Aalborg Universitet

Fra eksisterende software til modeller (P1)

Optimering af lyskryds ved brug af Reinforcement Learning

20. december - 2017

Gruppemedlemmer:
Asger Bertel
Daniel Thomsen
Hannah Lockey
Mads Faber
Magnus Kirkegaard
Niki Ewald Zakariassen
Simon Steiner

Hovedvejleder:
Tobias S. Jepsen

Bivejleder:
Chunfang Zhou

Projektgruppe:
A325b

Antal sider:
50 sider

AALBORG UNIVERSITET
STUDENTERRAPPORT

Abstract

This report examines the problem of oversaturation of traffic in signal-controlled intersections. The first part deals with the fundamental strategies of resolving traffic and the different parameters, to assess effectiveness and fairness of the traffic. The report presents data from a specific traffic light in Aalborg, which is used as a basis, to examine the greater problem. As a tool to make traffic lights more efficient, a reinforcement learning agent is introduced. To evaluate the results of the RL agent, a simulation of the traffic light is programmed and a basic time-based and traffic-based strategy is used to compare the results.

Forord

Denne rapport er udarbejdet i forbindelse med P1-projektet på Aalborg Universitet, som et svar på problemstillingen omkring intelligente lyskryds. Rapporten er udarbejdet af medlemmerne i gruppe A325b, i perioden fra d. 9 oktober til d. 20. december 2017. Rapporten gennemgår gruppens arbejdsmetodik, de valgte problemafgrænsninger og passende konklusioner heraf.

Indhold

1 Introduktion

Antallet af biler på vejene har i mange år været stigende, og i perioden mellem 2010 og 2016 er der kommet 16% flere personbiler på vejene [1]. Som en konsekvens af dette, bliver trafikken i storbyerne tættere, og kravene til at kunne afvikle trafikken effektivt bliver større. Når vejene bliver præget af tæt trafik medfører dette at fremkommeligheden for den enkelte bilist bliver kraftigt nedsat, og at trafikken i mange tilfælde kan næsten gå i stå. Behovet for effektive lyskryds i større byer er særligt stort i forhold til sikker og hurtig afvikling af trafikken. Tilstedeværelsen af lyskryds kan være en problemstilling for den enkelte bilist på grund af ventetider. [2]

Når myldretiden rammer et lyskryds er det meget naturligt, at der opstår tæt trafik og der bliver dannet kø, da ankomstraten bliver større end afviklingsraten grundet den pludseligt store mængde biler. [2]

For at optimere trafikafviklingen i eksempelvis myldretiden, kræves der mere intelligente, og derigennem mere effektive systemer til lyskrydsene. Intelligente signalanlæg er systemer, der på baggrund af trafikken nu og tidligere, kan regulere og optimere trafikflowet i et lyskryds. Dette sker ved at lade lyskrydset se og forudsige bilers ankomst til krydset [4]. Denne type systemer implementeres for at nedsætte ventetiderne, ved at adaptere signalanlægget til den specifikke situation. [4]

For at være optimal, er det nødvendigt at et intelligent system kan tage højde for flere parametre end bare effektivitet. Hvis den eneste parameter er effektivitet, kan lyskrydset teoretisk set lade den travleste vej være åben, indtil den anden vej bliver den travleste. Dette er grundet, at der ikke kommer til at være nogen ventetid i form af gult lys, acceleration og deceleration. I et scenarie som dette er der ikke taget højde for fairness, hvilket i sig selv har stor betydning for den enkelte bilist. Begrebet fairness er dog ikke entydigt og klart i alle tilfælde, eftersom antallet af biler på vejene i et kryds kan være af forskellig mængde.

I det følgende undersøges opbygningen af signalanlæg i trafikken, kødannelse i trafikken omkring lyskryds, afviklingsstrategier og hvilken rolle fairness spiller i trafikken. Ydermere undersøges det hvordan trafikforholdene i Aalborg Kommune er, samt hvordan trafiktællinger fra Aalborg Kommune kan bruges i sammenhængen.

2 Problemanalyse

2.1 Kødannelse omkring lyskryds

Når der snakkes om køer i forbindelse med lyskryds, forstås det som en række af biler, der befinder sig før eller efter et lyskryds. Det gælder at bilerne enten står helt stille, eller bevæger sig langsomt mod krydset for at blive afviklet. Køer er som udgangspunkt ikke altid problematiske når de opstår i forbindelse med lyskryds. Når antallet af ankomne biler i en kø overskrider det antal biler der kan afvikles i løbet af en grønperiode, sker en ophobning af biler og denne ophobning gør, at køen bliver et problem. [7]

I tæt trafikerede områder er kødannelser omkring trafikkryds ingen ualmindelighed, og det viser sig at de kan have store økonomiske og klimarelaterede konsekvenser [2]. For at beskrive køer snakker man om "traffic flow"som interaktionen mellem biler i trafikken, og hvordan deres handlinger påvirker hinanden. Det kan bruges som et redskab til at beskrive densiteten, afviklingen og samtidig hastigheden af biler i et givet område [8]. Når en bil ankommer til et lyskryds der signalerer grønt, kan den køre lige igennem. Når den derimod bliver signaleret til stop, medfører dette en decelerations-fase, tomgangs-fase og accelerationsfase. Ifølge vejdirektoratet er tomgangsforbruget for en personbil under normale forhold mellem 0.9 - 1.3 liter brændstof pr. time [2]. Disse brændstofmængder afhænger selvfølgelig af motorens temperatur og størrelse. Når biler decelererer svarer benzinforbruget til tomgangsforbruget, men de energimængder som den tidligere har opbygget, skal nu opbygges igen i accelerations-fasen [2]. Nogen vil mene at den største bekostning er den tid som folk mister ved at vente ved rødt lys. [2]

Køer i lyskryds opstår ofte på grund af menneskelige fejl, der forårsager en større tilvækst af biler end afvikling [9]. Når man snakker om menneskelige fejl, har afstanden og ventetiden mellem hver enkelt bilist en stor betydning for hvor mange biler der kan afvikles i et lyskryds. I praksis vil forsinkelser som forsinket start mellem biler have en betydning for hvor mange biler der kan nå at blive afviklet i en grøn periode. [9]

For at kunne forstå køens udvikling, kan man se på raten for bilernes ankomst til en kø og deres afvikling. Hvis der er flere biler som ankommer til en kø, end der bliver afviklet, vil køen være i vækst. Hvis antallet af biler der bliver afviklet derimod er større end mængden der ankommer til køen, bliver køen mindre. Det optimale er derfor at maksimere afviklingen af biler, hvilket vil øge antallet af afviklede biler i forhold til antallet af biler som ankommer til køen. [7]

2.2 Opbygning af signalanlæg

2.2.1 Signalbilleder

Man bruger i dag mange forskellige metoder til trafikafvikling afhængigt af hvilke områder man befinder sig i. I signalregulerede kryds er trafikken styret af lyskurver, der som minimum har henholdsvis en rød, gul og grøn lanterne. Lanternernes skift er låst fast så der ikke kan springes en gul fase over i et skift mellem grøn og rød. Tiden det tager for lyskurven at lave et fuldt omløb fra grøn til grøn kaldes grøntiden, og den kan generelt variere fra mellem 35-110 sekunder. Grunden til at omløbet varierer er fordi størstedelen af lyskryds er trafikstyrede. Gultiden dækker over perioden hvor lyset skifter mellem grøn og rød. I Danmark er gultiden som standard 4 sekunder. [5]

2.2.2 Sensorer

Når afstandene mellem signalanlæg er relativt store, er styringen af dem ofte afhængig af trafikken omkring dem, og krydset er dermed trafikstyret [32]. Den ankommende trafik kan registreres med forskellige sensorer. En af de mest typiske metoder til detektering af trafikken, er brugen af induktionsspoler. Metoden fungerer sådan, at der bliver lagt 20-50 spoler ned i asfalten fra 120 meter væk fra lyskrydset og op til stoplinjen [5]. Spolen danner et elektromagnetisk felt når biler kører hen over spolen, og dette signal bliver sendt fra sensoren til styreapparatet [32]. Dette fortæller at der kommer en bil, så den ved hvornår den kan skifte fra rød til grøn hvis der eksempelvis ikke kommer biler i de krydsende vejbaner, eller forlænge grøntiden [5].
Mere moderne teknikker indebærer brugen af kameraer og radarteknologier til at tælle biler og kortlægge kølængder [32]. Signalanlæggene kan agere på baggrund af denne data i realtid, og dermed forsøge at undgå lange køventetider. Denne type trafikkryds betegnes som uafhængige, eftersom beslutningerne der bliver taget ikke afhænger af andre ting end trafikken som den selv kan se. [6]

2.2.3 Styreapparater

For at kunne udnytte denne sensordata er langt de fleste lyskryds udstyret med et styreapparat, der kan anmode om signalskift eller forlænge grøntiden. Denne beslutning bliver taget på baggrund af noget logik der sidder i styreapparatet. En mulig situation kunne være at der bliver registreret en bil i en retning hvor krydset signalerer rødt, og der kommer ikke nogen biler i den anden retning. I dette tilfælde kan styreapparatet anmode om signalskift, og dermed vende signalet. [5]

2.3 Afviklingsstrategier

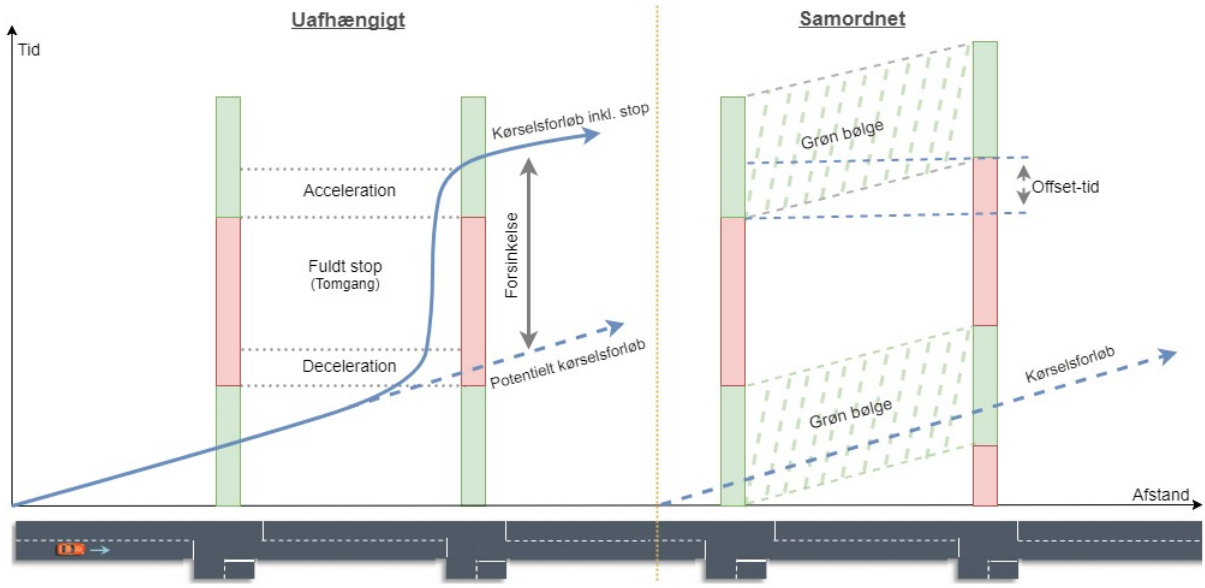
For at styringsapparatet kan tage beslutninger på baggrund af sensordataen, skal den bruge en afviklingsstrategi, som dikterer hvilken handling styringsapparatet skal foretage sig. En afviklingsstrategi betegner en enhed, som tager en trafiktilstand som input og giver en handling som output. Inputtet kan bestå af information som tid, tilstedeværelsen af biler og antallet af biler i hver vejbane. Enheden indeholder en strategi, som på baggrund af den givne trafiktilstand kan give et output i form af et signalskift eller en grøntidsforlængelse.

2.3.1 Klassiske afviklingsstrategier i signalanlæg

I storbyer og på travle veje bliver der brugt en del forskellige teknikker til at afvikle biler. Nogle lyskryds benytter sig af forudbestemte tidsintervaller, dette er dog ikke længere den eneste afviklingsmetode [32]. Ideen bag dette er, at man ved hjælp af indsamlet data, finder de tider på døgnet, hvor der er meget trafik og mindre trafik. Ud fra dataen bliver der udregnet hvad man mener, er de bedste intervaller til at afvikle trafikken med. For eksempel om morgenen hvor der er meget travlt, er intervallet for grønt lys på den ene vej måske 1 minut og den anden vej, der måske er mindre travl, har et interval på 40 sekunder [10]. Med denne metode tager man ikke højde for uregelmæssige situationer i trafikken, men man går ud fra, at det er det samme hver dag, hvilket selvfølgelig kan være et problem hvis der er en stor uventet mængde bilister, da der så også skal tages højde for acceleration og deceleration [2]. Det kan derfor ende med, at bilisterne bruger meget ekstra benzin når bilerne skal accelerere, og hvis køen er for lang, når man måske ikke igennem lyskrydset ved den første grøntid. [2]

I mange byområder på større veje kan man opleve at køre igennem det ene lyskryds efter det andet. På strækninger som denne hvor afstandene mellem krydsene er kortere, bruger signalanlæggene ofte en timer der danner en såkaldt "Grøn Bølge"ved at forskyde grøntiderne [10]. Signalanlæggets cyklus er beregnet på baggrund af afstandene til de omkringliggende lyskryds, og hastighedsbegrænsningen på vejstykket [10]. Denne type regulering kaldes samordnet, eftersom signalanlæggene afhænger af hinanden. Forskellene mellem den uafhængige og samordnede afviklingsstrategi fremgår i Figur 1. [11]

Signalanlæg med en kombination af elementer fra samordnede og uafhængige er i nyere tid blevet mere almindelige. Denne afviklingstype henvises til som adaptiv signalstyring, og tillader de enkelte lyskryds i et større netværk at indsamle data med deres respektive sensorer. Signalanlæggene kan agere på baggrund af denne trafikdata, og dermed styre dens cyklus i realtid.



Figur 1: *Illustration af forskelle mellem uafhængige og samordnede signalanlæg. [11]*

2.3.2 Fuzzy logic

For at implementere en adaptiv løsning kan man gøre brug af et forudbestemt regelsæt, som tager hensyn til forskellige trafiksituationer. Man kan bruge fuzzy logic til at udtrykke det ønskede regelsæt ved en kæde af beslutninger. [12]

Fuzzy logic adskiller sig specielt fra traditionel logik ved definitionen af sandhedsværdier. I traditionel boolsk algebra arbejder man med sandhedsværdier som kan være enten 1 eller 0 - sandt eller falsk. I fuzzy logic kan disse værdier repræsenteres ved ethvert reelt tal i intervallet [0,1]. Det vil sige at man kan arbejde med flere grader af sandhed som eksempelvis 'meget sand' og 'næsten falsk'. I et trafiklys kan man så beskrive antallet af biler i hver vejbane som en række fuzzy udtryk som 'få', 'mange' og 'temmelig mange'. Disse udtryk er defineret ud fra et interval som afgør hvilken sandhedsgrad der svarer til et bestemt antal biler. Eksempelvis kan 'mange biler' dække over 10-30 biler. Her kunne 15 biler give sandhedsværdi på 0.8 for 'mange biler'. En grafisk illustration af dette ses på Figur 2. [12]

For at kunne lave logiske slutninger ud fra fuzzy variable, kan man bruge logiske operatorer som kan defineres ved:

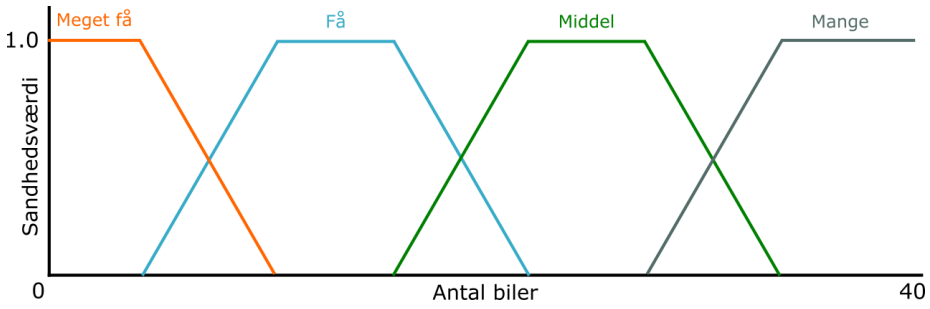
$$x \wedge y = min(x, y)$$

$$x \vee y = max(x, y)$$

$$\neg x = 1 - x$$

Heraf er min(x,y) og max(x,y) henholdsvis den mindste og største værdi af parametrene x og y [14]. Man kan fra dette opstille et regelsæt ved hjælp af en række af beslutninger, som bestemmer et udfald for hver mulig kombination af input. I en model af et trafiklys kan man tage tætheden af biler i hver vejbane som input. Disse tal konverteres til en fuzzy værdi, som bruges i en række af beslutninger som eksempelvis: Hvis der er mange biler som kører over for grønt, og få biler som holder for rødt, så forlæng grøntiden meget. Outputtet er ligeledes en fuzzy værdi som konverteres til et konkret antal sekunder. [13]

Fuzzy logic gør det let at implementere en intuitiv løsning, hvor det ellers er svært at opstille en matematisk model [15]. Metodens funktionalitet er dog dybt afhængig af de værdier, som angiver intervallet for hver sandhedsværdi. Disse fastsatte værdier kan forårsage ineffektiv adfærd, hvis den samlede mængde af trafik varierer meget [13]. Hvis man skal finde tilfredsstillende værdier manuelt, kræver det eksperimentelle justeringer, som kan være tidskrævende [16]. Desuden kan de optimale værdier variere fra kryds til kryds afhængigt af den totale mængde trafik.



Figur 2: *Grafisk repræsentation af sandhedsværdier. Y-aksen repræsenterer sandhedsværdien og x-aksen antallet af biler.*

2.3.3 Optimeringsalgoritmer

Den optimale styring af et trafiklys kan defineres ud fra en funktion, som finder den optimale løsning samtidig med, at den tager hensyn til faktorer som ventetider og fairness. Der er en række forskellige optimeringsalgoritmer, som kan styre et lyskryds, og disse algoritmer har hver deres svagheder og styrker. Problemet ved dette er, at der er forskellige løsninger til det samme problem, hvor det kan være svært at finde én enkelt korrekt løsning. Derfor kaldes dette slags problem for et optimeringsproblem. Det er målet, at finde den bedst mulige optimeringsalgoritme til det enkelte lyskryds, der kan bestemme de grøntider, som giver de bedst mulige værdier med hensyn til ventetider, fairness og antallet af afviklede biler.

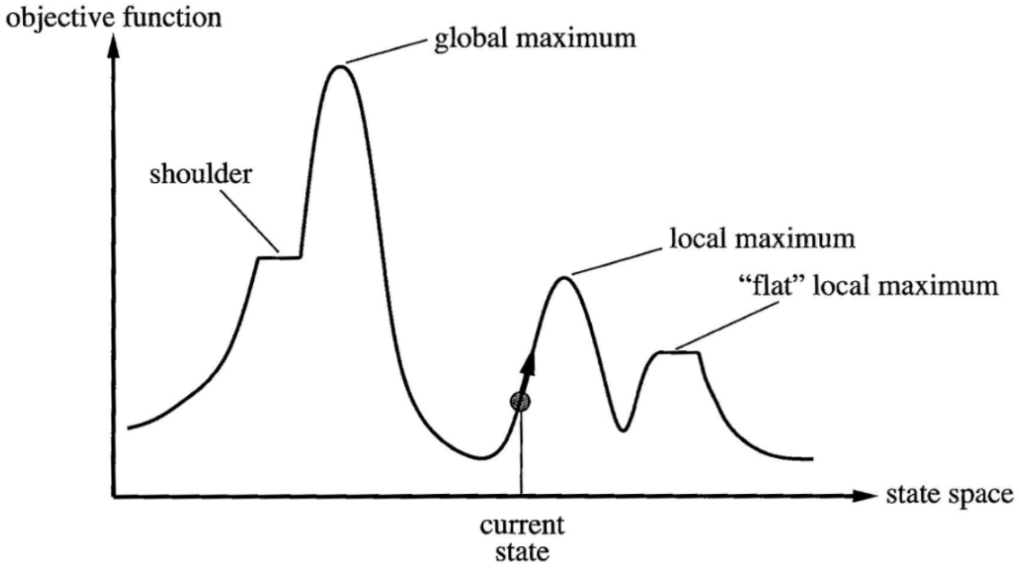
Som tidligere nævnt kan det være tidskrævende at skulle finde gode definitionsværdier manuelt for de intervaller, som fuzzy variabler dækker over. Man kan i stedet gøre brug af optimeringsalgoritmer, til at finde de værdier, som giver bedst mulig trafikafvikling.

En genetisk algoritme er en algoritme der for eksempel laver 100 tilfældigt genererede prototyper. Ud fra disse protyper ser den hvem der klarer sig bedst i softwaren og de 50 dårligste kasseres [17]. Efter dette "parres"de resterende prototyper indtil der er 100 prototyper igen. Denne cyklus gentager sig om og om igen indtil brugeren ikke ser forbedring mere. [17]

Der er to former for genetiske algoritmer: En multi-objektiv genetisk algoritme (MOGA) som er en genetisk algoritme der har flere løsnings muligheder og flere faktorer at rette sig til, og en single-objektiv genetisk algoritme (SOGA), som kun har én enkel løsning den kan finde frem til. Med genetiske algoritmer tildeler man ranks til individerne ud fra fitness point [17]. Med denne algoritme kan man komme frem til løsninger, som er delvist subjektive, som for eksempel traffic flow i et lyskryds. En SOGA kan udregne den hurtigste måde at få flest biler igennem et lyskryds, men en MOGA kan indvikle flere parametre som ventetid, miljø venlighed, fairness mm. hvilket ikke finder den hurtigste måde at få flest biler igennem et kryds på, men den finder den mest effektive måde der er bedst for alle de forskellige parametre. [17]

Et problem med genetiske algoritmer er at antallet af mulige strategier, som kan udforskes, stiger eksponentielt med antallet af tilstande. Dette gør det svært at finde frem til tilfredstillende strategier når problemstørrelsen bliver stor. Læringsprocessen i genetiske algoritmer er også ineffektive, fordi de ikke lærer direkte fra hver enkelt handling, men i stedet bedømmer hver prototypes samlede præstation. Derved ses det ikke direkte hvilke handlinger fører til gode eller dårlige resultater.

En hill climbing algoritme er en algoritme der beskriver hvordan et stykke software lærer at forbedre sig selv ved at gå et skridt til hver retning og derefter tage skridtet mod den retning der er højest. En hill climbing algoritme kan aldrig gå ned af så når den kommer op til der hvor den tror er det højeste punkt, stopper den. En hill climbing algoritme kan derfor finde det lokale maksimum og dette kan betyde at softwaren potentielt kan være bedre, men ikke kan blive bedre fordi den først skal blive dårligere før den bliver bedre og eftersom den ikke kan tage et skridt nedad, kan dette ikke lade sig gøre. Som set på Figur 3 er den nuværende tilstand på vej op af en bakke, som ender i et lokalt maksimum. Denne tilstand når derfor aldrig det globale maksimum [19]. Dette kunne bruges som en nemmere form for optimering til et lyskryds, men den ville ikke kunne finde den bedst mulige metode hver gang.



Figur 3: *Illustration af en hill climbing algoritme. [21]*

En måde at give sig i kast med at optimere afviklingen og køtiden i lyskryds, er at anse det som et optimeringsproblem. Det vil betyde at hvis man fryser et lyskryds i et givent sekund, og man har defineret klare linjer for hvad den 'bedste' løsning er, kan man i teorien gennemsøge alle mulige måder som lyskrydset kan agere på og finde den bedste indstilling. Som udgangspunkt ville den 'bedste' løsning afhænge af mere end bare antal biler afviklet. For et lyskryds ændrer miljøet sig hele tiden, hvilket også betyder at man af praktiske årsager skal gennemsøge søgeområdet efter den bedste løsning hvert eneste øjeblik miljøtilstanden ændres. Disse forskellige faktorer vil både øge søgeområdet, og antallet af gennemsøgninger af søgeområdet. Dette gør det altså klart at en mere intelligent løsning er nødvendig.

2.3.4 Reinforcement learning

Reinforcement learning (også kaldet RL) er baseret på en såkaldt 'agent' som kan foretage handlinger og observere den tilstand som den befinder sig i. Når agenten har foretaget en handling, observerer den sin tilstand og får tildelt en belønnings- eller strafværdi. Ud fra dette vurderer agenten sin handling, og justerer de parametre som angiver hvordan der tages beslutninger. Målet er så at lære at vælge de handlinger som giver optimale belønningsværdier. [20]

Der tages udgangspunkt i et eksempel hvor en RL algoritme anvendes i en model af et enkelt lyskryds. Ved at observere sin tilstand vil agenten få adgang til information om trafiktætheden i hver vejbane, samt tiden siden sidste faseskift. Hvert sekund kan agenten foretage en af to handlinger: skifte fase eller forlænge den nuværende fase. Belønningsværdien bestemmes ud fra bilernes ventetid samt en fairness faktor. [20]

Trafikstyrede algoritmer kan generelt tilpasse valget af grøntider til den aktuelle trafik. Potentielt set kan en RL algoritme endvidere tilpasse sig store ændringer i de omgivelser den opererer i. Dette kan ske ved at agenten løbende opdaterer sine beslutningsparametre, ud fra vurderinger af sine handlinger. Dette sker ved brug af en metode kaldet value iteration, som tillader agenten på baggrund af belønningsfunktionen og udfaldssandsynlighederne at tildele en handling et antal point. Disse point beskriver det summerede antal point som agenten kan opnå ved at handle optimalt fra dens nuværende tilstand over tidshorisonten H. Disse værdier kan tolkes som den forventede fremtidige belønning ved at gå til tilstanden.

Som tidligere nævnt bruger en genetisk algoritme erfaringer ineffektivt, fordi den kun vurderer summen af alle handlinger som en prototype har foretaget sig. En reinforcement agent vurderer derimod værdien af hver enkelt handling, og den kan lære fra sine erfaringer mere effektivt end en genetisk algoritme [18]. En reinforcement learning algoritme kan desuden også implementere fairness direkte ved at sætte en grænse for den maksimale og minimale grøntid, og har derfor en fordel i forbindelse med trafikstyring. [20]

2.4 Fairness

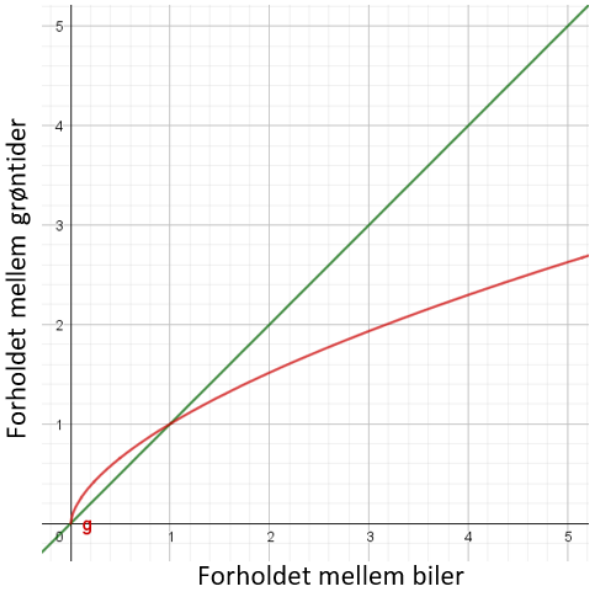
Vi ønsker at udvikle en intelligent styring af lyskryds, så vi kan optimere trafikafviklingen. Hvis den eneste parameter vi vurderer løsningen på, er antallet af biler afviklet, kunne en løsning hurtigt blive at lade hovedvejen være åben konstant, da dette ville få afviklet flest biler i minuttet. Dette fradrager dog muligheden for bilerne på sidevejene, at komme ind på hovedvejen og de må for evigt vente bag rødt lys, hvilket ikke ville være fair. Fairness er derfor et relevant aspekt at tage op til vurdering, men det er svært at nå frem til en specifik definition for fairness. Fairness i lyskryds kan derfor defineres som en balance mellem bilafvikling og den forventede ventetid for bilisterne fra hver vejbane.

Fairness er ikke entydigt eller konstant i alle tilfælde. I et lyskryds med en hovedvej og små sideveje, Må det være fair at de små sideveje skal vente længere bag rødt lys end hovedvejen, grundet at der typisk kommer mange flere biler, der skal afvikles på større veje. Alligevel må vente tiden på de små sideveje heller ikke nå over en vis grænse, da dette vil påvirke fairheden i krydset. Til gengæld i et lyskryds hvor den gennemsnitlige fordeling af biler er nogenlunde ens, er det i de fleste tilfælde ikke fair at den ene retning får mere grøn tid end den anden.

I moderne lyskryds med meget trafik er det yderst vigtigt, at lyskrydset kan afvikle så mange bilister som muligt på kort tid. Det er dog ikke det eneste, der gør et lyskryds optimal. Det mest optimale lyskryds skulle kunne afvikle så mange biler som muligt på kort tid, og samtidig sørge for at undgå lange ventetider for de andre vejbaner. Problematikken ved lange ventetider bag rødt lys er ikke kun, at det er unfair, men det kan også bidrage til en større mængde trafikuheld. Da en af grundene til bilister kører over for rødt, er at de mister tålmodigheden og prøver at køre igennem lyskrydset mens det er gult. [22]

Vi ønsker at optimere antallet af biler afviklet i minuttet, men uden at gøre det unfair, og risikerer et øget antal af trafikuheld. En mulig strategi kunne være, at desto større forskellen mellem antallet af biler pr. minut fra hver retning er, desto større må forskellen i ventetiden være. En lineær tilgang til dette vil nok ikke være praktisk, da tilfælde med meget store forskelle mellem antal biler i hver retning ville resultere i meget lange ventetider. Derfor bør forholdet mellem dette, være mere lignende en potensfunktion, som i starten stiger hurtigt i takt med at forholdet ændres, men senere hen ved store forhold, bliver mere flad.

På Figur 4 repræsenterer x-aksen forholdet mellem antal af biler som har grønt og antallet af biler som har rødt. y-aksen beskriver forholdet mellem grøntiderne. Den grønne linje beskriver en lineær tilgang hvor forholdet mellem grøntiden stiger proportionelt med forholdet mellem antallet af biler. Den røde linje beskriver en potentiel sammenhæng, hvor ventetiden eksempelvis bliver to gange så lang, når antallet af biler er tre gange så stort.



Figur 4: Eksempel på en mere fair sammenligning i ventetid

2.5 Case: Aalborg

2.5.1 Detektering

I Aalborg by bliver der brugt den førnævnte metode i afsnit 2.3.1, der udnytter elektromagnetiske spoler. Få steder i Aalborg bliver der brugt video- og radardetektering. Disse nye teknologier forventes i fremtiden, at blive meget mere attraktive, og især radarteknologi har nogle funktionelle og trafiktekniske fordele over både spoler og videodetektering. [33]

By og landskabs forvaltningen i Aalborg har planner om at integrere radarer flere steder i byen, og forventer i fremtiden at raderdetektering vil være at finde i de fleste lyskryds i byen. De magnetiske spoler er billige at etablere men dyre at vedligeholde, da reparation af spoler ofte kræver at vejbanen afspærres grundet arbejdet krævet for at tilgå de underjordiske spoler, og disse reparationer kan ikke foregå i perioder med frost da der skal anvendes vand i sammenhæng med skærearbejdet [33]. En defekt spole kan derfor ofte resultere i at trafikstyringen i krydset ikke fungerer optimalt over en længere periode. [32]

Et alternativ til spoler er radar- og videodetektering der er dyre at etablere, men billige at vedligeholde [33]. Teknologien kræver et stort opsættelsesarbejde, eksempelvis skal et kamera langt op i luften for at nå en tilstrækkelig rækkevidde, og for at undgå at høje biler dækker for udsigten bag dem. Ved kryds der er placeret ved veje hvor hastighedsbegrænsningen er omkring 80km/t, skal detekteringen af bilerne foregå tidligere end normalt, og derfor må man ofte placere flere master langs vejen med kameraer for at dække det ønskede vejstyk. Et tilfælde som dette ville øge anskaffelsesomkostningerne betydeligt. [32]

Elektromagnetiske spoler og radere er pålidelige i deres detektering, men de registrerer trafikken på forskellige måder. Spoler bruger metoden punkt måling hvor et kamera bruger område måling. For eksempel bliver et kamera placeret i 8 meters højde hvor den vil måle et område i trafikken. Denne metode har en ulempe, for når kameraret bliver udsat for blæst, kommer det til at sveje og det kan påvirke trafik registreringen. Hvilket kan have mange forskellige følgeeffekter som for eksempel, et køretøj ikke bliver registreret og derved bliver den grønne periode ikke forlænget. Dette kan resultere i at køretøjet bliver tvunget til at køre over for rødt og derved skaber en farezone i krydset som det også fremgår af Figur 5. [32]

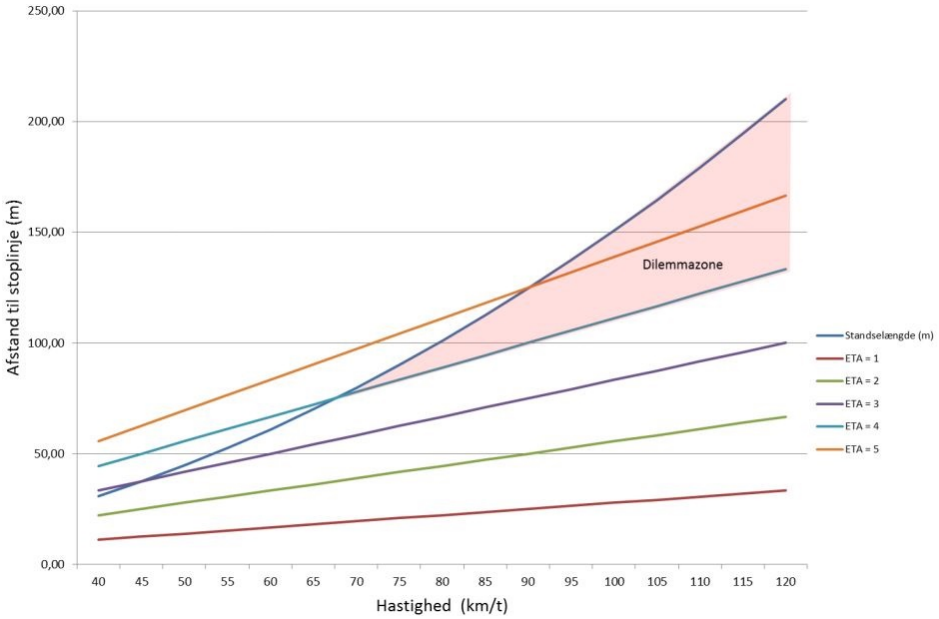
2.5.2 Afvikling

For at implementere raderer og videoregistrering og gøre dem kompatible med det nuværende system, oprettes virtuelle spoler på vejen. De virtuelle spoler har i princippet samme funtionalitet som spolerne i vejen. [32]

Ved byens fremtidig introduktion til rader teknologien, vil nye afviklingsstrategier gøres muligt, og flere trafiktekniske metoder kan forbedres som for eksempel lastbils prioritering og ulykkereduktion. [23]

Disse nye afviklingsstrategier som bliver mulige i fremtiden i sammenhæng med radere. Bliver gjort mulige på grund af, at radere har den funktionalitet, at de kan registrere og følge op mod 126 objekter på samme tid. Radarerne har en rækkevidde på op mod 300 meter som gør det muligt at registrere omkring 126 køretøjers placering, hastighed og bremselængde. Dette garanterer en større sikkerhed i krydset, da mere data er tilgængeligt. [32]

Som tidligere nævnt, er den gule periode i Danmarks kryds 4 sekunder, og hvis en bils bremselængde er længere end distancen til krydset, samt den forventede tid til krydset er længere end 4 sekunder, så ender bilisten i dilemmazone, fordi denne bil ikke kan nå at bremse før krydset og heller ikke kan nå bremse før lyset skifter til rød, som fremvist på Figur 5. For at undgå at biler ender i denne dilemmazone, vil et normalt kryds forlænge den grønne periode, indtil det er ansvarligt at skifte tilstand eller den maksimale grøntid er nået. Der efter vil tilstanden blive skiftet lige gyldigt hvad. Elektromagnetiske spoler og deres nuværende afviklingsystem har dog den problematik, at grundet den lave mængde information er det i visse tilfælde muligt, at biler fra samme bane skiftevis forlænger den grønne periode indtil den maksimale grøntid er nået, hvilket skaber farlige situationer. Raderteknologien vil kontinuerligt kunne måle på op mod 126 biler og beregne deres bremselængde og afstand til krydset i stedet for den nuværende punkt måling. Ud fra dette kan man sikre en bedre afslutning af de grønne perioder. [32]



Figur 5: Model der beskriver bremselængden og den forventede tid til ankomst ved kryds i forhold til hastighed og afstand. [23]

Foruden at sikre imod, at der bliver skiftet fra grøn til rød i upraktiske og farlige tilfælde. Så kan raderteknologien også forbedre sikkerheden på veje, hvor lyskryds har røde hvileperioder igennem hastighedsdæmpning ved først, at skrifte til grønt, når bilisten har en afstand fra krydset der er lig den bremselængde der er tilsvarende den tilladte hastighed. Dette gør hastighedsdæmpning mere dynamisk og tilpasset den specifikke bilists hastighed. [32]

2.6 Data - Trafiktællinger

For at optimere trafikken i et lyskryds, er det nødvendigt at inddrage data fra virkeligheden som for eksempel trafiktællinger. Der vil i denne rapport blive taget udgangspunkt i trafiktællinger i Aalborg Kommune, lavet af Vejdirektoratet [24]. Trafiktællingerne bruges til mange forskellige formål, blandt andet til at understøtte den overordnede vejplanlægning og til løsning af opgaver om trafiksikkerhed [24]. Datasættet indeholder trafiktællinger fra Aalborg Kommune, på lige strækninger såvel som i kryds, med og uden lysregulering. [24]

Der vil i denne rapport blive brugt dele af observationerne fra datasættet. Der vil blive set bort fra de observationer der ikke omhandler lyskryds, samt de dele af datasættet der beskriver bilernes hastighed, da vi kun interesserer os for antallet af biler der bevæger sig igennem lyskrydsene.

Datasættet indeholder flere nøgletal til beskrivelse af det gennemsnitlige antal biler, der kører igennem hvert kryds. Til at arbejde videre med datasættet fokuserer vi på nøgletallet hverdagsdøgntrafik (HDT). Dette tal beskriver det gennemsnitlige antal biler, der kører gennem krydset på en hverdag uden for juli. Juli måned er ekskluderet fra datasættet, da dette er en måned der er præget af ferie. En medtagelse af juli vil give visse skævheder i vores beregninger, da vi ønsker at optimere trafikken i de tidspunkter hvor der er størst risiko for kødannelse. Vi benytter HDT til at beregne det gennemsnitlige antal biler i timen i 22 timer af døgnet. De sidste to timer er spidstimerne; spidstid morgen og spidstid eftermiddag. Spidstimerne er de to timer på dagen hvor der er mest trafik, og det præcise tidspunkt for spidstimernes start varierer en smule fra kryds til kryds.

3 Problemformulering

Den almene billist kan i hverdagen bruge en stor procentdel af sin køretid på at holde for rødt ved signalanlæg, og i Aalborg er dette ikke en undtagelse. Dette gælder især hvis en billist skal ind i byen om morgenen, og hjem sent om eftermiddagen. Et signalanlægs valg af grøntider har betydning for effektiviteten af trafikafviklingen og der eksisterer flere forskellige styringsmetoder i lyskryds.

I øjeblikket er den mest brugte type sensor i Aalborg magnetiske spoler, som kan registrere biler i vejbanerne. I fremtiden skal Aalborgs signalanlæg indeholde radarteknologi som med høj præcision kan tælle antallet af biler og bedømme afstanden til enkelte biler. Denne information øger optimeringspotentialet og baner vej for mere avancerede afviklingsstrategier [33].

Ved hjælp af fuzzy logic kan man implementere en let-forståelig løsning, som styrer signalanlægget på baggrund af information om den aktuelle trafik, men fuzzy logic er afhængig af optimale valg for en række definitionsværdier samt et regelsæt, som begge kan være tidskrævende at finde frem til manuelt. For at imødekomme dette problem kan man bruge optimeringsalgoritmer, men her kan der opstå problemer i forhold til algoritmens tidsforbrug. Desuden kan de optimale værdier variere fra kryds til kryds, hvilket vil betyde at man skal finde frem til nye parametre for hvert signalanlæg, som denne løsning skal implementeres i.

For at skabe en mere adaptiv løsning kan man bruge reinforcement learning, som selv finder frem til afviklingsstrategier ved at vurdere værdien af dens handlinger. Den store fordel ved dette er at den er i stand til at vurdere udfaldet af tusindvis af situationer på et øjeblik, givet at man detaljeret har beskrevet belønninger ved alle dens handlinger. Den er således selv i stand til at træffe beslutninger om grøntidsforlængelse og signalskift, som giver optimal trafikafvikling.

Vi har valgt at bruge reinforcement learning til at udvikle vores agent da vi mener det er den bedste mulighed givet vores resurser og tid.

Grundet tidsbegrænsninger vælger vi at udvikle vores strategi med fokus på uafhængige kryds. Vi vil dog fortsat give forslag til hvordan vores løsning kan implementeres i et samordnet signalanlæg givet nok ressourcer og tid.

Ud fra foregående analyse vil vi undersøge følgende problemstilling:

"Hvordan kan man bruge reinforcement learning, på baggrund af radarteknologi, til at forbedre uafhængige afviklingsstrategier med udgangspunkt i Aalborgs lyskryds?"

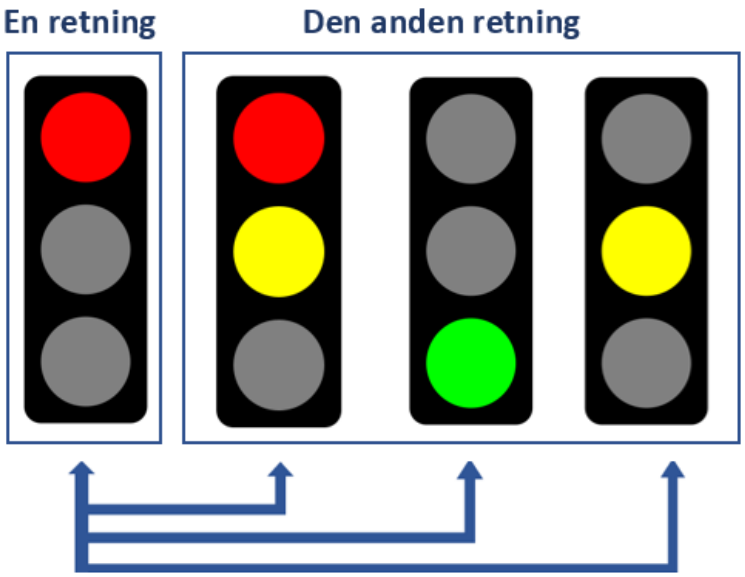
4 Problemløsning

4.1 Antagelser

I analysen blev der taget udgangspunkt i et specifikt kryds i Aalborg. Af hensyn til den tid der er tilgængelig til at arbejde med løsningen, samt sværhedsgraden ved at udarbejde det, er der lavet en række antagelser som væsentligt nedsætter kompleksiteten af krydsets signaler og trafikken omkring det. Disse antagelser er stadig valgt med henblik på at bibeholde en så nær virkelighedsgengivelse som det er muligt.

- Vejret i simulationen er altid tørt. Der er ingen årstider og temperaturen påvirker aldrig vejforholdene.
- Der tages ikke hensyn til afvikling af fodgængere og cyklister. Når cyklister og fodgængere afvikles er deres grøntider ofte flettet ind i bilernes, og ventetiden der tilføjes er derfor minimal, og vi har derfor valgt at se bort fra det.
- I krydset der blev undersøgt i casen havde 2 af de 4 retninger en separat venstresvings-lyskurv, hvoraf vi dog på grund af kompleksiteten af agenten antager, at der kun er ligeudbaner i alle fire retninger. Vi har 6 forskellige signaltilstande som fremgår af Figur 6. Illustrationen viser de 3 indstillinger lyskurvene kan være i, i forhold til hinanden, hvor de krydsende retninger altid er røde for begge lyskegler. Dette bliver altså 6 indstillinger fordi dette kan arrangeres i begge retninger.
- Simuleringen ser bort fra deceleration, eftersom implementationen var svær at gøre virkelighedstro og derudover menes det ikke, at have den store betydning for at vurdere de forskellige løsningsstrategier.
- RL agenten antager at folk altid kører 50 km/t når de ikke holder i kø, derudover antager agenten heller ikke acceleration eller deceleration.
- Der kan maksimalt afvikles cirka 3 biler pr. sekund. Denne antagelse er baseret på antagelsen om at biler kører 50 km/t (14m/s), og eftersom biler i gennemsnit er 3 meter lange og afstanden mellem biler mindst er 2 meter, kan der omtrent afvikles 3 biler på et sekund.
- De trafiktællinger som vi bruger er opdelt i 3 tidsperioder i løbet af dagen (2 spidstimer og resten). I forbindelse med simulering af trafikken er det beskrevet med en mere naturlig overgang, vores RL agent kan dog udelukkende tage højde for at den er i en af perioderne, og ikke det præcise tidspunkt.

- I forbindelse med brugen af poisson-fordeling antages det, at de nuværende biler der ankommer ikke har nogen sammenhæng med de biler der kom før og efter den . Dette er ikke altid sandt i trafikken eftersom kryds der ligger placeret tæt på hinanden, ofte vil sende en lang række biler mod krydset efter en rødperiode.



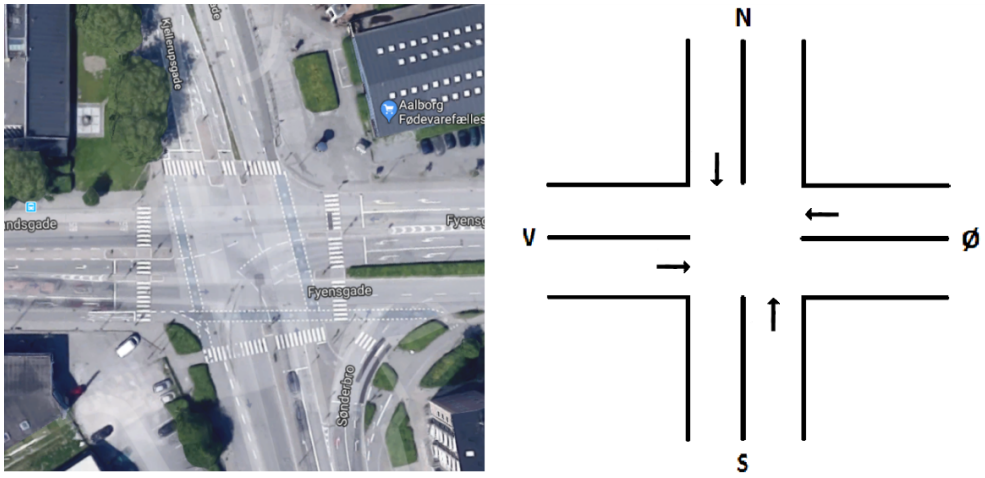
Figur 6: Illustration af de forskellige kombinationer af lyssignaler i de to retninger

4.2 Design af RL agent

Som det fremgik af analysen kan problemet formuleres som et reinforcement learning problem. I det følgende fremgår derfor elementerne som en RL agent består af. Alle mulige handlinger som agenten kan udføre, rummet af tilstande og agentens belønningssystem er alt sammen RL-agentens design, og vil blive beskrevet i de følgende afsnit:

- Handlinger
- Tilstandsrum
- Belønninger

For at kunne finde den bedste afviklingsmetode i et givent miljø, er agenten nødt til at kunne observere miljøet ved hjælp af sensorer, og dermed også vide hvordan det hele tiden ændrer sig. Et miljø er bygget op af en endelig mængde forskellige tilstande, som krydset kan befinde sig i. For et lyskryds' miljø er antallet af biler i de forskellige vejbaner, signaltilstande og om en tidsbegrænsning af grøntiden er nået, nogle af de parametre som der bør tages højde for i udviklingen af en RL agents strategi.



Figur 7: Venstre: Satellitbillede af kryds [30]. Højre: Simplificeret model af lyskrydset.

4.2.1 Tilstande

En RL agent skal være i stand til at skelne alle de relevante informationer den skal bruge i lyskrydset fra hinanden. Det vil altså sige, at et øjebliksbillede den ser, skal kunne identificeres og skelnes fra alle andre øjebliksbilleder. Disse øjebliksbilleder betegnes som tilstande, og skal defineres forud for at agenten laves. Den umiddelbare tanke når disse tilstande skal defineres er at lave dem så detaljerede som muligt, og på den måde øge agentens evne til at handle i specifikke situationer. Det er dog vigtigt at huske på at agenten skal kunne holde styr på alle disse i løbetiden af programmet, hvilket betyder at søgeområdet ønskes minimeret. Dette gøres ved at vælge detaljer om tilstandene med omhu.

Det blev i analysen beskrevet at de parametre, som RL agenten skal være i stand til at agere på baggrund af, er antallet af biler i de enkelte baner, en form for fairness, samt hvilket tidspunkt på dagen den befinder sig i. Ydermere skal agenten også vide hvilken signaltilstand den befinder sig i, for at kunne tage en beslutning omkring hvad den kan gøre. For at undersøge det umiddelbare søgeområde af disse tilstande antages det, at vi kigger på alle kombinationer af situationer med 0-126 biler i hver vejbane (Sensorerne kan maksimalt detektere 126 biler, og der sidder en ved hver vejbane). Tiden som den har været i denne signaltilstand kan være fra 0-120 sekunder. Tidspunktet på dagen er opdelt i 24 timer og på baggrund af afgrænsningen af vores kryds, som beskrevet i overblikket, har det 10 forskellige signaltilstande.
Det fremgår af Figur 7, at modellen som der tages udgangspunkt i, har 2 forskellige vejbaner, i 4 forskellige retninger. For at lave en agent der har dette niveau af detalje vil det kræve noget lignende den følgende mængde hukommelse der fremgår af følgende beregning:

$$127^4_{bilerL} \cdot 127^4_{bilerV} \cdot 10_{signaltilstande} \cdot 121_{sek.} \cdot 24_{timer} \approx 1,96 \cdot 10^{21}\ tilstande$$
$$\frac{1.96 \cdot 10^{21}\ tilstande \cdot 8\ bits}{10^{21}} \approx 16\ zettabytes\ hukommelse$$

Denne størrelse af søgeområde er en smule urealistisk, så det er nødvendigt at simplificere disse tilstande. En måde at gøre dette på er at opdele de enkelte tilstande i intervaller. Dette kunne se ud som følgende:

$$Biler\ ligeud = \{[0;0],[1;3],[4;8],[9;15],[16;25],[26;126]\}$$
$$Signalperioder = \{[0;15],[16;119],[120;\infty[\}$$

Tilstandenes intervaller er valgt med henblik på at give agenten alle de nødvendige informationer, men samtidig kun det absolut nødvendige. Agenten skal vide om vejbanen er helt tom, helt fuld, eller derimellem. Grunden til at intervallet af biler går til 126 biler er, at sensorerne beskrevet af Aalborg Kommune er i stand til at observere 126 biler af gangen. Tiden i signaltilstanden skal bruges til at definere en minimumtid for at være i en signaltilstand, samt en maksimumtid for at være i en signaltilstand. Tidspunktet på dagen opdeles i de to spidstimer samt resten af dagen. Med tilstandene opdelt i intervaller mindskes det til at strække sig over følgende antal tilstande og søgeområde:

$$6^4_{bilerL} \cdot 6_{signaltilstande} \cdot 3_{signalperioder} = 23.328\ tilstande$$
$$\frac{23.328\ tilstande \cdot 8\ bits}{10^6} \approx 0,19\ kilobytes\ hukommelse$$

4.2.2 Handlinger

For at tillade agenten at styre signalanlægget på baggrund af trafikken, skal den have adgang til at udføre handlinger i krydset. Som det blev beskrevet i afsnittet om signalanlæg i afsnit 2.2 i analysen, har styreapparatet adgang til at forlænge grøntiden samt lave et signalskift. Dette formuleres for vores agent sådan, at den hvert eneste sekund undersøger tilstanden som krydset er i, og på baggrund af dette har mulighed for enten at forlænge grøntiden med et sekund, eller lave et signalskift. Dette efterlader agenten med 2 mulige handlinger i alle situationer. Med disse to handlinger bliver signalskiftene dog låst i et fast omløb, og den har derfor kun mulighed for at skifte signaltilstand, når krydset befinder sig i en Grøn/Rød fase, hvor selve krydset selv står for at skifte fra grøn fase til gul og så til rød fase når dette anmodes i den grønne fase og ligesådan fra rød til grøn.

Det blev i forrige afsnit omkring tilstande beskrevet, at der var en minimumtid som krydset skulle være åbent i, før krydset havde lov til at skifte signaltilstand igen. Denne regel bliver dog kun overholdt når det er relevant. Dette betyder at krydset gerne må overskrive minimumstiden, hvis alle de grønne baner er tomme. Foruden har vi også en maksimal tid et kryds må befinde sig i en tilstand, før den tvinges til at skifte signaltilstand, for yderligere at håndhæve fairhed i krydset, men denne regel overholdes også kun når relevant. Hvis krydset bliver helt tomt, og alle baner derfor er tomme for biler, bliver uret der tæller tid i tilstand resettet til sekundet over minimumsgrænsen, for på denne måde tillade krydset at forblive i en signaltilstand så længe den ønsker, når det gælder at krydset har været tomt indenfor de sidste 2 minutter.
Når agenten vurderes i simulering i forhold til en trafikstyret og tidsstyret løsning, er signaltilstandende i alle tre tilfælde låste i et fast omløb hvilket på den måde giver en fair sammenligning.

4.2.3 Belønninger

Idet agenten observerer lyskrydset, skal den være i stand til at vurdere udfaldet af de handlinger som den foretager. For at gøre dette skal man i første omgang beskrive dens belønning for at befinde sig i bestemte tilstande. Agenten bedømmes primært på hvor mange biler der holder bag rødt, samt bag gul og grønt. På denne måde bliver den straffet for at holde en vejbane åben med mange biler i, men belønnes samtidig for at holde baner med biler åben. Straffen og belønningen tildeles i forhold til den midterste værdi i de førnævnte intervaller af biler. Hvis der eksempelvis er 5 biler der holder for rødt, bliver de straffet med -6 point, da intervallet er [4-8], eller hvis de befinder sig i banen med grønt belønnes den med 6 point, som det også fremgår herunder:

$$\text{Straf for biler bag rød} = \{0.5, -2, -6, -12, -20, -77\}$$
$$\text{Belønning for bil bag gul og grønt} = \{-0.5, 2, 6, 12, 20, 77\}$$

Det fremgår af ovenstående tabeller, at biler straffes for at holde i kø, men de belønnes tilsvarende for at kunne blive afviklet. Idet RL agenten hele tiden vægter hvad den handling er som optimerer dens pointsum, vil det være optimalt at tømme køer, og på den måde håndhæve en hvis grad af fairness. Derudover får den 0.5 point for at have rød for en retning uden biler og mister 0.5 point ved at have grønt for en retning uden biler i. På denne måde lægges der op til at pointsummene for bilerne i de krydsende retninger udligner hinanden, og på den måde vælger den handling der giver den flest point samtidig med at være fair.

Disse belønninger eller straffe kan beskrives med funktionen R, som afhænger af handlingen den udfører (h), tilstanden den er i nu (t), og tilstanden som handling h får den til at være (t'). Belønningsfunktionen bliver altså = $R(h, t, t')$. RL agenten ved hvor mange point den får for hver handling h i hver tilstand t, og ydermere hvad sandsynligheden for at komme til tilstand t' fra tilstand t. Med dette bruger den en metode kaldet value-iteration, til at forudsige hvor mange point den forventes at få ved at udføre handling h. På den måde gennemgår agenten i hvert sekund de handlinger den har mulighed for at udføre, og undersøger hvilken af dem optimerer antallet af point den kan få. For at gøre dette, skal nogle generelle regler og formler for at beregne udfaldssandsynligheder defineres for agenten. Dette undersøges i den følgende sektion

4.3 Modellering af RL agent

4.3.1 Udfaldssandsynligheder

For at kunne bruge value-iteration, skal RL agenten kende sandsynligheden for at alle ændringer i tilstande, sker fra alle givne tilstande, ved de to handlinger. Den skal kende sandsynligheden for eksempelvis, at gå fra et interval af biler $[4; 8]$ ned til intervallet $[1; 3]$, ved at biler afvikles. Derudover skal den også kende sandsynligheden for at gå op i interval, hvis der tilkommer biler til krydset, samt at blive i intervallet.
Der skal tages højde for både de biler der kan afvikles i løbet af et enkelt sekund, samt de biler, som der kan tilkomme krydset i løbet af et enkelt sekund og dermed udlede hinandens effekt.

Intervaller
Intervallet af biler $[A; B]$ defineres som:

$$[A; B] = \{A, A + 1, ..., B - 1, B\} \; hvor \; A \leq B$$

Antallet af elementer i intervallet betegnes som:

$$|[A; B]| = B - A + 1$$

Med dette betegner det følgende antallet af elementer mellem A og B, antallet

af elementer fra A til B-1 og antallet af elementer fra A-1 til B:

$$|]A;B[| = B - A - 1 \mid hvor A < B$$
$$|[A;B[| = B - A \mid hvor A < B$$
$$|]A;B]| = B - A \mid hvor A < B$$

Belastning
Radarteknologien i krydset er i stand til at tælle det præcise antal biler i en vejbane, dog holder agenten kun styr på hvilket interval dette ligger i. Det præcise antal biler T betegnes også som belastningen. Belastningen T befinder sig i intervallet $[A;B]$ når det gælder at:

$$T \in [A;B] \text{ hvor } A \leq T \leq B$$

Vi ønsker nu at undersøge de mulige belastninger T, der tillader at vi i løbet af det næste sekund, kan springe fra interval $[C;D]$ til $[A;B]$ hvor $C > B$. Det blev i overblikket antaget, at der i løbet af et sekund maksimalt kan afvikles 3 biler, og dermed også at der maksimalt kan tilkomme 3 biler pr. vejbane, hvilke begge betegnes med λ. Dette betyder at der maksimalt findes 3 forskellige belastninger T i intervallet $[C;D]$ der gør det muligt at springe fra interval $[C;D]$ til $[A;B]$ i løbet af et sekund. For at finde det antal forskellige belastninger der tillader springet, trækkes hullet mellem B og C fra λ:

$$Antal_T = \lambda - |]B;C[|$$

På baggrund af dette, er en mulig belastning givet, hvis det gælder at:

$$T \in [C;C + Antal_T[$$

I følgende eksempel fremgår de 2 belastninger der tillader at springe fra $[5;8]$ til $[1;3]$:

$$[A;B] = [1;3] \text{ og } [C;D] = [5;8]$$
$$Antal_T = 3 - |]3;5[| = 3 - 1 = 2$$

På baggrund af $Antal_T$ tjekkes hvorvidt de enkelte mulige belastninger befinder sig i intervallet $[C;C + Antal_T]$:

$$5 \in [5;5+2[$$
$$6 \in [5;5+2[$$
$$7 \notin [5;5+2[$$
$$8 \notin [5;5+2[$$

Det viser sig at belastningerne 5 og 6 tillader dette spring, hvilket betyder at der ved afvikling af 2 eller 3 biler kan ske et spring til det ønskede interval. Disse kaldes de mulige afviklinger og betegnes med R.

Tilkomne biler og afvikling af biler
Idet en formel definition på intervaller og mulige belastninger nu er givet, vil det være relevant at bruge disse definitioner på at definere tilkomne biler Z og afviklede biler R. I starten af sektionen blev det nævnt, at agenten skal kende sandsynligheder for både at springe et interval op, et interval ned, og blive i det nuværende interval. På baggrund af antagelsen i overblikket bruges her λ som symbol til både, at beskrive det maksimale antal afviklede biler, samt tilkomne biler i løbet af et sekund, hvilket er 3.
Det gælder generelt at de mulige afviklinger og tilkomne biler ligger i intervallerne:

$$0 \leq R \leq \lambda \quad \text{(afviklede biler)}$$
$$0 \leq Z \leq \lambda \quad \text{(tilkomne biler)}$$

I forrige eksempel fremgår det, at der er et hul mellem de to intervaller hvilket også betyder at $Antal_R$ og $Antal_Z$ ikke altid er 3, men kan beregnes med de følgende formler. Ved en afvikling R undersøges springet fra interval $[C; D]$ til $[A; B]$, og ved tilkomne undersøges springet fra interval $[A; B]$ til $[C; D]$ hvor det i begge tilfælde gælder at $B < C$:

$$Antal_R = min(\ \lambda,\ |[A; T[|\ ) - |]B; T]|$$
$$Antal_Z = min(\ \lambda,\ |[T; C[|\ ) - |]T; D]|$$

Formlerne tjekker om afstandene fra A til T og T til C er mindre end λ, fordi hvis de er, skal de bruge λ i stedet, så de ikke springer henholdsvis under og over intervallet de tjekker. Ydermere tjekker formlen også hvor mange elementer der er fra den nuværende belastning og hen til intervallet der ønskes at springe til, og trækker dette fra det foregående, for at finde ud af hvor mange forskellige mulige afviklinger der tillader springet disse spring. På baggrund af disse definitioner gælder det, at en afvikling R og tilkomne biler Z er mulige hvis:

$$|]B; T]| \leq R \leq min(\ \lambda,\ |[A; T[|\ ) \tag{1}$$

$$|]T; C]| \leq Z \leq min(\ \lambda,\ |[T; D[|\ ) \tag{2}$$

Sandsynligheder for spring mellem ikke-overlappende intervaller
Generelt gælder det at sandsynlighederne for spring mellem intervaller afhænger af sandsynligheden for at befinde sig i belastning T, sandsynligheden for at der tilkommer biler, samt sandsynligheden for at der afvikles biler.

Når der nu er fastlagt definitioner for de grundlæggende principper, beskrives de konkrete formler som agenten kan bruge til at beregne sandsynligheden for enten at springe mellem intervaller eller blive i et interval ved de to handlinger. Overordnet gælder det at disse sandsynligheder afhænger af 3 parametre:

- Sandsynligheden for at der er en belastning på T hvis man befinder sig i intervallet $[A; B]$
- Sandsynligheden for at der tilkommer specifikke antal biler
- Sandsynligheden for at der afvikles specifikke antal biler

For at bestemme sandsynligheden for at der er en belastning på T, givet intervallet $[A; B]$ hvor T $\in [A; B]$, benyttes følgende formel:

$$P(T \mid [A; B]) = \frac{1}{|[A; B]|} \text{ hvor T} \in [A; B]$$

Det gælder at der antages at der er lige stor sandsynlighed for at være i en vilkårlig belastning i intervallet $[A; B]$, og derfor er denne sandsynlighed en gyldig antagelse.

For nu at bestemme sandsynligheden for at der tilkommer eller afvikles et specifik antal biler, introduceres princippet bag en Poissonfordeling. Poissonfordelingen er en funktion der finder frem til sandsynligheden for at et specifik antal hændelser sker ved hjælp af det gennemsnitlige antal hændelser pr. tidsinterval. Ved brug af Poissonfordelingen antages det at bilernes ankomst eller afvikling ikke har noget at gøre med tidligere og senere bilers ankomst eller afvikling [28]. Poissonfunktionen ser ud som følgende:

$$P(k) = e^{-\omega} \cdot \frac{\omega^k}{k!}$$

ω er det gennemsnitlige antal biler pr. tidsinterval.

k er antallet af biler som der skal findes en sandsynlighed for.

e er Eulers tal.

Sandsynligheden for at der ankommer k biler i løbet af et sekund antages at følge en Poissonfordeling, hvor ω beskriver det gennemsnitlige antal biler der ankommer pr. sekund, hvilket er baseret på trafiktællingerne fra Aalborg Kommune. Sandsynligheden for at afvikle R biler antages også at følge en Poissonfordeling, hvor ω her betegnes som det gennemsnitlige antal biler der afvikles pr. sekund, og antages at være $1\frac{bil}{sek}$, hvis der er en eller flere biler i banen, og banen har grøn signaltilstand.

Når disse sandsynligheder beregnes, tager Poissonfordelingen ikke højde for antagelsen omkring at der maksimalt kan tilkomme og afvikles 3 biler i løbet af et sekund. Når sandsynlighederne beregnes skæres alle sandsynligheder væk hvor $k > 3$. Dette betyder at disse sandsynligheder ikke summerer til 1. Det er derfor nødvendigt at normalisere disse sandsynligheder. Dette gøres ved sandsynlighederne for afviklingen af en bil for at demonstrere det, men der ses bort fra denne detalje i forhold til tilkomne biler, på grund af tidspres, hvilket betyder at sandsynlighederne i sidste ende ikke summerer til 1.

For at normalisere sandsynlighederne benyttes følgende formel:

$$Pr(a) = \frac{P(a)}{\sum_{k=0}^{\lambda} P(k)} \; hvor \; 0 \leq a \leq \lambda$$

$P(x)$ betegner sandsynligheden for x udfald ved en Poissonfordeling
k er antallet af biler som der skal findes en sandsynlighed for
a er antallet af biler der ønskes afviklet
T er belastningen af biler

Heraf fremgår det at sandsynlighederne for at hhv. 0, 1, 2 og 3 biler afvikles i løbet af et sekund, normaliseres til 1 ved at summere sandsynlighederne fra 0 til 3 udfald, og på den måde finde ud af hvor stor en del af sandsynlighederne ligger over 3. Dette fordeles altså over sandsynlighederne for de 4 forskellige afviklinger.

Med disse tre byggesten kan sandsynlighedsformlerne der beskriver det at forblive i intervaller, samt springene op og ned defineres. Disse formler afhænger af hvilken signaltilstand vejbanen befinder sig i, hvilket betyder at der her kun beskrives de formler der kan beregne sandsynligheder for en vejbane der er i en grøn signaltilstand, dog er formlerne for de røde vejbaner baseret på de samme principper og implementeret i RL-agentens kode.

Det er nødvendigt at kende de summerede sandsynligheder for at springe fra $[C; D]$ til $[A; B]$ ved de mulige belastninger:

$$Pr([A; B] \mid [C; D]) = \sum_{T=C}^{D} Pr(T)$$

Pr(T) betegner sandsynligheden for at springe et interval ned ved den nuværende belastning T. Denne sandsynlighed Pr(T) er beskrevet ved summen af sandsynlighederne for at alle mulige afviklinger sker. Dette er beskrevet i nedenstående formel:

$$Pr(T) = \sum_{R=]B;T]}^{min(\ \lambda,\ |[A;T]|\ )} \Big(Pr_{tilstand}(T \mid [C; D]) \cdot Pr_{afvikling}(R) \cdot Pr_{modvirkning}(0 \leq k \leq R - \ |]B; T]|) \Big)$$

Den nedre og øvre grænse af summen gennemgår alle mulige afviklinger og er givet af Formel 1. Heraf summeres sandsynlighederne for at være i den specifikke belastning T, at der afvikles R biler - så mange det er nødvendigt at springe ned i interval - og sandsynligheden for at denne afvikling modvirkes. At modvirke beskriver en ændring der kan forårsage at afviklingen af R biler alligevel ikke er nok til at springe ned til det ønskede interval. Modvirkningen udligner afviklingen af biler, og er derfor sandsynligheden for at der tilkommer biler til krydset, i dette tilfælde. Sammensat kommer formlen for sandsynligheden for spring ned i interval dermed til at se ud som følgende:

$$Pr([A;B] \mid [C;D]) = \sum_{T=C}^{D} \Bigg(\sum_{R=|]B;T]}^{min(\ \lambda,\ |[A;T]|\ )} \Big(Pr_{tilstand}(T \mid [C;D]) \cdot Pr_{afvikling}(R) \cdot \ Pr_{modvirkning}(0 \leq k \leq R - \ |]B;T]|) \Big) \Bigg) \tag{3}$$

Formlen for at springe op i interval fra $[A;B]$ til $[C;D]$ ved en signaltilstand der er grøn beskrives med en lignende formel som ovenstående, dog tages der nu udgangspunkt i sandsynligheden for at der tilkommer biler, og modvirkningen er nu de afviklede biler. Derfor gælder det også at den nedre og øvre grænse er bestemt fra Formel 2 i forhold til ankommende biler. Formlen beskrives herunder som Formel 4:

$$Pr([C;D] \mid [A;B]) = \sum_{T=A}^{B} \Bigg(\sum_{Z=|]T;D]}^{min(\lambda,\ |[T,C]|\ )} \Big(\ Pr_{tilstand}(T \mid [C;D]) \cdot Pr_{afvikling}(R) \cdot \ Pr_{modvirkning}\big(0 \leq R \leq min(\lambda, (T+Z) - C)\big) \Big) \Bigg) \tag{4}$$

Den sidste af disse tre formler beskriver sandsynligheden for at forblive i intervallet. Den nedre og øvre grænse går altså fra 0 til antallet af mulige tilkomne biler. Modvirkningerne afhænger af både tilkomne og afviklede biler eftersom begge ting modvirker målet om at blive i intervallet. Når sandsynligheden for en afvikling beskrives er det vigtigt at understrege at der på et enkelt sekund i simulationen først tilkommer biler, hvorefter der afvikles biler. På den måde er det muligt at der tilkommer 3 biler, hvilket betyder at der springes op til et højre interval, hvorefter der i samme sekund afvikles 3 biler, hvilket betyder at belastningen T ender i samme interval som den startede. Denne sandsynlighed inkluderes ved at tilføje den nedre grænse $max(0, T+Z-B$ der undersøger hvor langt over intervallet den kan komme ud hvor den har mulighed for at komme tilbage igen. Den øvre grænse betegner det maksimalt mulige antal afviklinger uden at komme ud af intervallet.

$$Pr([A;B] \mid [A;B]) = \sum_{T=A}^{B} \Bigg(\sum_{Z=0}^{min(\lambda, B+\lambda-T)} \Big(\ Pr_{tilstand}(T \mid [A;B]) \cdot Pr_{tilkomne}(R) \cdot \ Pr_{afvikling}\Big(max(0, T+Z-B) \leq R \leq min\big(\lambda, (T+Z-A)\big)\Big) \Big) \Bigg) \tag{5}$$

Herover er alle formler der skal bruges til at beregne sandsynligheder for spring mellem intervaller defineret, det er dog også relevant for agenten at kende sandsynligheden for blandt andet at bevæge sig til næste signaltilstand og tid i tilstand. Disse sandsynligheder en mere ligetil og derfor overordnet beskrevet i bilag 2 og 3.

4.3.2 Implementering og optimeringstricks

Formeloptimering
Vores formler for sandsynligheder er blevet simplificeret i rapporten for at fremhæve forståelsen, dette giver dog mulighed for optimering af formlerne ved implementeringen af dem i programmet.
Et eksempel er Formel 3 som beskriver sandsynligheden for at bevæge sig fra et interval ned til et andet ikke overlappende interval:
Her gælder der for T at $C <= T <= D$. Vores maksimale afvikling er defineret som λ, og derfor betyder det at for en belastning T, at hvis afstanden mellem det ønskede interval og belastningen T er større end λ, er det ikke muligt at bevæge sig til det ønskede interval, og sandsynligheden er derfor 0% og kan derfor springes over i beregninger af optimeringsårsager. Dette kan beskrives således:

$$C \leq T \leq C + min(\ \lambda - |]B;C[|,\ |[C;D]|\ )$$

Fordi C stadig er startgrænsen og λ er vores maksimale afvikling, og derfor også det maksimale antal belastninger vi behøver at tjekke, men da der kan være en afstand mellem B og C som skal springes over for at nå vores ønskede interval, så skal denne afstand trækkes fra λ for at få det reelle maksimale antal afviklinger. I nogle få tilfælde ved små intervaller skal man undgå ikke at gå over det nuværende interval, og dette undgår man ved at sige at der maksimalt må søges op til $C + |[C;D]|$. Dette ændrer formlen til:

$$Pr([A;B] \mid [C;D]) = \sum_{T=C}^{min(\ \lambda-|]B;C[|,\ |[C;D]|\ )} \Bigg(\sum_{R=|]B;T]}^{min(\ \lambda,\ |[A;T]|\ )} \Big(\tag{6}$$

$$Pr_{tilstand}(T \mid [C;D]) \cdot Pr_{afvikling}(R) \cdot\ Pr_{modvirkning}(0 \leq k \leq R - \ |]B;T]|) \Big) \Bigg)$$

Kodeoptimeringer
Foruden at kunne optimere vores formler, har vi også kunne yderligere optimere vores programstruktur således at programmet har sparet meget lange beregningstider.

```
1 if (fooBar(foo, bar)){
2     doSomethingTimeExpensive();
3 }
```

vs.

```
1 if (fooBar(foo, bar) && doSomethingIsPossible(foo, bar)){
2     doSomethingTimeExpensive();
3 }
```

Hvis man ser på eksemplerne ovenfor fremgår det af forskellen at der er tilføjet et ekstra tjek i if-sætningen, som har til formål at se om funktionen doSomethingTimeExpensive() er tilgængelig, eller om dens output allerede er kendt ud fra funktionen doSomethingIsPossible(), hvilket kan spare programmet for mange gennemkørsler af denne tidskrævende funktion.

Den ovenstående optimering af programmet er implementeret når sandsynligheden for at gå fra et stadie til et andet skal beregnes, hvilket er en funktion der ofte bliver kørt. Ved at tilføje en funktion som på baggrund af nogle simple beregninger kan finde ud af om sandsynligheden bliver 0%, fik programmet en hastighedsoptimering af faktor 10. En beregning som før tog cirka 500 sekunder at beregne, tog bagefter cirka 50 sekunder.

```
1 for (i = 0; i < Calc_Max_Limit(foo, bar); i++){
2     doSomething(1);
3 }
```

vs.

```
1 limit = Calc_Max_Limit(foo, bar);
2
3   for (i = 0; i < limit; i++){
4     doSomething(1);
5   }
```

Ved at kigge på eksemplerne ovenfor er forskellen mellem dem at der i den nederste version er beregnet en slutgrænse og gemt den som en værdi "limit", hvorimod den øverste hvor beregningen er placeret inde i for-loop'et. Rent praktisk betyder det i det øverste for-loop at hver eneste gang loop'et køres igennem og skal tjekke om det skal stoppe, skal det køre og beregne funktionen Calc_Max_Limit(), hvilket resulterer i at beregningerne i Calc_Max_Limit() bliver kørt lige så mange gange som for-loop'et bliver kørt.
Den nederste version laver kun beregningen én gang, og der kan derfor spares en del beregninger og tid. At implementere dette generelt igennem programmet forkortede køretiden yderligere 1 sekund, hvilket i sammenhæng med forrige eksempel bringer den ned på 49 sekunder.

```
1 if (5 <= fooBar && fooBar => 1){ /* 5 muligheder */
2     doSomething(1);
3 } else if (20 <= fooBar && fooBar >= 6){ /* 15 muligheder */
4     doSomething(2);
5 } else if (fooBar > 20){ /* Uendelige muligheder */
6     doSomething(3);
7 }
```

vs.

```
1 if (fooBar > 20){ /* Uendelige muligheder */
2     doSomething(3);
3 } else if (20 <= fooBar && fooBar >= 6){ /* 15 muligheder */
4     doSomething(2);
5 } else if (5 <= fooBar && fooBar >= 1){ /* 5 muligheder */
6     doSomething(1);
7 }
```

Hvis man ser på koderne foroven er den eneste forskel rækkefølgen af if-else kæderne. I den nederste - og optimerede version - er de placeret i den rækkefølge som oftest forekommer, hvor i den øverste er rækkenfølgen ikke således, hvilket resulterer i at man i gennemsnit skal lave langt flere sammenligninger. Ved at ændre rækkefølgen kan der altså spares en del sammenligninger.
I små programmer har denne optimering ikke den store forskel, men når if-else kæden bliver kørt flere hundredetusindevis af gange i løbet af køretiden af programmet, kan selv de mindste optimeringer forkorte køretiden med sekunder.

```
1 unsigned long factorial(unsigned long f){
2   if ( f == 0 )
3       return 1;
4   return(f * factorial(f - 1));
5   }
```

vs.

```
1 int fac[10] = {1,1,2,6,24,120,720,5040,40320,32880};
2
3 int factorial(int f){
4     return fac[f];
5 }
```

Hvis man ser på koderne foroven er der først en rekursiv funktion, og forneden en simpel funktion som returnerer et forudberegnet resultat fra et array. En rekursiv funktion har generelt mange funktionskald, og dernæst kommer udregningerne. Dette kan ofte optimeres ved at forudregne resultaterne i det søgerum som man anvender funktionen, da det ofte er hurtigere at finde en gemt værdi end at beregne den igen hver gang den skal bruges.

```
1 if (isFoo(foo) && bar == 1){
2     doSomething(1);
3 } else if (isFoo(foo) && bar > 1){
4     doSomething(2);
5 } else if (isFoo(foo) && bar < 1){
6     doSomething(3);
7 }
```

vs.

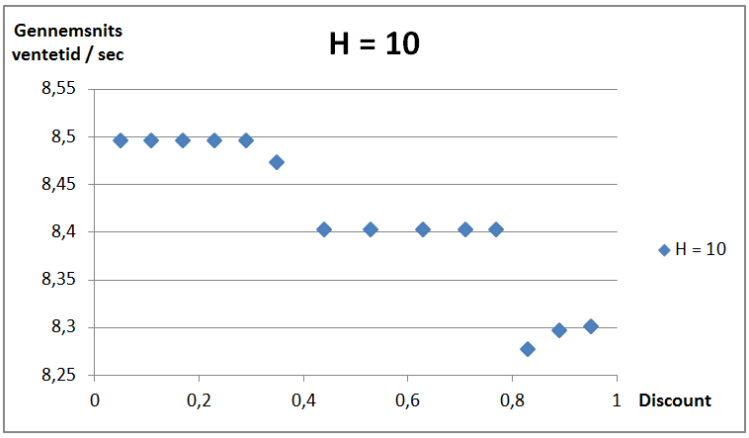
```
1 is_foo = isFoo(foo);
2
3 if (is_foo && bar == 1){
4     doSomething(1);
5 } else if (is_foo && bar > 1){
6     doSomething(2);
7 } else if (is_foo && bar < 1){
8     doSomething(3);
9 }
```

Ser man på eksemplerne foroven er forskellen at der i den øverste er det samme funktionskald i alle if-else betingelser, hvor der forneden er en variabel som holder resultatet af denne funktion, og dette resulterer i at funktionen isFoo() bliver kørt langt færre gange end ellers, hvilket i store programmer kan betyde flere sekunder til forskel.

4.3.3 Læring af strategi

For at finde en god strategi for vores RL agent testedes først forskellige point- og strafsammensætninger igennem en stikprøvetilgang hvor det blev tydeliggjort hvilket point- / strafsystem var bedst.
Efterfølgende påbegyndte træningen af mange forskellige agenter med forskellige discount-værdier samt tidshorisonter. På denne måde blev der dannet et stort set af agenter, som efterfølgende kunne testes og derigennem finde den bedste / en god agent.

Vi trænede i alt ca. 1.350 agenter og det tog ca. 35 sekunder for en normal computer at simulere et helt døgn, hvilket betyder at teste alle mulige agenter ville tage ca. 13 timer, men at teste alle agenter er en unødvendig ineffektiv metode til at finde en god agent, så derfor valgte vi først med stikprøver for hver discount-værdi at finde den discount-værdi der virkede til at være bedst, som fremvist på Figur 8, hvilke viste sig at være 0,83. Herefter testedes alle tilgængelige tidshorisonter med et interval af 5, og opdagede at omkring tidshorisont 11 var den bedste løsning, og derfor testede vi så efterfølgende alle nærliggende tidshorisonter, og fandt frem til at tidshorisonten 10 kombineret med discount-værdien 0.83 sandsynligvis er en af de bedste agenter tilgængelige.



Figur 8: Eksempel på en stikprøve test af discount værdier ved en fast tidshorisont H = 10.

5 Evaluering

Til evalueringen af vores løsning har vi lavet en simulation. Med denne simulerer vi et lyskryds, som en løsning skal forsøge at afvikle. Vi tester en tidsstyret og trafikstyret løsning samt vores RL Agent og sammenligner resultaterne, for at kunne evaluere RL agentens effektivitet.

5.1 Design af simulering

For at opnå en virkelighedstro simulering bruges der, som tidligere nævnt trafikdata fra Aalborg kommune. Vi har derfor også valgt at basere vores simulering på et af de kryds, som der er flest trafiktællinger for. Det valgte kryds kalder vi "Karolinelund". Det er placeret i nærheden af Aalborg centrum og forbinder de fire gader; Jyllandsgade, Kjellerupsgade, Fyensgade og Sønderbro. Karolinelund har fodgængerovergange og hver gade har op til tre dedikerede svingbaner. Venstresvingsbanerne har tilhørende venstresvingssignaler. Den simplificerede model af Karolinelund som vi har lavet har kun ligeud baner, og der er ingen dedikerede svingbaner. Dette kan ses på figur 7 som viser en model af lyskrydset. I vores simulering møder billisterne ingen forhindringer i disse baner som de skal stoppe for, det vil sige at når en bil kører ind i krydset kan den anses som afviklet.

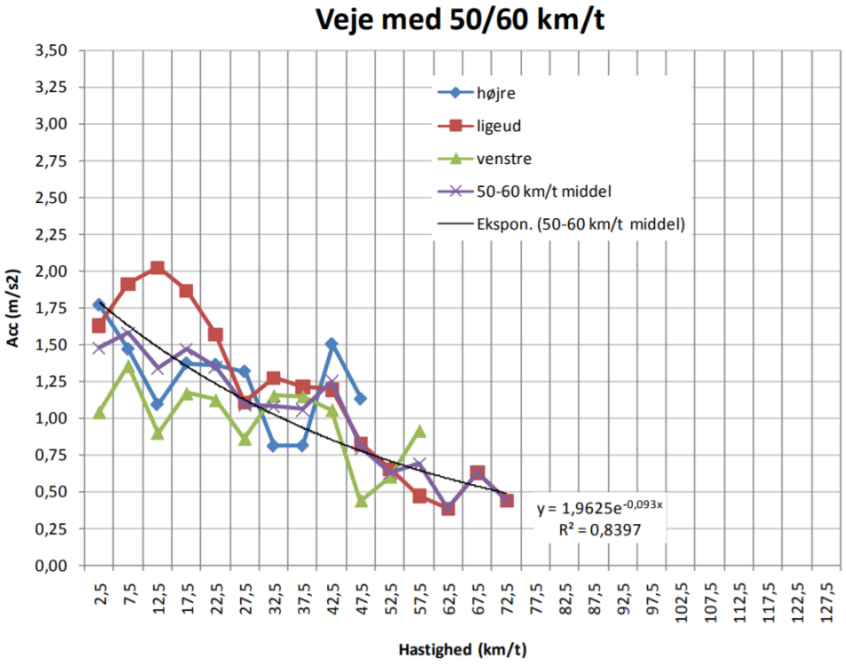
Med de antagelser der blev lavet om krydset i overblikket, vil vi skabe en simulering der kan genskabe et lyskryds ud fra de informationer vi har om Karolinelund. Simuleringen er modelleret til at kunne simulere et kryds ud fra de informationer den modtager, i form af trafiktællinger. Dette betyder at vores simulering kan bruges på alle enkelte kryds.

5.1.1 Acceleration og deceleration

Accelerationen i simuleringen er baseret på data fra en undersøgelse lavet af trafictec. Herfra bruger vi i simuleringen en gennemsnitlig eksponentiel sammenhæng til at beskrive accelerationen som funktion af hastigheden. Bilerne får derved en tilnærmelsesvis realistisk acceleration, hvilket er en stor faktor for ventetiden i krydset.

Grundet tidsbegrænsninger har vi valgt at se bort fra deceleration samt varierende accelerationer fra bil til bil.

Vores kilde til acceleration havde en gennemsnitlig accelerations funktion $F(x) = 1,29625e^{-0,093}$. Dette gennemsnit er for veje med 50/60 km/t zoner, men det er den bedste antagelse vi kunne finde frem til.

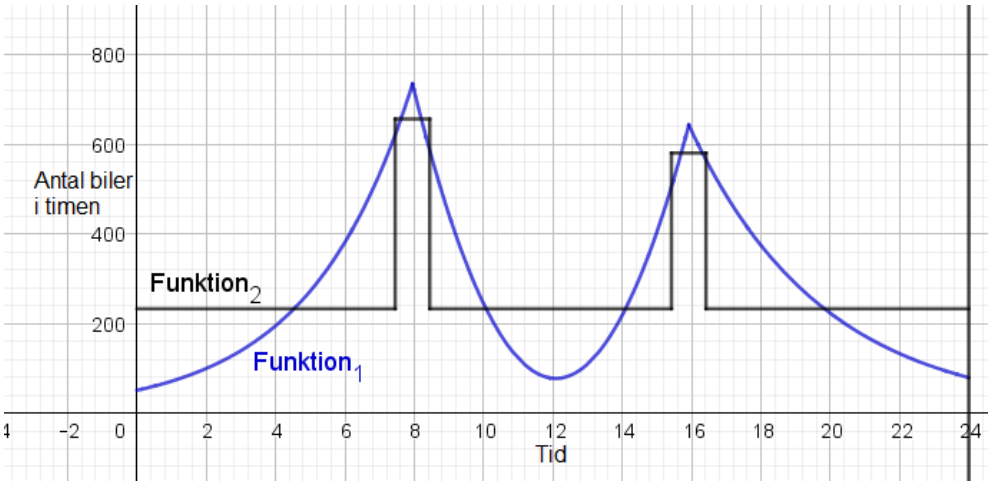


Figur 9: Graf der repræsenterer acceleration i 50-60 km/t zoner. [31]

5.1.2 Trafiktællinger

Det gennemsnitlige antal biler gennem dagen er en antagelse vi har lavet ud fra dataen fra Aalborg kommune. Vi har antaget med en funktion hvordan en mere "normal"fordeling af biler ville blive igennem dagen.

Som set på figur 9 ville bilerne i følge dataen alle komme præcis når spidstimerne starter (funktion 2). Dette er selvfølgelig ikke virkelighedsnært og vi har derfor lavet antagelser med en ny funktion, der varierer i antal biler (funktion 1). Der er lavet en lignende funktion for de tre øvrige retninger i lyskrydset. Funktionerne består af tre dele: to eksponentielle funktioner, hvor den ene er voksende og den anden er aftagende, og en parabel. De tre underfunktioner er lavet ved løsning af ligningssystemer med henholdsvis to og tre variable. Funktionerne er lavet ud fra et par antagelser angående hvornår vi ville lade antallet af biler stige over/falde under det gennemsnitlige antal biler i timen, ifølge datasættet fra Aalborg Kommune. Vi har primært valgt disse værdier efter hvad der lyder mest realistisk, og hvad der giver den pæneste graf. Det skal bemærkes, at disse funktioner ikke giver et fuldkommen realistisk billede af, hvordan antal af biler udvikler sig i løbet af døgnet, men vi antager at de giver et bedre billede end den rå data.



Figur 10: Graf der repræsenterer antal biler i timen.

5.2 Implementering

5.2.1 Ankomstrate

I simuleringen er ankomsten af biler bestemt ud fra vores grafer som dikterer antallet af biler pr. time. I et realistisk scenarie ændres bilernes ankomstsrate ikke nødvendigvis glidende, men i stedet med en grad af tilfældighed. For at simulere dette tilføjes et tilfældighedselement til bilernes ankomst.
Denne tilfældighed opnås ved at bruge en poisson-fordeling som bestemmer chancen for at en hændelse sker k gange inden for et tidsinterval.

5.2.2 Poissonfordeling

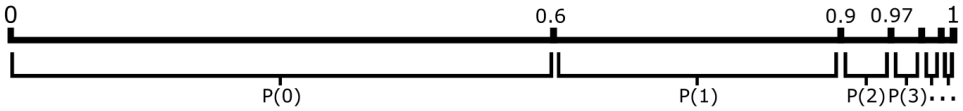
Med Poisson fordelingen som beskrevet i afsnit 4.2.4 kan vi lave antagelser omkring sandsynligheden for biler der ankommer til krydset. Det lyder ikke helt præcist at der er større chance for 0 biler end for 1 bil, men vi bruger Poisson som en antagelse i vores løsning, og man kunne senere hen selv indsætte det rigtige data for at få en mere præcis løsning. Denne sandsynlighed udregnes ved først at bruge vores trafikfunktioner til at bestemme det gennemsnitlige antal biler, som ankommer inden for de næste 15 sekunder. Ud fra trafikfunktionerne ved vi eksempelvis, at ved klokken 11 er det gennemsnitlige antal biler pr. 15. sekund ∼0,5.

Sandsynligheden for at der ankommer 0, 1 og 2 biler bliver så:

$$P(0) = 0,60 = 60\%$$

$$P(1) = 0,30 = 30\%$$
$$P(2) = 0,07 = 7\%$$

Vi bruger Poisson fordelingen i vores kode til at spawne biler. Efter vi har udregnet sandsynligheden for hver vej i vores kryds, sætter vi disse sandsynligheder sammen. Som set på figur 10.
Dernæst genereres et tilfældigt tal i intervallet [0,1]. Hvis dette er i intervallet som P(k) dækker over, tilføjes der k biler. Med udgangspunkt i figuren tilføjes der eksempelvis 1 bil hvis den tilfældige værdi er imellem 0.6 og 0.9.



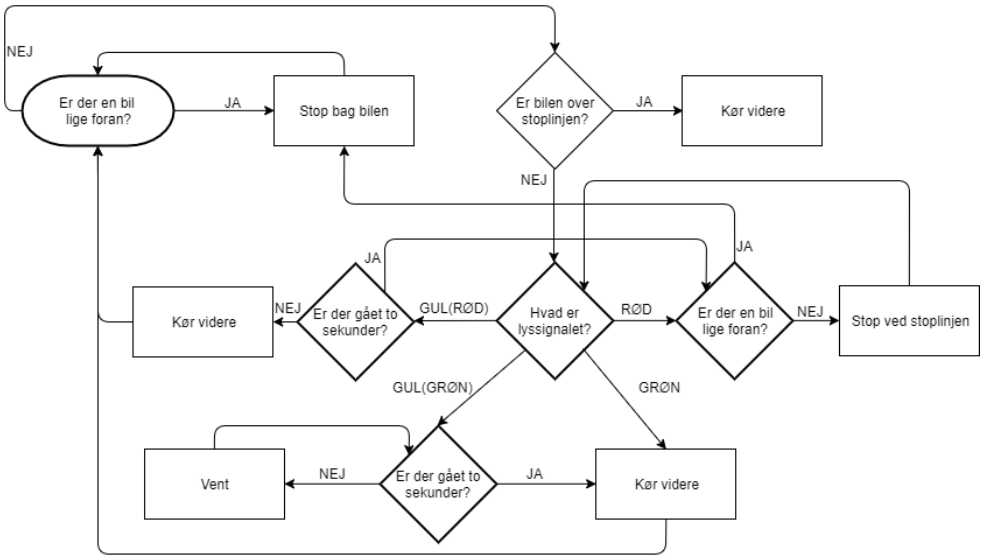
Figur 11: Fordelingen af sandsynlighederne mellem 0-1.

5.2.3 Bag simuleringen

I vores kode afhænger vi meget af structs. Som set på figur 12 har vi et struct i et struct i et struct. Det vil sige at vi har et struct til vores fire gader, og inden i hver af disse fire gader er der én bane (højre og ligeud) og i hver af banerne kan der i alt være 126 biler. Med disse structs kan vi holde styr på alle fire vejbaner på samme tid.

Figur 12: Flowchart over structs i structs

I vores simulering har vi biler der reagerer på dens "omgivelser". Bilerne har fået regler for hvad de kan gøre i simuleringen, feks hvornår de må køre og hvornår de skal stoppe. Figur 13 viser hvordan en bil i simuleringen forholder sig. Den tjekker først om der er en bil foran den og derefter reagerer den til stoplinjen og efter dette reagerer den til lyssignalet. Dette bliver bilen ved med indtil den er afviklet.

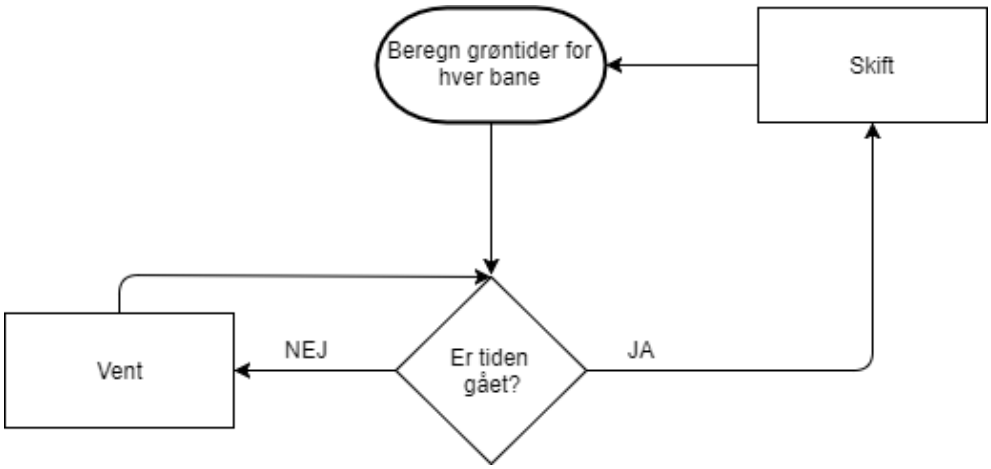


Figur 13: Flowchart over hvordan bilen vælger at opføre sig.

Som den simpleste løsning til vores simulering har vi det tidsstyrede lyskryds. I denne løsning skifter krydset signal hver gang tiden er gået, og den tager ingen hensyn til om bilerne er der eller ej. Figur 14 viser hvordan vores tidsstyrede simulering virker. Det den gør er at konstant at vurdere om tiden er gået eller ej. Den skifter signal hvis tiden er gået.

Figur 14: Hvordan den tidsstyrede løsning virker.

Vores trafikstyrede løsning er en lidt mere kompleks løsning i forhold til den tidsstyrede, den tjekker også bilerne. Som set på figur 14, så udregner den trafikstyrede først en grøntid for bilerne i det givne tidspunkt. Herefter kører den signaltilstandene og beregner forfra.

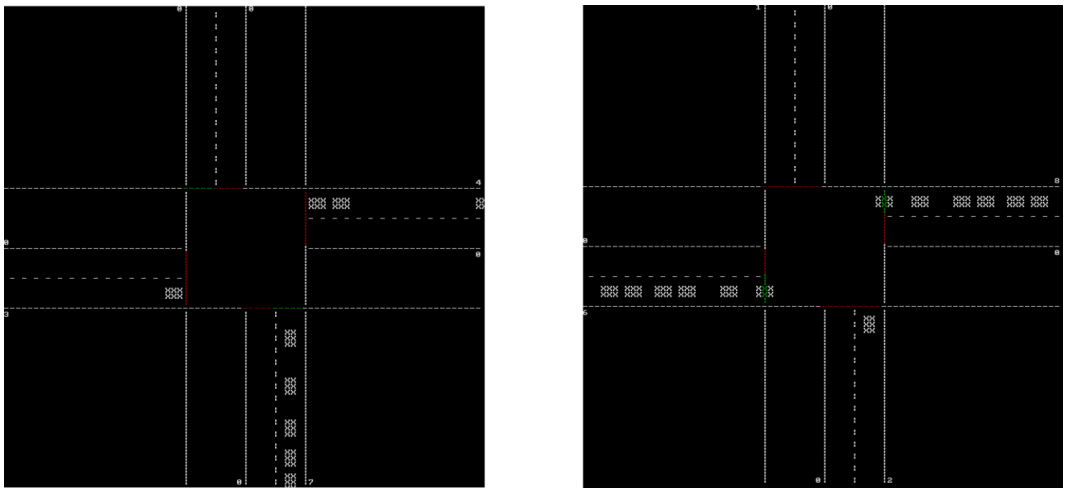


Figur 15: Flowchart over den trafikstyrede simulation.

Vores simulation indeholder en grafisk repræsentration af krydset som kan opdateres i realtid. Den grafiske flade viser vejbaner, signalfarve og biler som opdateres flere gange hvert sekundet. Herved kan man se at bilerne først kører når der er grønt, og at de stopper ved stoplinjen når det er rødt. Figur 15 viser to billeder af vores output fra simulationen. Man kan på disse outputs se stregerne på banerne viser hvilket lyssignal der er for denne bane. Tallet ud fra hver bane viser hvor mange biler der der i pågældende bane på det givne tidspunkt. Denne grafik er med til at gøre det lettere at opdage og identificere fejl i både simuleringen og løsningsstrategier.

Simuleringen bruger et felt af 85 gange 85 teksttegn i konsollen som alternativ til pixelgrafik. Grafikken består primært af tegnene 'x', '—', '-' og mellemrum. Objekter som biler og vejer kan således repræsenteres grafisk ved en kombination af disse tegn.
Alle tegnene opbevares i et todimensionelt array hvor hvert tegn har et koordinatsæt (x,y). Tegnene kan derved tilgås og manipuleres før de printes på skærmen. Det sidste tegn i hver linje er desuden nulafslutning '\0' således at der kan printes en hel linje ad gangen. Skærmen opdateres flere gange i sekundet ved at rydde konsollen og printe nye tegn.

Simuleringen med grafik kører som standard i real-tid, men kan indstilles til andre tidsforhold. Grafikken kan også slås fra så der kan simuleres flere dage på få minutter.



Figur 16: Output i konsollen af simulationen.

5.2.4 Tidsstyret kryds

Vi har først lavet et udelukkende tidsstyret lyskryds, der har adgang til begrænsede mængder information. De tidsstyrede lyskryds får ingen information fra omverdenen, når først det er sat i gang, udover tidspunktet på dagen. Lyset er programmeret til at skifte signal efter et forud bestemt antal sekunder. Til at starte med har vi bestemt, at lyskrydset skal skifte signal efter 30 sekunder. I datasættet med trafiktællinger fra Aalborg Kommune kan vi se, at der kommer væsentligt flere biler i den øst- og vestgående retning. På baggrund af dette har vi valgt at øge grøntiden for Jyllandsgade og Fyensgade med 50%. I morgenspidstimen er forholdet mellem bilerne i det to retninger ikke nær så stort, så i denne periode vil de horisontale vejbaner have 25% mere grøntid. Denne løsning er ikke særlig god, da den ikke tager hensyn til antallet af biler og mulige uregelmæssigheder. Der er meget unødvendig ventetid ved denne form for styring af et lyskryds. Vi vil derfor lave en forsimplet udgave af et trafikstyret lyskryds

5.2.5 Forsimplet udgave af trafikstyret kryds

Vi har ud over det tidsstyret kryds lavet en forsimplet udgave af et trafikstyret kryds. Dette kryds har adgang til information, såsom om der er biler i en vejbane eller ej og kan handle derudfra. Lyskrydset kan derfor skifte signal, alt efter hvilken vejbane der er biler i.

Ved beregningen af grøntiden for lyskrydset, kigger det på, hvor mange biler der er i hver bane efter en cyklus af signalskift(det vil sige en grønperiode i nord/-syd retningen og en grønperiode i øst/vest retningen), og bedømmer derudfra hvilken retning skal have mest grøntid. Vi bruger funktionen for $x^{0.5}$, hvor x er forholdet mellem antallet af biler i hver retning, til at bestemme forholdet mellem grøntiderne, da det ikke ville være fair, at en retning vil have 6 gange så lang grøntid, når der er 6 gange så mange biler.

I forhold til vores RL agent løsning, tager dette trafikstyret kryds ikke højde for fairness ud over en lille ændring i grønperioder afhængigt af antal biler og en fast maksimal grøntid. Man kan derfor med god tilnærmelse sige, at dette lyskryds derfor har en lille fordel over RL agenten, i form af, at den ikke behøver tænke på bilers ventetid på samme niveau, og derfor ikke behøver at bruge tid på at skifte signal hvis det ikke kan svare sig i forhold til antallet af afviklede biler.

5.3 Eksperimentering og resultater

Vi startede ud med originalt, at lave 2 baner per gade i simuleringen, en til biler der skulle dreje til venstre og en til biler der skulle køre lige ud eller dreje til højre. Dette fungerede i simuleringen, men da vi udregnede tiden det tog at træne en agent, var det ikke muligt at få venstresving med i vores metode i reinforcement learning, givet den mængde tid vi har til rådighed. Vi besluttede derfor at forsimple det ved at ændre simuleringen så den havde færre tilstande og ikke havde nogen dedikeret venstresvingsbane. Hvis vi havde haft mere tid ville vi have fundet en metode til at få venstesving med i simulationen, eksempelvis ved brug af q-learning.

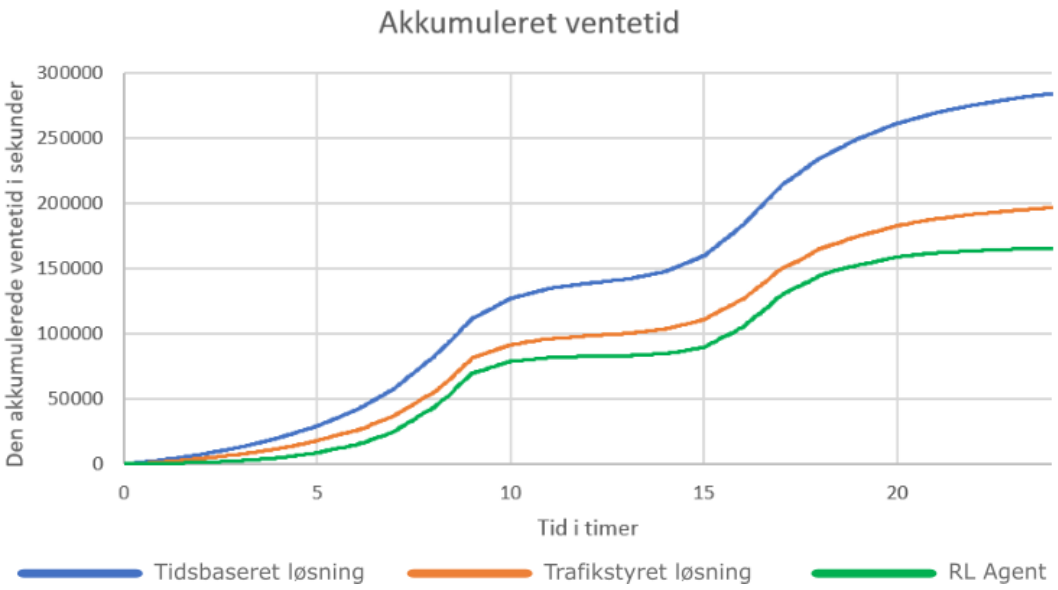
For at vise at vores reinforcement agent er bedre end andre løsninger, laver vi også en tidsstyret og en trafikstyret løsning. Vi laver den tidsstyrede løsning så godt vi kan ud fra antagelser fra det givne data, så resultatet er så fair som det kan være. Det trafikstyrede laves med hensigt på den nuværende trafik, og ikke dømmer på fremtidige estimater, hvor vi også forsøger at bruge dataen til at lave løsningen ud fra antagelser ligesom ved det tidsstyrede.

5.3.1 Reinforcement learning kryds vs. tidstyret og trafikstyret kryds

For at kunne sammenligne resultater mellem et trafikstyret kryds og vores reinforcement learning kryds, bør begge kryds have de samme parametre derfor bør maksimal grøntid være den samme.

Vi ønsker at sammenligne resultaterne fra vores tidsstyrede og trafikstyrede simuleringer med resultaterne fra simuleringen, som er styret af vores RL agent. Resultaterne vi vil sammenligne, er den akkumulerede ventetid og gennemsnittet af afviklede biler over tid. Dette vil vi illustrere i form af grafer og nøgletal.

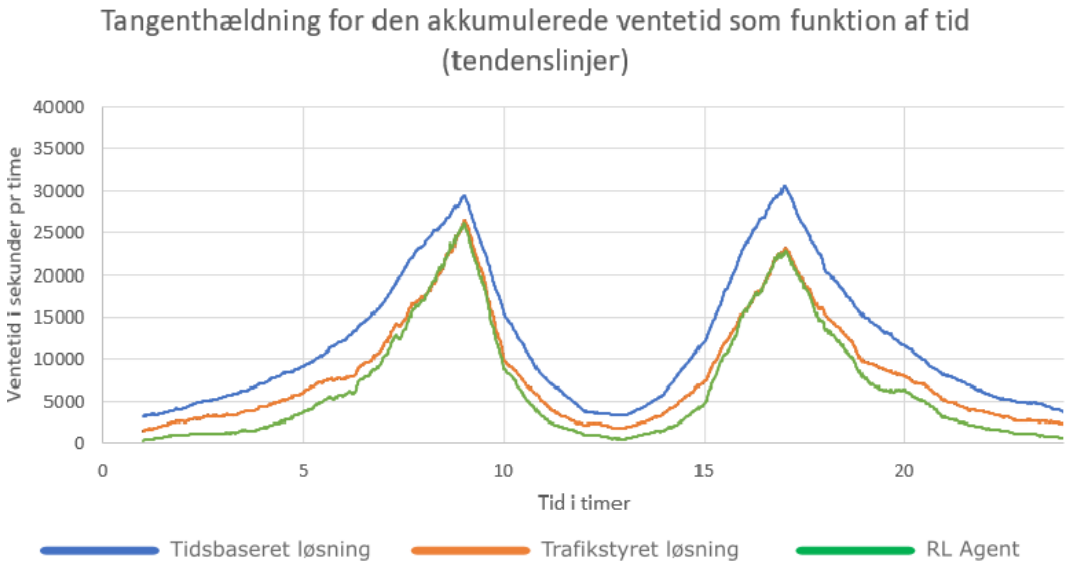
Ventetid defineres i vores statistik, som den ekstra tid en bil bruger på at komme i gennem krydset. Ekstra tid vil sige differencen mellem den tid bilen har brugt i krydset, og den tid som det vil tage at køre igennem krydset med topfart.



Figur 17: Graf for den akkumulerede ventetid.

Ser man på Figur 17, vises den akkumulerede ventetid for de forskellige strategier. Den akkumulerede ventetid er opsamlet ved at checke, om der er stilstående biler i hver vejbane hvert sekund i løbet af de 24 timer, som vi har kørt simuleringen i. Som der afbildedes på graferne, stiger hældningen på den akkumulerede ventetid i spidstimerne, hvilket giver god mening, da der tilkommer flere biler i disse perioder. Det kan tydeligt ses på grafen ud fra den totale ventetid for de forskellige strategier, at RL agenten og den trafikstyret strategi er mere effektiv end den tidsstyrede. Dette giver også god mening, da disse strategier tager højde for ændringer i trafikken i en højere grad, end den tidsstyrede.

Derudover som tidligere beskrevet, tager den trafikstyrede ikke højde for fairness på samme niveau og på trods af dette, har vores RL agent lavere akkumuleret ventetid end den trafikstyrede strategi. På baggrund af resultaterne tyder det på, at reinforcement learning kunne være en effektiv metode til at bearbejde lyskryds.
Grafen for akkumuleret ventetid viser i sig selv ikke noget specielt, andet end at de forskellige strategier medfører en forskellig mængde af sekunder ventetid efter 24 timer, hvor RL agenten har samlet set mindst ventetid, den trafikstyret har anden mindst ventetid og den tidsbaserede har mest ventetid.



Figur 18: Graf for tangentlinjen for hvert datapunkt i den akkumulerede ventetids graf. Dette er tendenslinjerne.

For at kunne bruge den data, som vi har fra simulationerne, kan vi kigge på tangenthældningerne for de forskellige funktioner, hvilket er afbildet som en graf på Figur 18. Her kan vi se hældningen af graferne, hvilket viser hvor gode funktionerne er til at holde ventetiden nede. Som grafen viser, følger de tre funktioner den samme bevægelse, hvilket kan tyde de to spidstimer, som der er i trafikken i løbet af en hverdag. Det interessante ved denne graf er, at man kan se hvornår en funktion er bedre end en anden til at afvikle biler, og at den trafikstyrede og RL agenten konsekvent er bedre end den tidsbaserede løsning.

Her kan vi se, at udenfor spidstimerne er RL agenten en del bedre end både den tids- og trafikstyrede strategi, i forhold til hvor meget ventetid der kommer i løbet af de 24 timer, hvorimod i spidstimerne er RL agenten ligeså god som den trafikstyrede.

En mulig forklaring på, hvorfor den trafikstyrede er ligeså god i spidstimerne som vores RL agent kunne skyldes, at de bil intervaller vi har valgt at bruge, bliver større og større jo flere biler der er. Med en mere begrænset viden, vil agenten derfor ikke kunne handle optimalt hver gang. Derudover har vores agent ikke nogen viden om hvilken tid på dagen den befinder sig i. Dette betyder også, at den ikke tager højde for en større tilkomst af biler ligesom den trafikstyrrede.

Tabel 1: Ventetider og kølængder

	Tidsstyret	Trafikstyret	RL styret
Gennemsnitlig ventetid 1 dag	12.5	8.8	7.4
Max ventetid 1 dag	65.8	42.3	73.6
Max kølængde 1 dag	23	15	17

disse resultater kan man se som både dårligere og bedre da den gennemsnitlige vente tid for RL er lavere end den trafikstyrede men maks ventetid er større.

Grunden til dette kan være svært at afgøre, kunne foreksempel være at var mange biler i den nord- og sydgående retning, men måske kun en bil i den øst- og vestgående retning. I dette tilfælde er det mest logiske og måske også det mest fair valg, at lade den store mængde biler igennem først, hvilket desværre ville give en lang vente tid for den ene bil. De andre parametre der skal tages højde for er for eksempel, at den trafikstyrede har mulighed for at se præcis hvor mange biler der er på banen og kan derved ændre sin lys tilstand efter det.Hvor den RL baserede kan se i intervaller hvilket betyder den ikke kan se det præcise antal biler. Der ved kan den trafikstyrede have en fordel når der skal afvikles biler. Men den RL styrede har til gengæld en fordel ved at den ved hjælp af tidligere data og matematiske formler kan udregne chancen for at der kommer biler og der efter ændre sin tid efter det.

6 Epilog

6.1 Konklusion

I vores indledende problemanalyse blev det undersøgt og analyseret hvilket problemer, der eksisterer i den tætte bytrafik. Der blev her lagt fokus på parametre som kødannelse, ventetid og fairness, med henblik på at optimere disse. Det blev tydeliggjort i problemanalysen, hvordan signalanlæg i lyskryds er opbygget, samt hvilke teknologier der bruges i Aalborg nu og er planer om i fremtiden. Der blev undersøgt forskellige optimeringsalgoritmer, og konkluderede heraf at problemet kunne anskues som et reinforcement learning problem, og udviklede derfor en RL agent til løsning af problemet.

I problemløsningen gives der to bud på metoder, der skal bruges til at repræsentere virkeligheden, i forhold til hvordan trafikken bliver afviklet i lyskrydsene i Aalborg i dag. Vi lavede en tidsstyret løsning, som skifter signalet i lyskrydset efter forudbestemte intervaller. Tidsintervallerne er baseret på forholdet mellem det gennemsnitlige antal biler i timen i begge retninger. I forbindelse med dette lavede vi også en trafikstyret løsning. Denne løsning tjekker antallet af biler i hver af de fire retninger, hvorefter den beregner den optimale grøntid for både den nord- og sydgående retning, og den øst- og vestgående retning. De beregnede grøntider udføres, hvorefter der beregnes nye grøntider. Det kunne konkluderes heraf at den trafikstyrede løsning var markant bedre end den tidsstyrede. Som den primære løsning til optimeringsproblemet udvikledes en RL agent. Agenten bruges som en intelligent løsning, der observerer miljøet omkring den, og bestemmer den optimale afviklingsstrategi ud fra udfaldssandsynligheder og belønninger.

Til at evaluere og bedømme den optimale løsning, lavede vi en simulering, der genererer det miljø, som RL agenten skal agere i. Simuleringen er en visuel repræsentation af lyskrydset, der genererer biler bestemt ved sandsynligheder, der varierer efter tidspunktet på dagen. Simuleringen er blevet simplificeret på baggrund af de antagelser der er beskrevet i kapitel 4.1. Kvaliteten af løsningen vurderes ud fra parametrene: Gennemsnitlig ventetid, maksimal ventetid og maksimal kølængde over et døgn.

Samlet set kan det konkluderes at en RL agent har potentiale som en mere effektiv løsning på problemet end de nuværende afviklingsstrategier, da den gennemsnitlige ventetid pr. bil var lavest, da det var den der styrede omløbstiden i simuleringen.

6.2 Projektforslag

Udviklingen af et intelligent lyskryds er langt fra overstået, når man har lavet en strategi der tilsyneladende virker godt til ét lyskryds. Der er mange faktorer som spiller ind og som man skal tage højde for, når man skal implementere en strategi i et lyskryds. Måden vi har taget højde for de forskellige faktorer, er at gøre brug af en række antagelser, som har gjort vores valg af parametre og lyskrydsets udvikling en del nemmere. Nogle af disse antagelser har været nødvendige, for at kunne holde projektet indenfor de rammer, som er mulige for den tid der er sat af til dette projekt.

Hvis man skulle lave et nyt projekt, ud fra det foregående arbejde vi har lavet indtil nu, kunne man arbejde videre på dette lyskryds hvor man, i stedet for at lave atagelser om f.eks. bilisters kørsel og at det kun er biler der er fokuset, implementerer de faktorer, som i første omgang i vores projekt har været set bort fra ved hjælp af antagelser.

Dette er alle parametre, som vi har valgt at se bort fra eller forsimple vha. antagelser i vores projekt:

- Vejr
- Bilisters kørsel
- Forsimpling af krydset
- Krydset er uafhængigt af andre kryds
- Bilers fart, herunder deceleration og acceleration
- Trafikdataen vi har fået udleveret er ikke misvisende

Disse ting, er nogle som hvis man havde mere tid, kunne arbejde med at implementere i dette projekt.
Derudover hvis vi havde mere tid til projektet, kunne en personlig trafiktælling være med til at bidrage til nogle mere autentiske data, som kan afspejle den trafik, som der er i f.eks. tiden før og efter myldretiden ved det kryds, som vi har valgt at arbejde med.

7 Litteraturliste

Litteratur

[1] Vej Direktoratet: *http://www.vejdirektoratet.dk/DA/viden_og_data/statistik/trafikken%20i%20tal/hvordan_udvikler_trafikken_sig/Sider/default.aspx*
11. oktober - 2017

[2] Vej Direktoratet: *http://vejdirektoratet.dk/DA/viden_og_data/temaer/its/Documents/VD_Bedre_trafiksignaler.pdf*

[3] Shyamalee Manage: *http://www.easts.info/2003journal/papers/0871.pdf*
Oktober - 2003

[4] Wikipedia.org: *https://en.wikipedia.org/wiki/Smart_traffic_light*
12. september - 2017

[5] Steen Lauritzen:
http://www.vejbanken.dk/Net_Docs/CFP_Artikler/1117.pdf

[6] Aalborg kommune:
https://www.aalborg.dk/trafik-pas-og-transport/trafik/trafikregulering/signalanlaeg

[7] Wikipedia: *https://en.wikibooks.org/wiki/Fundamentals_of_Transportation/Queueing*
9. September - 2017

[8] Wikipedia: *https://en.wikipedia.org/wiki/Traffic_flow*
6. november - 2017

[9] Queueing analysis: incidents in traffic and it's effects on traffic queues: *nptel.ac.in/courses/105101008/downloads/cete_45.pdf*

[10] ororsociety: *http://ocw.nctu.edu.tw/course/tco001/Millers_Paper.pdf*

[11] Mikkel Færgemand: *http://projekter.aau.dk/projekter/files/259735843/Adaptiv_signalstyring_i_realtid.pdf*
Juni - 2017

[12] Mathworks.com : Foundations of Fuzzy Logic
https://se.mathworks.com/help/fuzzy/foundations-of-fuzzy-logic.html
Downloadet 22. November - 2017

[13] Marco Wiering, Jelle van Veenen, Jilles Vreeken, Arne Koopman
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1.5180&rep=rep1&type=pdf
9. Juli - 2004

[14] Petr Cintula, Christian G. Fermüller, and Carles Noguera
https://plato.stanford.edu/entries/logic-fuzzy/
18. Juli - 2017

[15] Ms. Girija H Kulkarni, Ms. Poorva G Waingankar
http://lib.tkk.fi/Diss/2002/isbn9512257017/article2.pdf
December - 1998

[16] Pedro Albertos, Atonio Sala
http://personales.upv.es/asala/publics/papers/C24ALCA98.pdf
14. September - 1998

[17] Carlos M. Fonseca og Peter J. Fleming:
https://pdfs.semanticscholar.org/11ec/6378801e6f8a800260b565439ff1cd9c5f5a.pdf
1993

[18] David L. Poole og Alan K. Mackworth:
https://www.researchgate.net/file.PostFileLoader.html?id=5440e3bdd5a3f298288b45fe&assetKey=AS%3A2
November 2017

[19] Computerphile: *https://www.youtube.com/watch?v=oSdPmxRCWws*
9. september - 2014

[20] Arel C. Liu T. Urbanik A.G. Kohls:
https://pdfs.semanticscholar.org/8bc8/0ccc97902264e9d98c48f7ad36f7dcd61b3c.pdf
3. Februar - 2010

[21] Classconnection: *https://classconnection.s3.amazonaws.com/608/flashcards/1910608/jpg/fig_4_1_hill-climbing1349062649219.jpg*

[22] Michael Schafer: *http://www.mikeschaferlaw.com/blog/three-reasons-why-people-run-a-red-light.cfm*

[23] COWI - Bo Westhausen
http://www.vejforum.dk/Net_Docs/CFP_Presentationer/3295.pdf

[24] Vejdirektoratet:
http://www.vejdirektoratet.dk/da/viden_og_data/publikationer/sider/publikation.aspx?pubid=000061229
7. marts - 2012

[25] Wikipedia: *https://en.wikipedia.org/wiki/Little%27s_law*

[26] IOSR Journal of Mathematics (IOSR-JM) A comparison between M/M/1 and M/D/1 queuing models: *http://www.iosrjournals.org/iosr-jm/papers/Vol11-issue1/Version-2/B011121315.pdf*

[27] WATER RESOURCES SYSTEMS PLANNING AND MANAGEMENT
https://ecommons.cornell.edu/bitstream/handle/1813/2804/05_chapter05.pdf;sequence=16
2005

[28] Wikipedia: Poissonfordelingen
https://en.wikipedia.org/wiki/Poisson_distribution

[29] Artificial intelligence 2E: Foundations of computational agents: Kaptiel 12: learning to act
http://artint.info/2e/html/ArtInt2e.Ch12.html

[30] Google Maps: sattelit billede af krydset
https://www.google.dk/maps

[31] Trafitec; Poul Greibe:
Accelerations- og decelerationsværdier for personbiler
http://www.trafitec.dk/sites/default/files/publications/notat%20acc-dec%20vaerdier.pdf Oktober 2009

[32] Trafik og Veje; Thomas Roslyng:
http://asp.vejtid.dk/Artikler/2016/06-07/8445.pdf

[33] Ingeniør ved Aalborg Kommune; *Bilag 1*

8 Bilag

8.1 Bilag 1

Email fra anonym ingeniør hos Aalborg Kommune, By- og Landsforvaltningen:

Vi har hidtil brugt elektromagnetiske spoler, som også kaldes induktive detektorer, til detektering af køretøjer i trafikstyrede signalanlæg i de fleste kryds. Spolerne er forbundet med styreapparatet og via den apparatets detektorlogik anvendes til registrering af køretøjer til brug for signalanlæggets funktion, for eksempel anmeldelse for signalskift eller forlængelse af grøntider. Spolerne er forholdsvis billige at etablere, men forholdsvis dyre i drift og vedligeholdelse.

I et mindre antal kryds anvendes videodetektering. Videokameraer opsættes i master og forbindes med styreapparatet, og virtuelle spolers placering indkodes i kameraets synsfelt. De virtuelle spoler har i princippet samme funktion som almindelige skoler.

I et enkelt signalanlæg i krydset Jyllandsgade – Dag Hammerskjølds Gade er der anvendt radardetektorer. Ved brug af radar defineres virtuelle spoler som ved videodetektering. Vi forventer at radarer i fremtiden vil blive anvendt de fleste steder.

Video og radarer er forholdsvis dyre i anskaffelse, men billige i drift. Især radarer har også en række funktionelle og trafiktekniske fordele i forhold til traditionelle spoler, som I kan læse om i vedlagte artikel.

8.2 Bilag 2

Handling: Gør intet

Objekt:	Partner:	Ændring:	Metode:
BilerL	Signaltilstand [Grøn, gul]	Ned i interval	Formel 3
BilerL	Signaltilstand [Rød]	Ned i interval	0%
BilerL	Signaltilstand [Grøn, gul]	Op i interval	Formel 4
BilerL	Signaltilstand [Rød]	Op i interval	Ikke beskrevet i rapporten
BilerL	Signaltilstand [Grøn, gul]	Bliv i interval	Formel 5
BilerL	Signaltilstand [Rød]	Bliv i interval	Ikke beskrevet i rapporten
Signaltilstand	Signaltilstand [Grøn, Rød] [Rød, Grøn]	Signaltilstand [Sig selv] [Sig selv]	100%
Signaltilstand	Signaltilstand [Grøn, Rød] [Rød, Grøn]	Signaltilstand [Alle andre] [Alle andre]	0%
Signaltilstand	Signaltilstand [Rød-Gul(Grøn)] [Rød-Gul(Rød)] [Gul(Grøn)-Rød] [Gul(Rød)-Rød]	Signaltilstand [Rød-Grøn] [Gul(Grøn)-Rød] [Grøn-Rød] [Rød-Gul(Grøn)]	StepSize/GulTid
Signaltilstand	Signaltilstand [Rød-Gul(Grøn)] [Rød-Gul(Rød)] [Gul(Grøn)-Rød] [Gul(Rød)-Rød]	Signaltilstand [Sig selv] [Sig selv] [Sig selv] [Sig selv]	1-(StepSize/GulTid)
Signaltilstand	Signaltilstand [Rød-Gul(Grøn)] [Rød-Gul(Rød)] [Gul(Grøn)-Rød] [Gul(Rød)-Rød]	Signaltilstand [Alle andre end de to overstående] [Alle andre end de to overstående] [Alle andre end de to overstående] [Alle andre end de to overstående]	0%
Tid i signaltilstand	Tid i signaltilstand [0;15] [16;119] [120;Og over]	Et interval op	1/IntervalStørrelse
Tid i signaltilstand	Tid i signaltilstand [0;15] [16;119] [120;Og over]	Alle andre intervaller end en op og bliv	0%
Tid i signaltilstand	Tid i signaltilstand [0;15] [16;119] [120;Og over]	Bliv i interval	(IntervalStørrelse- 1)/IntervalStørrelse

Figur 19: *Sandsynligheder ved handlingen: Forlæng grøntid*

8.3 Bilag 3

Handling: Skift signaltilstand (Antaget at handlingen er mulig - Ellers = 0%)

Objekt:	Partner:	Ændring:	Metode:
BilerL	Signaltilstand [Grøn, gul]	Ned i interval	Formel 3
BilerL	Signaltilstand [Rød]	Ned i interval	0%
BilerL	Signaltilstand [Grøn, gul]	Op i interval	Formel 4
BilerL	Signaltilstand [Rød]	Op i interval	Ikke beskrevet i rapporten
BilerL	Signaltilstand [Grøn, gul]	Bliv i interval	Formel 5
BilerL	Signaltilstand [Rød]	Bliv i interval	Ikke beskrevet i rapporten
Signaltilstand	Signaltilstand [Grøn, Rød] [Rød, Grøn]	Signaltilstand [Sig selv] [Sig selv]	0%
Signaltilstand	Signaltilstand [Grøn, Rød] [Rød, Grøn]	Signaltilstand [Gul(rød)-rød] [Rød-gul(rød)]	100%
Tid i signaltilstand	Tid i signaltilstand [0;15] [16;119] [120;Og over]	Alle andre end [0;15]	0%
Tid i signaltilstand	Tid i signaltilstand [0;15] [16;119] [120;Og over]	Tid i signaltilstand [0;15]	100%

Figur 20: *Sandsynligheder ved handlingen: Skift signal*