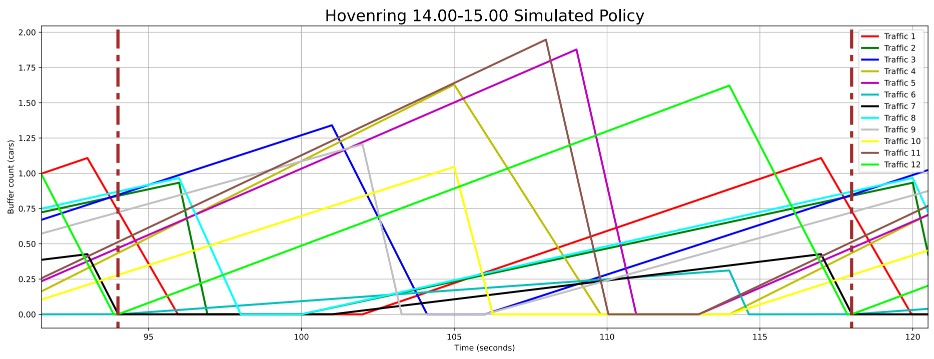


Figure C.27: Hovenring simulated buffer evolution

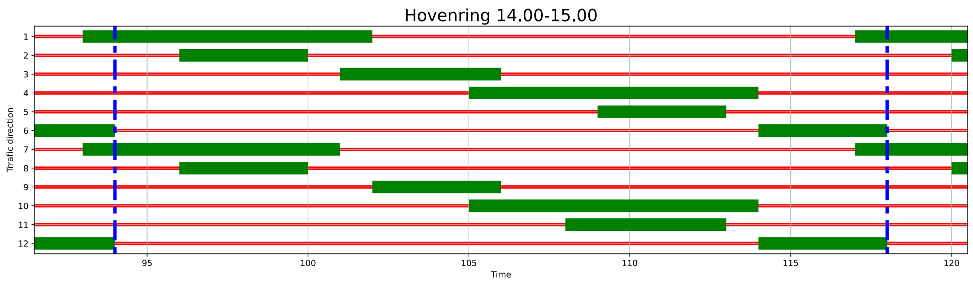


Figure C.28: Hovenring simulated cycle

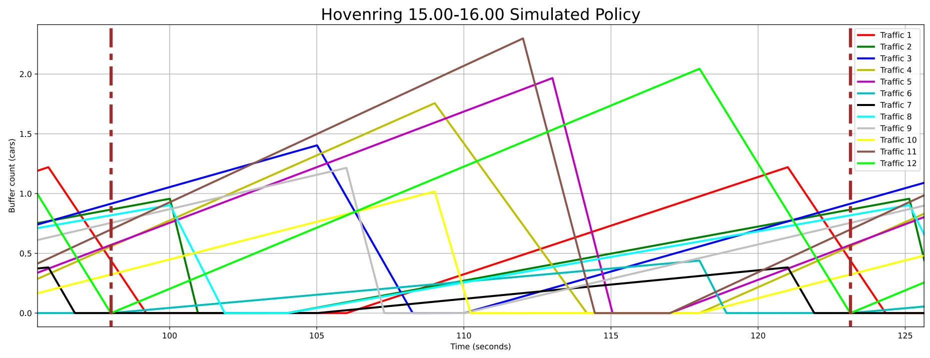


Figure C.29: Hovenring simulated buffer evolution

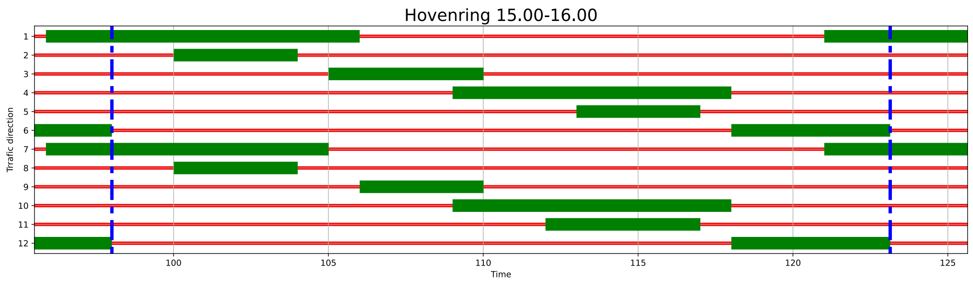


Figure C.30: Hovenring simulated cycle

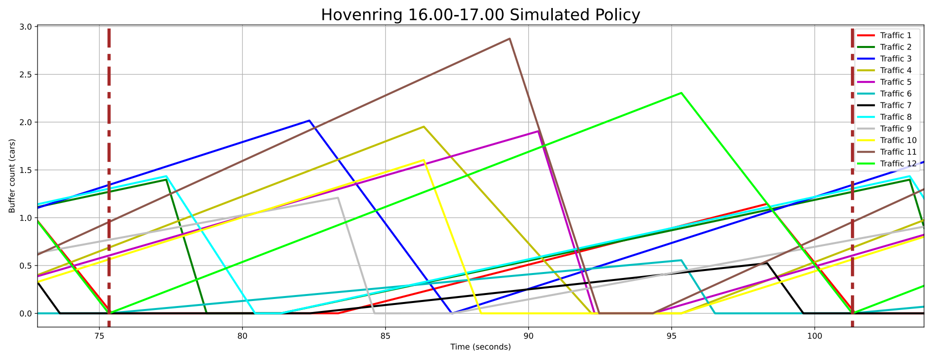


Figure C.31: Hovenring simulated buffer evolution

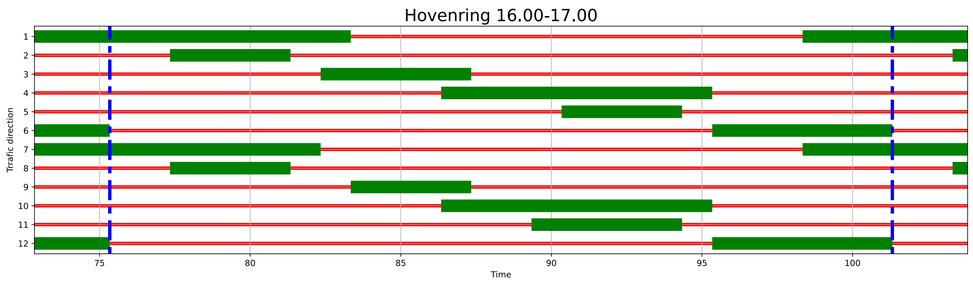


Figure C.32: Hovenring simulated cycle

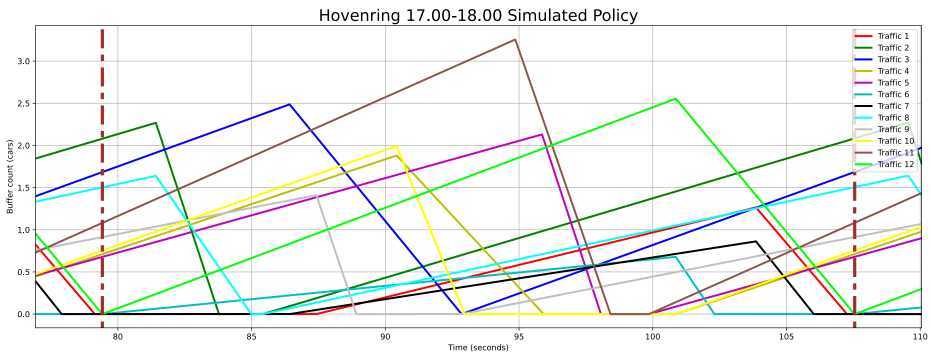


Figure C.33: Hovenring simulated buffer evolution

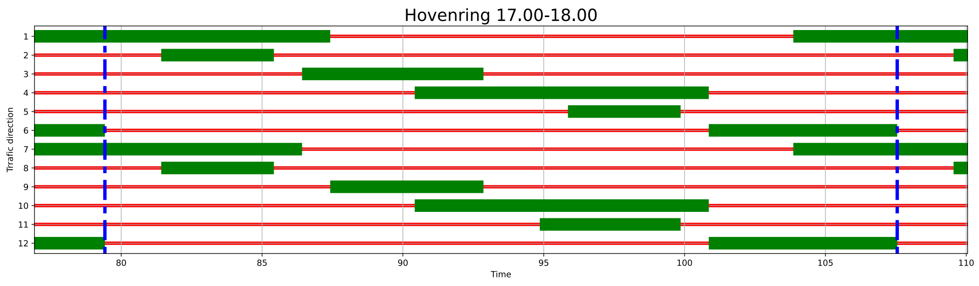


Figure C.34: Hovenring simulated cycle

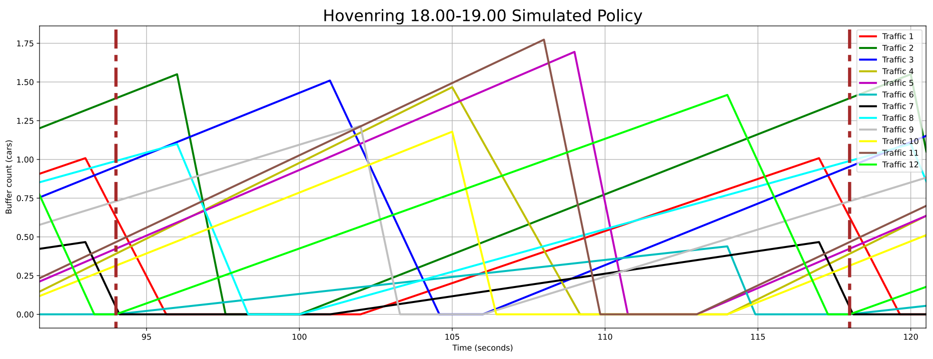


Figure C.35: Hovenring simulated buffer evolution

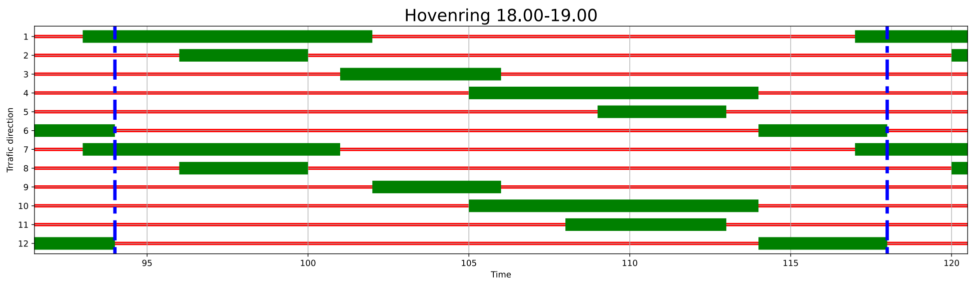


Figure C.36: Hovenring simulated cycle

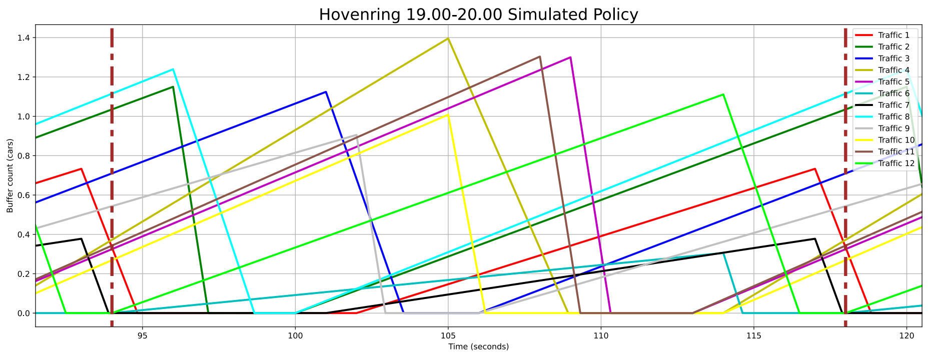


Figure C.37: Hovenring simulated buffer evolution

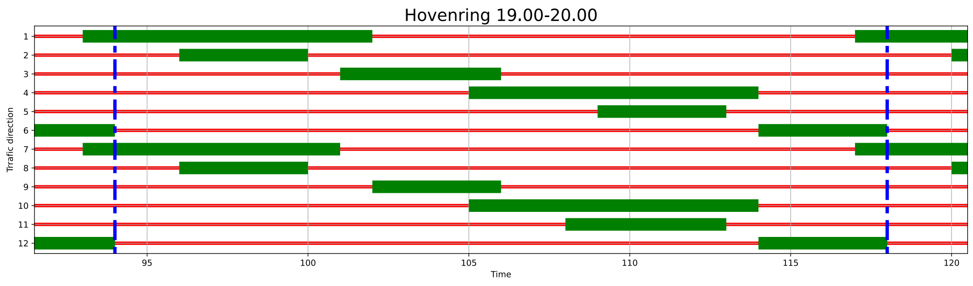


Figure C.38: Hovenring simulated cycle

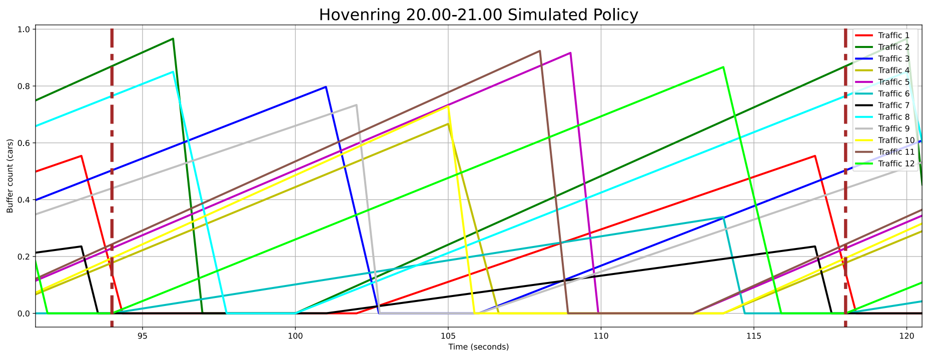


Figure C.39: Hovenring simulated buffer evolution

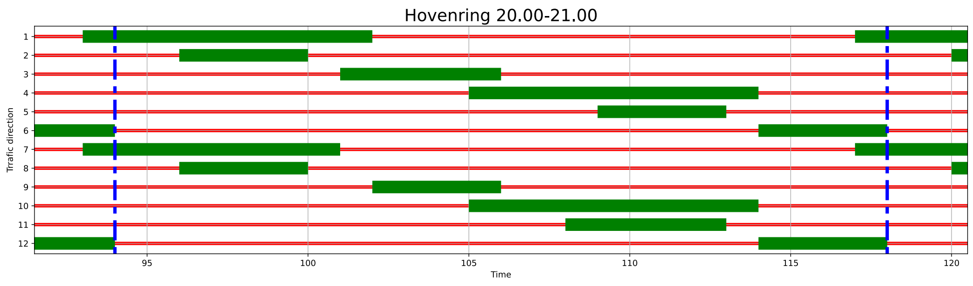


Figure C.40: Hovenring simulated cycle

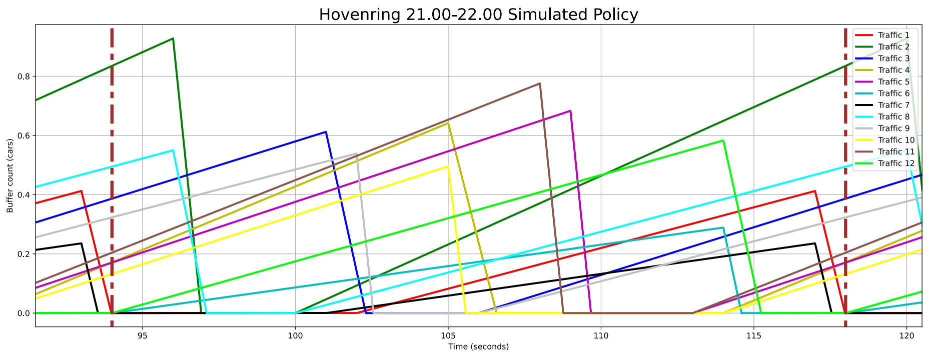


Figure C.41: Hovenring simulated buffer evolution

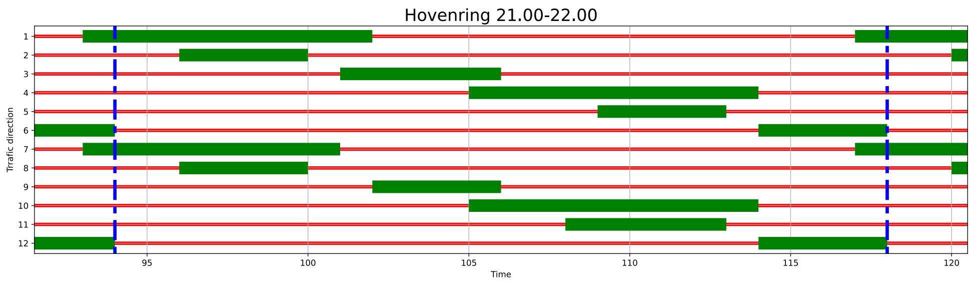


Figure C.42: Hovenring simulated cycle

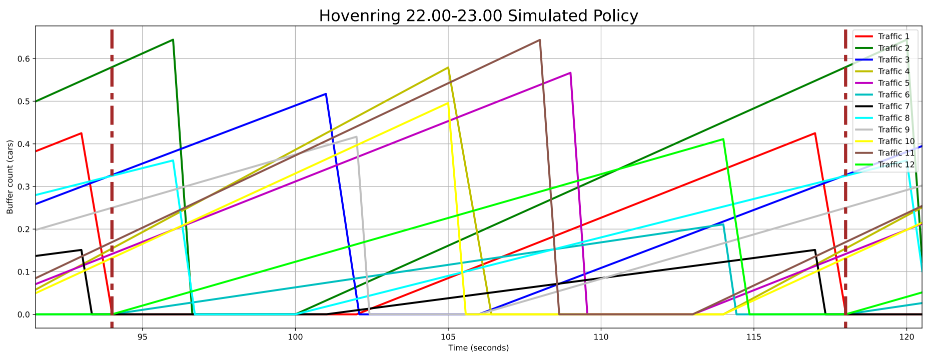


Figure C.43: Hovenring simulated buffer evolution

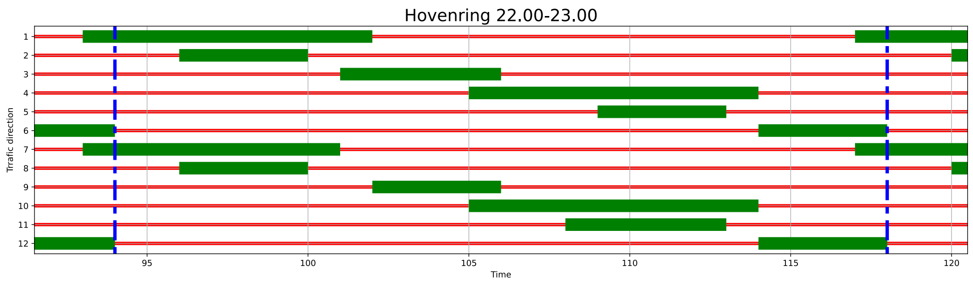


Figure C.44: Hovenring simulated cycle

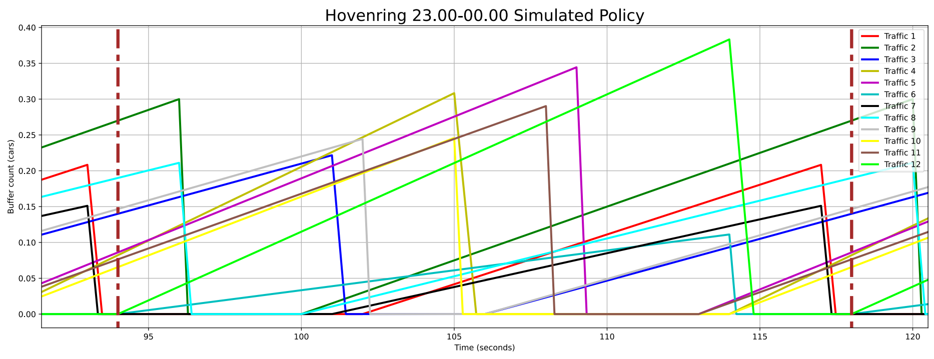


Figure C.45: Hovenring simulated buffer evolution

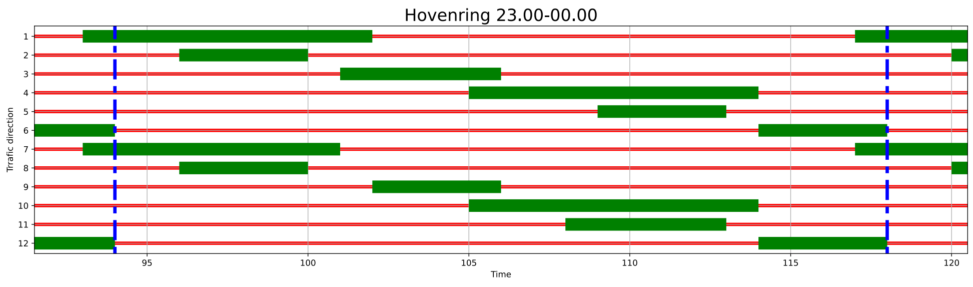


Figure C.46: Hovenring simulated cycle