KU LEUVEN

FACULTEIT WETENSCHAPPEN

Ants in harbours - autonomous vehicles for large-scale harbour operations

Implementing sequential interdependent tasks using Delegate MAS, based on a case study

Raphaël Ottevaere

Promotor:
Prof. dr. T. Holvoet

Assessoren:
Prof. dr. ir. T. Schrijvers
Dhr. Hoang Tung Dinh

Begeleider:
Dhr. Hoang Tung Dinh

Proefschrift ingediend tot het
behalen van de graad van
Master of Science in Toegepaste Informatica
Specialisatie Artificiële intelligentie

Academiejaar 2019-2020

Voorwoord

Ik zou graag iedereen bedanken die mij hielp tijdens het schrijven van deze masterproef. Als eerste wil ik mijn promotor Prof. dr. Tom Holvoet en mijn begeleider dhr. Hoang Tung Dinh bedanken voor hun begeleiding tijdens het ontwikkelen van deze masterproef van begin tot einde. Verder wil ik ook iedereen bedanken die zijn tijd opgaf om deze masterproef na te lezen. Ik zou ook graag mijn vader, broers en zus willen bedanken voor de morele steun tijdens mijn academische jaren. Nu ik de kans krijg, zou ik ook graag mijn grootouders willen bedanken die mij steunden toen ik op mijn diepste punt zat en die nu nog altijd mijn steunpilaar zijn, zoveel jaar later. Daarnaast gaat mijn dank ook uit naar Joren, die mij tijdens de overstap naar het universitaire leven als mentor heeft bijgestaan en die, ook al eindigt mijn universitaire carrière binnenkort, deze rol nog ongetwijfeld zal blijven dragen. Als allerlaatste wil ik nog Manon vermelden, die meer geduld had met mij dan ik voor mogelijk achtte, voor haar hulp bij zowel het nalezen van deze masterproef als voor het bijstaan en oppeppen als ik door het bos de bomen niet meer zag.

Samenvatting

In deze masterproef wordt een oplossing voorgesteld om onderling sequentiële afhankelijke taken op te lossen met delegate multi-agent systemen (DMAS), gebaseerd op een casestudy. De taken in de casestudy kunnen herleid worden tot 'dial-a-ride-problems with deadlines' (DARPWD) door de aanwezige software op de casestudy-site. De onderling sequentiële afhankelijke taken kunnen op twee verschillende manieren worden opgelost. De volledige verzameling van taken kan als één grotere taak worden beschouwd en worden opgelost via coalitievorming. De taken kunnen ook individueel worden opgelost om de grotere taken op een pragmatische manier op te lossen. De onderling sequentiële afhankelijke taken worden in het werk van Korsah et al. [24] omschreven als cross-schedule dependencies singlerobot-taken wanneer alle taken pragmatisch na elkaar worden uitgevoerd om tot het eindresultaat te komen. De taken kunnen volgens Korsah ook als als multirobot-taken worden beschouwd als het verplaatsen van alle auto's op basis van een coalitie als één enkele taak wordt bekeken. Beide methodes worden geïmplementeerd door middel van de miereninfrastructuur van DMAS. DMAS wordt gebruikt om de meest optimale taken en paden te zoeken en die voor de automatic guided vehicles (AGV's) op specifieke momenten te reserveren. DMAS zorgt dus voor de coördinatie tussen meerdere AGV's. Er werden ook preventiemethodes ingevoegd op deadlocks in de oplossing en er werd een oplossingsmethode bedacht voor de deadlocks die toch optraden. Om de twee oplossingsmethodes met elkaar te vergelijken, worden er enkele evaluatiecriteria gehanteerd die in meerdere mogelijke scenario's worden getest. Daarvoor wordt de simulator RinSim gebruikt die realtimesimulaties kan uitvoeren en ontwikkeld werd voor MAS. Uit de experimenten blijkt dat de scenario's een grote invloed hebben op de performantie van beide oplossingen. De coalitiemethode laat wel een grotere variabiliteit toe door de maximale grootte van AGV's die deelnemen aan de langzaam groeiende coalitie aan te passen. Er blijkt een verhouding te zijn tussen de rij- en laadtijd die de optimale coalitiegrootte bepaalt. Er is ook een manier nodig om de de AGV's tegen te houden zodat ze andere AGV's niet blokkeren door te snel een parkeerplaats te willen oprijden waar een andere AGV nog aan het laden is. Uit beide oplossingsvoorstellen kan er geen ideale oplossing gekozen worden, onafhankelijk van het scenario waarin de taken zich afspelen. De opties die werden meegegeven aan de oplossingsmethodes, zoals de grootte van de coalitie, spelen ook een grote rol in het bepalen van de performantie per oplossingsmethode.

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

Verklarende woordenlijst

Afkortingen

AGV	Automatic guided vehicle
MAS	Multi-agent systemen
DMAS	delegate Multi-agent systemen
GPDP	general pickup and delivery problem
PDP	pickup and delivery problemen
PDPTW	PDP met time windows
PDPWD	PDP met deadlines
DARP	Dial-a-ride problemen
DARPTW	DARP met time windows
DARPWD	DARP met deadlines
VRP	Vehicle routing problem
VRPPD	Vehicle routing problem with pickup and delivery
BDI	Belief-desire-intention model
SR	Singlerobot-taken
MR	Multirobot-taken
ST	Singletaak-robots
MT	Multitaak-robots
IA	Onmiddelijke toewijzen (Instantaneous assignment)
TA	Tijd verlengde toewijzen (Time-extended assignment)
ND	No dependencies
ID	In-schedule dependencies
XD	Cross-schedule dependencies
CD	Complex dependencies
HoldIA	HoldInfrastructuuragenten

Hoofdstuk 1

Inleiding

In de laatste decennia is het automatiseren van menselijke arbeid steeds belangrijker geworden. Waar de simpele, zware robots in het begin enkel eenvoudige taken konden uitvoeren, zijn die robots geëvolueerd naar complexe machines die zonder invloed van de mens steeds complexere taken kunnen uitvoeren. Automatic guided vehicles (AGV's) bestaan al sinds 1950 en zijn verplaatsbare robots die zich kunnen navigeren. De eerste uitgebrachte AGV was niet meer dan een simpele takelwagen die, in plaats van een spoor, een touw volgde dat over de vloer gespannen was [1]. De huidige AGV's zijn onherkenbaar in vergelijking met die eerste en ook nu nog veranderen ze continu. De AGV's zijn ondertussen ook een onmisbaar element geworden in veel productieprocessen. Tegenwoordig kunnen sommige AGV's zich perfect bewegen in sites waar steeds meer menselijke interacties hen kunnen storen. De AGV's moeten dus naast het beslissen over de paden en taken die ze uitvoeren ook nog eens de omgeving waarin ze zich bevinden, kunnen verwerken en zich over die omgeving verplaatsen zonder aanrijdingen. In de laatste jaren gebruiken meer en meer grote magazijnen, zoals die van Amazon [12], AGV's om hun vracht efficiënt intern te kunnen verplaatsen. De groei van magazijnautomatie blijkt ook niet te stoppen, in de komende vijf jaar wordt verwacht dat die markt in waarde zal verdubbelen.

Met de groei van de capaciteiten van zelfrijdende auto's worden AGV's nu ook interessant om te gebruiken buiten de gecontroleerde omgevingen van de magazijnen. Het plannen van schema's voor de AGV's op zowel pad- als taakniveaus is ook geen klein werk. Iedereen die ooit al eens geprobeerd heeft om grootschalige projecten te organiseren met veel deelnemers kan hiervan getuigen. Wat nu met een project waar een honderdtal AGV's moeten samenwerken om zo snel en optimaal mogelijk een grote hoeveelheid taken uit te voeren? En hoe plannen die AGV's dat in met minimale kosten zonder over enige kennis te beschikken van de locatie en plannen van de andere AGV's? In de standaardgevallen wordt er een taakplanning opgemaakt via een gecentraliseerde server die de AGV's zal aansturen en de beste taakschema's per AGV zal berekenen [29]. In sommige gevallen gebeurt dat ook met inspraak van de AGV's via een combinatie van een gecentraliseerde en gedecentraliseerde methode. Dat betekent echter dat alle communicatie via één enkel punt moet verlopen. In echt grote netwerken met honderden AGV's op een oppervlakte van enkele vierkante kilometers is dat een groter probleem. De accurate werkwijze en de steeds groeiende capaciteiten van de AGV's maken ze steeds interessanter om te gebruiken voor industriële doeleinden om zo een een groot voordeel te behalen op concurrerende bedrijven. De vraag is nu hoe een echt grote hoeveelheid AGV's kan samenwerken om

complexere taken uit te voeren en hoe die AGV's onderling afspraken kunnen maken zonder dat ze echt met elkaar kunnen communiceren. Ons onderzoek bevindt zich ook op de groeiende markt en is onderdeel van een haalbaarheidsonderzoek om een kade te automatiseren door gebruik te maken van AGV's om auto's te verplaatsen. Delegate multi-agent systemen gebruiken mierberichten om op een gedecentraliseerde manier grote, complexe opdrachten te kunnen uitvoeren in dynamische omgevingen zoals deze kade.

1.1 Probleemstelling

Het plannen van een distributie wordt in de literatuur pickup-and-delivery problemen (PDP) genoemd en werd door Savelsbergh en Sol [32] beschreven. In die problemen worden, net zoals de naam beschrijft, goederen op een locatie opgepikt en op een andere locatie terug afgeleverd. In het basisgeval ondervindt dit probleem geen invloed van buitenaf, dat is echter vaak de uitzondering. Dynamische PDP krijgen tijdens het uitvoeren van de taken te maken met routes die wegvallen of zelfs mogelijke AGV's die wegvallen. Dat houdt in dat er bij elke gebeurtenis van die aard een mogelijke herberekening kan plaatsvinden of het schema van taken die die AGV en alle andere AGV's hebben, nog optimaal is. In het specifiekere dial-a-ride-probleem (DARP) worden dynamische nieuwe orders doorheen de dag bijgevoegd aan de uit te voeren taken. Volgens Cordeau en Laporte [5] is het deur-naar-deurtransport zoals taxi's het meestvoorkomende voorbeeld van DARP. Traditioneel zal een PDP-algoritme de berekening van de meest optimale taaktoewijzingen slechts een paar keer per dag berekenen. In het geval van taxi's is het niet moeilijk om in te zien dat dit problemen kan veroorzaken als men een taxi maar op een paar vaste tijdstippen zou kunnen bestellen. De planning van de AGV's zal zich doorheen de dag dynamisch moeten vernieuwen. In dial-a-ride-problemen met tijdslots zullen de DARP samengaan met een specifiek tijdslot waarin de pickup en de levering moeten gebeuren.

Het oplossen van zulke dial-a-ride problemen volgt twee grote denkpistes [29], namelijk via een centraal of een gedecentraliseerd algoritme waarin de AGV's meer beslissingsmacht hebben. Het nadeel van de gedecentraliseerde oplossingen is het ontbreken van globale informatie over de andere AGV's tijdens het beslissingsproces. De voordelen ervan zijn de beter schaalbare natuur en de dynamische beslissingskracht van de AGV waardoor vaker naar oplossingen kan worden gezocht dan bij een centraal algoritme. In een geval waar honderd AGV's zich moeten coördineren op een kade waar mensen de AGV sterk kunnen storen, is een centraal algoritme al een stuk ingewikkelder. Als er bovendien nog spontane dringende taken worden toegevoegd, zoals kenmerkend bij een DARP, blijken de gedecentraliseerde algoritmen een stuk interessanter. Multi-agent systemen (MAS) zijn zo een voorbeeld van een typische gedecentraliseerde oplossing. In MAS beslissen verschillende agenten over de beste actie die ze kunnen uitvoeren. Delegate MAS is een uitbreiding op MAS dat met een feromoneninfrastructuur en mierberichten informatie rondom zich verzamelt om zo betere beslissingen te nemen in beperkte zoekruimtes. Onderling sequentiële afhankelijke taken zijn taken die samen moeten worden uitgevoerd, of in een bepaalde volgorde. Voor het uitvoeren van die taken moeten meerdere AGV's samenwerken. Dit bemoeilijkt het probleem van taakschema's opnieuw doordat AGV's hun keuzes op elkaar zullen moeten afstemmen. Dat is zonder directe communicatie tussen de AGV's niet zo eenvoudig, maar DMAS biedt daartoe een oplossing. RinSim is een simulator ont-

wikkeld in de Imec-Distrinet-onderzoeksgroep van KU Leuven, door van Lon en Holvoet [37]. RinSim is specifiek ontwikkeld voor onderzoek rond het General PDP-probleem met tijdsloten in academische kringen en past dus perfect in het onderzoek naar die onderling sequentiële afhankelijke taken in een PDP-omgeving met DMAS.

1.2 Doelstellingen

De casestudy laat ons toe om onderling sequentiële afhankelijke taken te bestuderen in een werkelijke situatie en met die taken een realistisch DARPWD-scenario op te zetten. Dit biedt een mooie kans om de DMAS-architectuur aan te passen en uit te breiden om de paden en taken te verkennen en vast te zetten. We houden ons ook aan de beperkingen die ons opgelegd worden in hoofdstuk 3. Deze masterproef is onderdeel van een groter haalbaarheidsonderzoek om vast te leggen hoe een deel van de verplaatsingen van het bedrijf in de casestudy kan worden geautomatiseerd. Het hoofddoel van deze masterproef is dus om een oplossing aan te bieden die AGV's onderling sequentiële afhankelijke taken laat uitvoeren met behulp van DMAS. Bijkomend is het de bedoeling om de oplossingsmethode formeel te specifiëren aan de hand van het werk van Korsah et al. [24] en het effect van de oplossingsmethodes op de AGV's en hun coördinatie gebaseerd op meerdere situaties en variabelen te beschrijven. Hieruit volgt dan de volgende centrale onderzoeksvraag:

Kunnen onderling sequentiële afhankelijke taken efficiënt in een DMAS-omgeving geïmplementeerd worden?

Met als deelvraag:

Wat is het effect van de oplossingsmethode op de AGV's en hun coördinatie?

We onderscheiden hier ook enkele hypotheses. Ten eerste zal er een manier moeten worden gevonden waardoor de AGV's elkaar niet storen wanneer één AGV laadt en de andere de parkeerplaats benadert. Gebeurt dit niet, dan zullen de deadlocks die dit proces ontwikkelen de performantie van de oplossing sterk doen dalen. Ten tweede zal de performantie van de AGV's afhangen van hun verhouding tussen rij- en laadtijd. Dit gebeurt voornamelijk in de coalitiemethode waar meerdere AGV's een punt kunnen benaderen. Als meerdere AGV's wachten omdat ze te snel aan het laadpunt aankomen, zal de performantie van het systeem lager liggen.

1.3 Inhoud

De theoretische grondslag van deze masterproef, in de vorm van een literatuurstudie, wordt in hoofdstuk 2 besproken. Die achtergrond focust zich vooral op PDP, DARP, MAS en de state-of-the-art op het vlak van taakallocatie. De opdeling van de state-of-the-art-taakallocatie wordt gebruikt om de mogelijke oplossingsmethodes die DMAS kan bieden te identificeren. Hoofdstuk 3 biedt meer details over de casestudy die in de vorige subsectie werd ingeleid. Die casestudy bevat meerdere soorten onderling sequentiële afhankelijke taken die kunnen worden gebruikt om een eerste oplossing te testen en te bestuderen. In hoofdstuk 4 wordt de DMAS-oplossing die bovenop RinSim wordt gebouwd, uitgelegd.

Die DMAS-oplossing behandelt zowel het toewijzen van taken aan de AGV's, het bepalen van de paden van de AGV's en hoe deadlocks worden opgelost en vermeden. Wat het effect is van de DMAS-oplossing en welk effect de verschillende variabelen hebben op de oplossing wordt in hoofdstuk 5 verder besproken. Tot slot wordt er in hoofdstuk 6 een besluit geformuleerd. Naast het besluit worden er ook eventuele optimalisaties en vervolgonderzoeken aangekaart.

Hoofdstuk 2

Literatuurstudie

Deze literatuurstudie geeft een overzicht van de gangbare concepten en begrippen die noodzakelijk zijn binnen het onderwerp van deze studie. Het literatuuronderzoek begint met een globaal overzicht van Pickup and Delivery Problems. Vervolgens worden Multi-agent systemen en delegate multi-agent systemen grondig besproken. Daarna bekijken we taakallocatie in multirobot-systemen en afsluiten doen we met bijkomende informatie over deadlocks en de gebruikte simulator, RinSim.

2.1 Pickup and delivery problems

Dit deel gaat dieper in op Pickup and Delivery Problems (voortaan PDP). PDP hebben verschillende vormen en zijn vaak het onderwerp van wetenschappelijk onderzoek geweest. In dit hoofdstuk halen we enkel aan wat noodzakelijk is voor het verloop van deze masterproef.

2.1.1 General PDP

De General Pickup and Delivery Problem (GPDP) kan volgens Savelsbergh en Sol [32] beschreven worden als een geconstrueerde set van routes in een volgorde die aan de transportaanvragen voldoet. Die set van routes kan door een groep voertuigen uitgevoerd worden. In veel toepassingen wordt zo'n set dagelijks berekend, terwijl andere toepassingen die berekeningen meermaals per uur kunnen uitvoeren. Elke transportaanvraag heeft een lading, een beginpunt en een eindpunt. Het oplossen van een PDP wordt beschouwd als NP-Hard [2, 35, 32]. PDP kunnen ook afhangen van tijdstippen. Daarnaar wordt verwezen als Pickup and Delivery Problems with Time Windows, afgekort als PDPTW. Die hebben naast een lading, een begin- en eindpunt ook nog een start- en eindtijdslot. Tijdens die twee tijdslots moet de taak uitgevoerd worden. Als het starttijdstip ook het begintijdstip is van de routes, dan wordt dit een Pickup and Delivery Problem with Deadlines (PDPWD) genoemd. Volgens Parragh et al. [29] bestaan er meer dan 30 grote varianten op PDP, die zowel geïdentificeerd zijn als onderzocht worden.

Het GPDP kan nog verder worden opgesplitst in dynamische en statische Pickup and Delivery Problems. Een statisch PDP heeft een set van taken die moet worden uitgevoerd en die tijdens de uitvoering ongewijzigd blijft. Daarnaast vallen er bij statische PDP ook geen routes of voertuigen weg, zoals in de praktijk bijna altijd het geval is. Bij

dynamische PDP daarentegen kunnen er bij het uitvoeren van de routes taken worden toegevoegd of verwijderd, waardoor ook nieuwe routes en tijdstippen moeten worden toegevoegd. Routes en voertuigen kunnen ook door realtime omstandigheden wijzigen, door bv. ongevallen of problemen met het voertuig. Volgens Hanif et al. [16] zijn dynamische PDP een stuk lastiger om in te plannen door de bijkomende taken en weggevallen wagens. In dat geval is er veel communicatie nodig tussen de voertuigen om beslissingen te maken. Daarnaast zijn echte PDP grootschalig, met honderden voertuigen en duizenden aanvragen [14]. Schaalbaarheid vormt dus ook vaak een probleem, zeker in combinatie met de NP-harde natuur van GPDP. Als PDP centraal worden opgelost en niet door de individuele agenten, daalt de hoeveelheid noodzakelijke communicatie maar wordt de PDP-oplossing minder dynamisch en minder resistent tegen mogelijke problemen. Om die redenen wordt gedecentraliseerde MAS beschouwd als geschikt voor grootschalige dynamische PDP [9, 39].

2.1.2 Vehicle Routing Problem en zijn heuristische waarden

In het Vehicle Routing Problem with Pickup and Delivery (VRPPD) moet een mogelijks heterogene vloot van pickup-voertuigen vanuit een set van terminals aan een set van transportaanvragen voldoen en die set kunnen uitvoeren [6]. Bovenop deze set van aanvragen kan op elke transportaanvraag ook nog een set van restricties aanwezig zijn, zoals o.a. pickup- en leveringstijdstippen en groeperingen van vrachten. In VRPPD zijn een aantal van de aanvragen vaak op voorhand gekend en kunnen de initiële routes zo worden uitgepland. De aanpassingen en nieuwe aanvragen worden dan ingevoegd in de reeds gemaakte plannen. Als de toevloed van nieuwe taken hoog is, zal het algoritme dat de routes berekent dus veel worden herstart. Dat kan een grote invloed hebben op de optimalisatie van de voorgestelde oplossingen, zeker als die herberekening in korte tijd moet gebeuren. De combinatie met dynamische werkomgevingen verergert dit probleem. In VRPPD worden vaak de volgende heuristische waarden [6, 17] gebruikt om de kwaliteit van een route in te schatten:

- De pickup- en leveringstijd mogen niet eerder of later vallen dan de aangevraagde tijdslots
- De wachttijd en en de rijtijd zijn beperkt
- Voertuigen mogen vaak niet inactief zijn terwijl passagiers en pakketten vervoerd worden
- De routes voor de voertuigen moeten altijd uitvoerbaar zijn, met het zicht op pauzes voor bestuurders en beperkingen op het bereik van voertuigen
- Minimaliseren van transportkosten, rijafstanden en operatiekosten

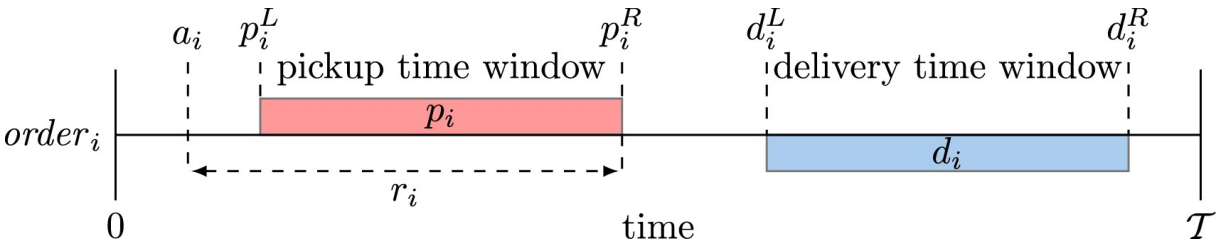
2.1.3 Dial-a-ride met deadlines

Dial-a-ride problemen (DARP) zijn een subcategorie van de GPDP en zijn een dynamische versie van VRP. In tegenstelling tot GPDP is er geen centraal depot vanwaar alle pakketten beginnen, maar zijn de startpunten verspreid over de volledige omgeving.

Volgens Cordeau en Laporte [5] is het meestvoorkomende voorbeeld van DARP het deur-naar-deurtransport, zoals taxi's. In de frequentste versie van DARP wordt het transport behandeld door een set van identieke voertuigen, een homogene vloot. DARP generaliseren een aantal van voertuigrouteringsproblemen zoals VRP en VRP with Time Windows. Door de meest frequente toepassing, namelijk het deur-naar-deurtransport, wordt de kwaliteit van de dienstverlening (quality of service) vaak ook als een belangrijk criterium beschouwd. DARP met deadlines is een bijzonder geval van het generieke DARP. In DARP met deadlines wordt er namelijk geen begintijdstip opgegeven en moet de taak slechts tegen een bepaald tijdstip uitgevoerd zijn. Hetzelfde effect kan worden verkregen als het DARP door een applicatie pas zichtbaar wordt gemaakt voor de gebruikers als de begintijd ingaat; een gebruiker moet dan niet nagaan of hij of zij te vroeg zal aankomen voor de pickup op de gewenste locatie. PDPTW kunnen herleid worden naar DARP met deadlines. Dat kan door de problemen als aparte vrijgekomen taken te beschouwen, aangeboden op tijdstippen die overeenkomen met het begintijdstip van de taak en waar geen eindtijdstip voor het pickuptijdslot bestaat.

Daarnaast kunnen DARP nog verder worden opgesplitst in DARP met verschillende niveaus van heterogeniteit in de pakketten en voertuigen [17]. Tijdens het zoeken naar een passende oplossing moet er rekening worden gehouden met die graad van heterogeniteit. Daarnaast wordt er bij sommige DARP ook rekening gehouden met mogelijke overstappen van pakket en passagier, vaak in een depot of tijdelijke stop. Hier worden de pakketten dan herverdeeld voor verdere optimalisatie. De populairste doelfuncties van DARP zijn het minimaliseren van de operatiekosten door het minimaliseren van de transportkosten [17]. Daarnaast lost een aanzienlijk aantal DARP nog meer multi-objectieve problemen op die vaak herleid worden naar een eenduidig doel via gewogen sommen.

In DARP met tijdsloten kan er een tijdslot zijn voor zowel pickup als de leveringen, zoals voorgesteld in Figuur 2.1. Hierbij is de pickup traagheid gelijk aan de pickuptijd $t_p - p_i^R$ en leveringstraagheid gelijk aan leveringstijd $t_d - d_i^R$, met p_i^R en d_i^R respectievelijk het eindtijdstip van de pickup tijdslot en de leveringstijdslot. Bij sommige optimale oplossingen worden die tijdslots deels verbroken om in latere delen van het algortime meer optimale beslissingen te nemen. De tijd die verstrijkt na het einde van het pickuptijdslot en voor het effectieve oppikken, wordt pickup traagheid genoemd en de tijd die verstrijkt na het einde van het leveringstijdslot en het effectieve afleveren is de leveringstraagheid [36]. Die traagheid wordt dan verrekend in de gewogen sommen als een negatieve waarde. Traagheid kan ook gebruikt worden als een evaluatiewaarde die de precisie van pickups en leveringen aanduidt. Belangrijker is echter de tijd r_i of de reactietijd tussen het invoeren van een order en het tijdstip waarop dit order moet worden opgepikt. In sterk dynamische omgevingen waarin vaak orders bijkomen met een kleine reactietijd, is het herberekenen van een tijdschema voor verschillende voertuigen niet altijd mogelijk. Prioriteiten en lexicografische orders van de doelfuncties worden ook gebruikt. Daarin worden eerst de hoogste prioriteiten geoptimaliseerd en daarna worden de onderliggende prioriteiten in lagen verder geoptimaliseerd zonder de optimalisaties in de hogere lagen aan te tasten.



Figuur 2.1: Visualisatie van tijdgerelateerde events van een enkel order, overgenomen uit Lon et al.[36]. Met a_i het invoermoment van de taak, p_i^L de start van de pickuptijdslot, p_i^R het einde van de pickuptijdslot, d_i^L de start van de leveringstijdslot, d_i^R het einde van de leveringstijdslot en r_i de reactietijd tussen het invoeren van een taak en het einde van de pickuptijdslot.

2.2 Multi-Agent Systemen

Multi-Agent Systems (MAS) zijn een veelgebruikt ontwerp voor het bouwen van gedecentraliseerde oplossingen voor problemen. In MAS zullen de probleemoplossers, zoals voertuigen en pakketten, gemodelleerd worden als autonome en samenwerkende agenten. Door de gedecentraliseerde implementatie is er minder nood aan gecentraliseerde data of controle, met als gevolg dat er betere opties zijn voor het schalen van de oplossingen [14]. MAS die gespecialiseerd zijn in het oplossen van PDP zijn moeilijker te ontwerpen door de grote hoeveelheid cohesie en samenwerking die de vele agenten aan de dag moeten leggen.

2.2.1 Agent

In deze paragraaf werpen we een blik op agenten en hun belangrijkste functies en eigenschappen in een MAS-omgeving. Een algemene definitie van Ali Dorri et al. [8] luidt als volgt:

"An entity which is placed in an environment and senses different parameters that are used to make a decision based on the goal of the entity. The entity performs the necessary action on the environment based on this decision".

Deze definitie telt vier kernwoorden, zijnde entity, environment, parameters en actions. Het doel van elke agent is het oplossen van een specifieke taak of actie in een environment of omgeving zonder menselijke invloed, met informatie verzameld vanuit die omgeving. Volgens Wooldridge et al. [40] zijn er vier belangrijke modellen om agenten te ontwerpen. Eerst bespreken we de deductieve redeneringsagenten. Die nemen al hun beslissingen vanuit logische redeneringen, bijvoorbeeld predicatenlogica met een duidelijke structuur. Die logische expressies zullen echter complex zijn en soms onvolledig door de moeilijkheid om alles voor te stellen in pure logica. Het oplossen en bewijzen van die logische expressies kan ook enige tijd vergen door de complexiteit ervan.
Een tweede soort is de praktische redeneringsagenten. Daarbij ligt de aandacht op de acties die de agenten ondernemen. Dat betekent dat er eerst een set van mogelijke acties moet worden opgemaakt en dat de agent vervolgens de beste keuze moet nemen uit die set van mogelijke acties. Het belief-desire-intention model, voorgesteld in sectie 2.2.3, is

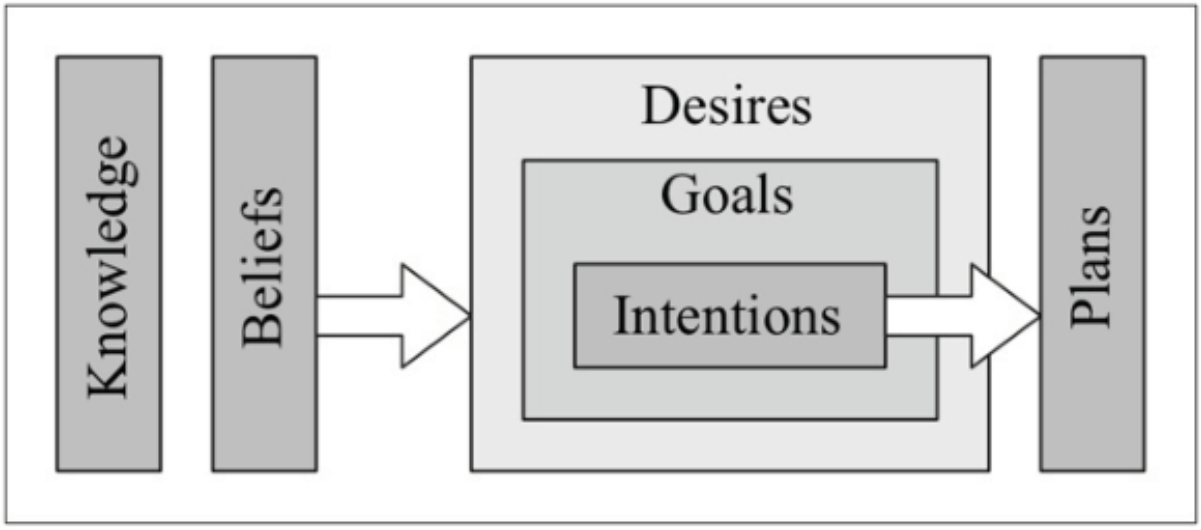
hiervan een vaak gebruikt model.
Voor het oplossen van de problemen gerelateerd aan symbolische redeneringen zijn reactieve agenten aangeraden. Daarbij wordt het gedrag van de agenten bepaald door een set van als-dan regels, in tegenstelling tot de explicite voorstelling van de omgeving of het redeneren met logica.
Als laatste zijn de hybride agenten aan de beurt. Zij combineren enkele van de vorige modellen zodat ze zowel logisch kunnen redeneren als reageren op de omgeving. Daardoor kunnen nog complexere taken worden uitgevoerd, al leidt dat ook tot complexere agenten.

2.2.2 Zwermintelligentie

Zwermintelligentie (swarm intelligence) is een subcategorie van MAS waarbij software wordt onderverdeeld in identieke softwarecomponenten. Die kleine agenten kunnen grote hoeveelheden complexe berekeningen uitvoeren door de interactie met de omgeving en met zichzelf. Systemen die gebruikmaken van zwermintelligentie worden aangenomen om gedecentraliseerd en zelf-organiserend te zijn. De agenten kunnen enkel op lokaal niveau interacties aangaan met andere agenten maar complexere operaties komen voort uit die lokale interacties. De inspiratie voor die werkwijze komt uit de natuur. Mieren gebruiken namelijk feromonen om lokale beslissingen door te geven aan een groot netwerk van andere mieren. Op dezelfde manier kunnen systemen gebaseerd op zwermintelligentie ook feromonen achterlaten om hun beslissingen door te geven naar andere agenten in het systeem. Die feromonen tonen lokale beslissingen aan die niet noodzakelijk de eindbeslissingen zijn die de agenten zullen nemen. Daarom mogen ze niet worden beschouwd als volledige waarheid tot een agent zijn keuze terug bevestigt en blijft herbevestigen. agenten kunnen namelijk van keuze veranderen, zoals aangetoond in subsectie 2.2.3. Daarom moeten feromonen vluchtig van aard zijn en hebben ze constante herbevestiging nodig. Vervolgens kunnen de feromonen beslissingen en informatie doorgeven aan andere agenten zonder dat er directe communicatie nodig is. Zo kunnen grootschalige operaties van agenten beslissingen maken op basis van lokale informatie, zonder dat ze over alle informatie moeten beschikken.

2.2.3 Belief-Desire-Intention systems

Het Belief-Desire-Intention software model [30] (BDI) is een softwaremodel gericht op de implementatie van intelligente agenten en wordt voorgesteld in Figuur 2.2. Wat de agent gelooft dat de werkelijke situatie is, wat hij wenst te doen en zijn uiteindelijke keuze worden losgekoppeld van elkaar. De beliefs van een agent of de kennis waarover hij beschikt hangen af van de applicatie. Dat kunnen onder meer een kaart van de directe omgeving zijn of communicatiekanalen met mogelijke partners. De wensen of desires daarentegen zijn de mogelijke oplossingen die een agent kan nemen om zijn doel te bereiken. Die wensen kunnen daarna geüpgraded worden naar een intentie waartoe de agent zich wijdt. Om van een desire naar een intentie te gaan, moet een selectiepatroon worden gevolgd, dat afhangt van applicatie tot applicatie en in dezelfde applicatie ook kan afhangen van de uiteindelijke einddoelen van een agent, zoals snelheid versus veiligheid. Deze keuzes worden vaak genomen op basis van heuristische waarden die de agent verkrijgt door zijn kennis uit te buiten. Het is ook belangrijk dat een BDI-agent zijn intenties kan herevalueren, er kunnen zich namelijk veranderingen voordoen in de omgeving waardoor



Figuur 2.2: Belief-Desire-Intention model voor redenering procedures [27].

de eerdere intentie op een later tijdstip niet meer wenselijk is. Bij kalme strategieën duurt het langer vooraleer een intentie ongeldig verklaard wordt en er over een nieuwe wordt beslist. Een nerveuze strategie daarentegen laat een intentie sneller vallen indien een andere desire beter wordt geacht. Dat betekent echter dat intenties bij te nerveuze agenten te snel worden herzien, waardoor de agent te snel nieuwe beslissingen zal nemen en het systeem chaotisch maakt. Als de agenten te kalm zijn, worden de intenties te lang behouden en vermindert de dynamische aard van het systeem omdat slechte beslissingen te lang behouden worden. Dat kan leiden tot een minder efficiënt systeem.

2.3 Delegate MAS

Er zijn enkele opvallende kenmerken van PDP die het ontwikkelen van een oplossing, zeker met interacties tussen agenten, moeilijk maken [15]. De uitdagingen komen naar voor in een dynamische en schaalbare oplossing. Desondanks argumenteren onder andere Fisher et al. [10] dat MAS geschikt is voor een transportdomein. Dit komt voornamelijk door de gedecentraliseerde aard van het transportdomein. Daarnaast is gedecentraliseerde MAS beter in staat om dynamische omgevingen correct te behandelen bij beperkte tijd. Bovendien willen commerciële bedrijven vaak geen details vrijgeven voor globale optimalisaties, waardoor lokale informatie geregeld als vervanger dient. Door de verschillen in informatiebeschikbaardheid tussen de lokale agenten en de beslissingsmaker op hoogste niveau bestaat er een onvermijdelijke beperking op dergelijke volledige gedecentraliseerde systemen. Hiervoor wordt er een aanpak op twee niveaus geïntroduceerd waarbij de top-level agenten informatie verzamelen en distribueren naar de lokale agenten. Die maken dan de effectieve, voor hen optimale beslissingen. De uitdaging aan de dynamische kant van PDP is de dynamische omgeving, vaak veroorzaakt door files, werkzaamheden, aankomst van nieuwe taken, annuleringen en de dynamische aankomst van andere agenten. Het schalingsprobleem is vanzelfsprekend, honderden robots worden geacht samen te werken over vaak grote afstanden. Volgens Holvoet et al. [15] zijn de belangrijkste uitdagingen de volgende:

- Beperkte middelen (resources): niet elke agent kan accurate en globale informatie bijhouden door communicatie en beperkte opslagcapaciteit. Daarnaast zijn ook de computationele middelen beperkt. Er moet bovendien rekening worden gehouden met enkele fysieke restricties, zoals de snelheid, maximale rijafstand, capaciteit van de weg, enz.

- De beslissingen van een agent beïvloeden de beslissingen van elke andere agent. De beslissingen zijn daarom sterk samenhangend. Als een agent een beter pakket vindt om op te pikken, kan dat pakket al gereserveerd zijn door een andere agent en kan het niet worden opgepikt zonder de beslissingen van een andere agent te dwarsbomen. Als de agent al op weg was naar een andere reservatie en er een betere beschikbaar wordt, kan de oorspronkelijk geplande pickupreservatie geannuleerd worden, met de nieuwe pickup als voorkeur. Die annulatie kan dan een betere oplossing betekenen voor een andere agent. Dat impliceert ook dat de agenten elkaar impliciet of expliciet moeten informeren van hun keuzes en zelf moeten bijhouden welk pakket ze gereserveerd hebben.

- Beslissingen die gebaseerd zijn op informatie die enkel lokaal beschikbaar is, zijn op globaal niveau niet altijd optimaal, waardoor de globale oplossing ook niet optimaal kan zijn.

De rest van dit deel loopt verder als volgt, in onderdeel 2.3.1 wordt dieper ingegaan op de patronen gebruikt in delegate MAS (DMAS) en de redenen waarom de patronen zijn ingevoegd. Onderdeel 2.3.2 legt verder uit hoe die aparte patronen samen de coördinatie van de agenten regelen. De informatie uit de eerste twee paragrafen wordt vervolgens verder samengevoegd tot het multirobot-routeringsprotocol door middel van DMAS van Lon et al. [7]. In het laatste deel 2.3.3 wordt een protocol besproken dat de verspreiding van vertraging doorheen het netwerk behandelt.

2.3.1 DMAS Patronen

Het allereerste patroon dat DMAS gebruikt is een slimme-bericht-patroon (smart-message pattern) [15] dat is ingevoerd in een omgeving waarin alle knopen connecties hebben met hun naburige knopen, met gevarieerde kwaliteitsniveaus. Vaak is het in dergelijke contexten niet mogelijk om simpele en directe interacties te hebben door agenten, vooral in situaties waarin er geen accurate globale informatie beschikbaar is. Een slim bericht kan zich autonoom doorheen het systeem bewegen en interageren met andere knopen op aanvraag van de versturende agent. Het gedrag van een agent is gericht op:

- Het opvragen van lokale informatie op knopen en berekeningen op die lokale informatie

- Migratorisch gedrag ingebouwd in de slimme berichten waardoor deze autonoom doorgestuurd kan worden naar de volgende sprong

- Een manier om zichzelf te klonen of andere slimme berichten aan te maken in een uitvoerende context

De uitvoerende context speelt zich af in een node met computationele kracht, waarin gedurende de context de slimme-berichten kunnen uitgevoerd worden, doorgestuurd worden, of informatie kan verzamelen.

De mieren gebruikt in delegate ant MAS zijn een voorbeeld van een slim bericht die aangepast is voor specifiek gebruik. De mieren worden gekenmerkt door het toevoegen van de volgende patronen [19]:

- Feromoneninfrastructuur die eigen is aan zwermintelligentie zoals besproken in subsectie 2.2.2. Hierbij wordt elke infrastructuurnode voorzien van infrastructuur om feromoondata op te slaan. Die data worden opgeslagen door informatie bij te houden van de mieren die zich over deze node voortbewegen en de feromoondata in een uitvoeringscontext kunnen bereiken. Daarnaast wordt aan knopen ook een verdampingsfunctie toegevoegd.

- Voorwaarts-achterwaarts gedrag: een verschil in gedrag (zowel wat verplaatsings-, reproductie- als uitvoeringsmodus betreft) voor een slim bericht tussen een voorwaartse fase en een achterwaartse fase. De voorwaartse fase dient om de omgeving te verkennen en informatie te verzamelen of om een oplossing te bekomen. In de achterwaartse fase kruipt het slimme bericht terug naar de verantwoordelijke agent die het slim bericht aanvankelijk verstuurd had. Indien noodzakelijk kunnen de feromonen ook geüpdatet worden op weg terug naar de oorspronkelijke verantwoordelijke agent. De fase waarin een mier zich bevindt, wordt opgeslagen in de mier zelf.

Naast de mieren wordt er ook gesproken van de infrastructuuragenten. Die agenten kregen de functie om informatie en notificaties te ontvangen, naast het verwerken van de mieren. De agenten stellen een bepaald stuk infrastructuur voor, zoals wegen zoals in [7], maar stellen in het algemeen alles voor dat geen beslissingen maakt, in tegenstelling tot de beslissingsagenten. Beslissingsagenten zijn praktische redenerings- of mogelijks zelfs hybride agenten die gebruikmaken van een BDI-systeem (zie subsectie 2.2.3) om beslissingen te maken. De beslissingen worden genomen in een gedragsmodule die afgeschermd is van de overige software om complexiteit te onderdrukken [15]. Een gedragsmodule is een strikt gedefinieerd gedrag dat een agent kan uitvoeren om een specifieke taak of actie te vervolledigen. Het gedrag van een agent wordt bereikt door het selecteren en uitvoeren van gedragsmodules op bepaalde tijdstippen, in een specifieke volgorde. DMAS is een gedragsmodule gekoppeld aan extended slimme berichten om informatie te verzamelen om beslissingen te nemen. DMAS wordt ook belast met het maken, uitvoeren en managen van slimme berichten en met het encapsuleren van de slimme berichten voor de overige aanwezige software.

2.3.2 Routeringsprotocol via DMAS

Om een VRP op te lossen met DMAS worden twee hoofdtypes van primaire agenten gebruikt, namelijk voertuigagenten en pakketagenten. Die bevinden zich beide op een graafinfrastructuur (de omgeving) die de wegen voorstelt. Op die graafinfrastructuur worden de verschillende knopen voorgesteld door infrastructuuragenten. De voertuigagenten zijn hybride agenten die hun beslissingen maken volgens een BDI-systeem en ze

zijn verantwoordelijk voor het bewegen over het wegennetwerk en het zoeken naar naburige pakketten. De pakketagenten zijn de taken die zich op het netwerk bevinden. Ze bevatten informatie over zichzelf en passen die data aan tijdens hun levenscyclus. Bij het naderen van de deadline zal het pakket zijn aard veranderen naar kritiek om zijn prioriteit aan te tonen. Om coördinatie tussen de agenten te verwezenlijken worden DMAS-mieren gebruikt. De mieren gebruikt in een VRP-DMAS-oplossing, volgens Holvoet et al. [16], bestaan uit drie grote mogelijke klassen:

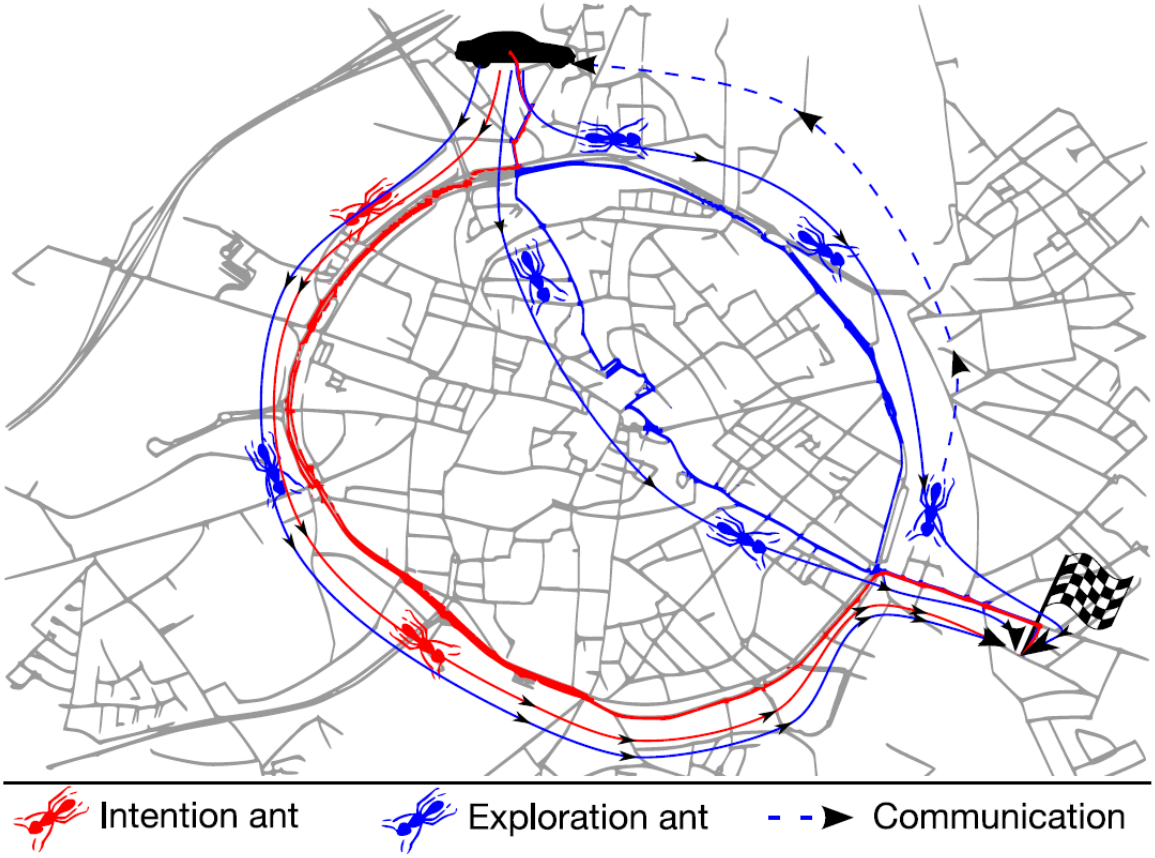
- Haalbaarheidsmieren (feasibility ants): Om lokale informatie te verspreiden naar andere agenten, wordt de informatie gedelegeerd naar een haalbaarheids-DMAS (feasibility DMAS). De haalbaarheidsmieren begeven zich enkel naar naburige andere mieren binnen een bepaalde afstand en laten informatie achter die terugverwijst naar de originele zender. Door gebruik te maken van haalbaarheidsmieren kan informatie gedeeld worden op een gedecentraliseerde manier en ontstaan er lokale paden die kunnen worden gebruikt voor andere DMAS.

- Padexploratiemieren worden uitgestuurd door voertuigagenten om een optimaal pad te vinden. Exploratiemieren overlopen het netwerk op zoek naar informatie die gedeeld is door mieren. Doorgaans worden padexploratiemieren regelmatig verstuurd over meerdere paden om over lokale optima te geraken. Na n sprongen zonder gevonden eindpunt wordt een exploratiemier teruggestuurd. Daarna worden de gevonden informatie en paden gedeeld met de voertuigagent. Op basis van heuristische waarden, die rekening houden met het doel, worden de gevonden paden in een volgorde gerangschikt.

- Intentiemieren zijn verantwoordelijk voor de coördinatie en reservaties tussen de pakket- en de voertuigagenten. Die worden regelmatig herstuurd om de feromonen up-to-date te houden en dienen als doorgeefluik voor de voertuigagenten om hun intenties door te geven naar de omgeving.

Niet alle drie de klassen worden in elke oplossing gebruikt; vaak worden haalbaarheidsmieren weggelaten of gecombineerd met een andere soort mier zoals de exploratiemieren.

De interactie van deze mieren en hun beslissingslogica wordt aan de hand van een voorbeeld geïllustreerd. Figuur 2.3 stelt een grafenkaart voor die het wegennetwerk van Leuven voorstelt [19]. De voertuigagent, voorgesteld door de auto, zoekt een pad naar het eindpunt, voorgesteld door de vlag. Het uitgangspunt van deze situatie is het feit dat de voertuigagent een BDI-schema volgt. Hij kent de omgeving al, namelijk de graaf en de wegen die de graaf kan afleggen en hij gaat ervan uit dat er een mogelijk pad is naar het gewenste eindpunt. Om dat pad te vinden, kunnen enkele mogelijke paden via een A*-padbepalingstechniek gezocht worden. Om de uitvoerbaarheid van de paden te testen, maakt de DMAS gedragsmodule van de voertuigagent drie padexploratiemieren aan. Elke mier, voorgesteld door blauwe mieren, krijgt een mogelijk pad toegewezen, zoals in Figuur 2.3 afgebeeld door blauwe lijnen en wordt vervolgens de omgeving ingestuurd. De omgeving zelf bevat infrastructuuragenten met een feromoonmodule die lokale informatie door middel van haalbaarheidsmieren doorgeven aan naburige infrastructuuragenten. De padexploratiemieren verzamelen bij elke sprong de nodige informatie, in dit geval de drukte, snelheid en verwachte tijdspanne. Op die manier kunnen de mogelijke begintijdstippen



Figuur 2.3: Visualisatie van mierexploratie en -intentie, overgenomen uit Holvoet et al. [19]. Mieren die de omgeving van Leuven (België) verkennen. Exploratiemieren, afgebeeld in het blauw, verkennen de omgeving en verzamelen informatie. Op basis van die informatie maakt de voertuigagent zijn keuze voor een te volgen route en stuurt hij de route door via intentiemieren, afgebeeld in het rood.

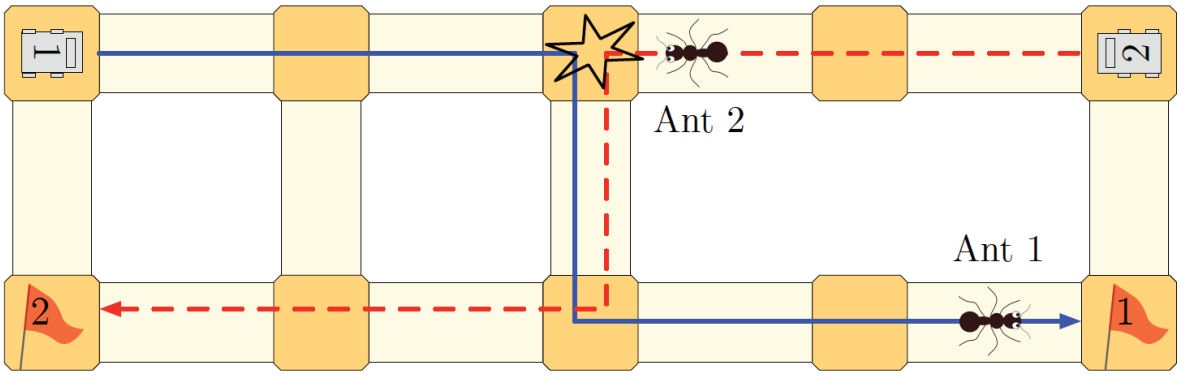
per sprong worden bepaald en kan er een accurate schatting voor het volledige pad worden gemaakt. Na het bereiken van het eindpunt worden de mieren via directe communicatie teruggestuurd naar de originele voertuigagent. De gedragsmodule van de voertuigagent zal nu een beslissing nemen gebaseerd op de vergaarde lokale informatie, gekoppeld aan heuristische beslissingen en waarden. Uit die beslissing blijkt dat het rode pad op Figuur 2.3 verkozen wordt. Om aan te geven dat de voertuigagent dit pad neemt, wordt een intentiemier over dat pad verstuurd die feromonen achterlaat per infrastructuuragent, die als een reservatie dient. Om de feromonen actueel te houden, worden er tijdens het rijden regelmatig nieuwe intentiemieren verstuurd. Het pad-zoek-proces zal tijdens het rijden meermaals herhaald worden op zoek naar nieuwe en betere paden. Als een nieuw en beter pad gevonden en verkozen wordt, zullen de niet meer gebruikte reservaties door de feromoonreservatie niet geüpdated worden en na een tijd verdampen. Zo worden die tijdstippen opnieuw bruikbaar voor latere zoektochten naar een pad.

Het vorige brengt ons naar de gedragsmodule van de voertuigagenten. De module volgt een simpele lus, zoals voorgesteld in Algoritme 1. In de lus worden voortdurend mogelijke paden opgezocht en bekeken of ze uitvoerbaar en mogelijks beter zijn dan de huidige

Algoritme 1 Vereenvoudigde gedrags rederningslus van een voertuigagent

```
1: while canDoAction() do
2:     routes ← getPossibleRoutes(currentLocation,endLocation)
3:     selectedRoute ← choose(routes)
4:     if reviseIntention(currentIntention,selectedRoute) then
5:         currentIntention ← selectedRoute
6:     propagateIntention(currentIntention)
7:     instructDriver(currentIntention)
```

intentie. Daarna wordt de huidige intentie, die eventueel aangepast is, gepropageerd naar de omgeving. Tot slot wordt de intentie doorgestuurd naar de bestuurder en begint de lus opnieuw. Doorheen die iteraties wordt het plan stelselmatig verbeterd.
In tegenstelling tot contextbewuste routering, dat agenten enkel toelaat om sequentiële plannen te maken, laat delegate MAS toe dat agenten parallel plannen maken en conflicten oplossen, die veroorzaakt zijn door tegelijkertijd exploreren en reserveren [7]. Als twee intentiemieren op hetzelfde moment aankomen en een reservatie aanvragen aan een resource agent, dan selecteert de resource agent een willekeurig order om eerst te verwerken. Dat wordt gevisualiseerd in Figuur 2.4. voertuigagent 1 en voertuigagent 2 exploreren routes gelijktijdig. Op dat moment zijn er geen reserveringen. Na het exploreren, kiest agent 1 de volle blauwe lijn terwijl agent 2 de rode stippellijn verkiest. Als de intentiemieren worden aangestuurd om de reservaties vast te zetten en ze zich allebei aan hun gevonden routes houden, zal er een botsing plaatsvinden. De resource agent aanvaardde de reservatie van mier 1 doordat die als eerste aankwam of doordat die mier willekeurig als eerste gekozen werd. Intentiemier 2 stopt en wordt afgewezen, terwijl mier 1 doorgaat en ook een reservatie achterlaat op de andere knopen. Intentiemier 2 rapporteert terug aan voertuigagent 2, die na het afwijzen van de intentiemier opnieuw zoekt naar een nieuw mogelijk pad via exploratiemieren. Doordat de reeds aangemaakte reservaties door mier 2 niet worden geüpdated, zullen ze uit het systeem verwijderd worden en geen verdere impact hebben op andere mogelijke paden.



Figuur 2.4: Twee voertuigagenten die tegelijkertijd middelen proberen te reserveren op twee conflicterende paden. De resource agent op de eerste conflicterende middel verwerkt aanvragen sequentieel. Dat laat de reservatie van intentiemier 1 toe, terwijl de aanvraag van intentiemier 2 wordt verworpen [7].

2.3.3 Vertragingen propageren doorheen het netwerk

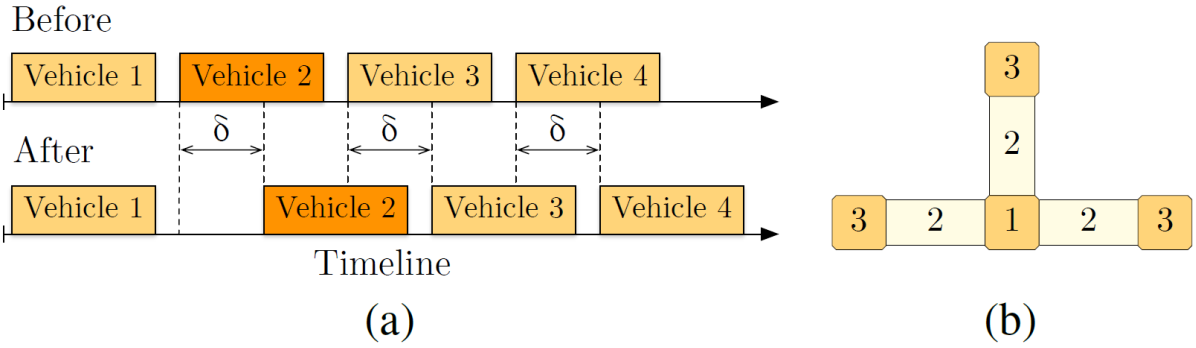
Als er zich een incident voordoet, stuurt de voertuigagent de verwachte of geschatte vertraging δ door en updatet vervolgens zijn originele plan:

$$(\langle r1, [t1_s, t1_e]\rangle, \langle r2, [t2_s, t2_e]\rangle, ..., \langle rn, [tn_s, tn_e]\rangle)$$

tot het nieuwe plan:

$$(\langle r1, [t1_s, t1_e + \delta]\rangle, \langle r2, [t2_s + \delta, t2_e + \delta]\rangle, ..., \langle rn, [tn_s + \delta, tn_e + \delta]\rangle)$$

waar r1 het middel is die de voertuigagent op dat moment inneemt [7]. Daarna stuurt de voertuigagent een intentiemier naar de relevante middelen met de aangepaste tijdstabel. De geüpdate reservaties van een vertraagd voertuig kunnen inconsistentie met bestaande reservaties van andere voertuigen veroorzaken die het knooppunt na het vertraagde voertuig bezoeken, zoals geïllustreerd in Figuur 2.5 a. De resource agent geeft de vertraging aan al zijn naburige resource agenten aan, die de vertragingen recursief toepassen aan alle getroffen en toekomstige voertuigen (zie Figuur 2.5 b). Die manier om vertragingen te propageren doorheen de volledige virtuele omgeving, garandeert dat de plannen consistent en deadlock vrij blijven na het updaten indien ze voor het updaten al consistent en deadlock vrij waren. Het informeren over de gewijzigde plannen gebeurt via intentiemieren op dezelfde manier waarop de informatie van de originele plannen naar de voertuigagent worden gebracht. Nadat de vertraging doorheen het systeem gepropageerd is, zullen sommige niet-vertraagde voertuigen wachten op een vertraagd voertuig. Ook dat is geen probleem omdat er wordt verwacht dat voertuigagenten regelmatig exploratiemieren sturen op zoek naar eventuele betere routes. Als er een betere route beschikbaar is voor een voertuigagent, krijgt die route de voorkeur en kan de mogelijke vertraging door het wachten worden vermeden door een route in te plannen rond de vertraagde voertuigen.



Figuur 2.5: Visualisatie van vertragingspropagatie, van Lon et al.[7]. (a) Het updaten van het schema van een resource agent voor en achter een aangevraagde update van de reservaties van voertuig 2. De reservatie van voertuig 2 en alle achtervolgende reservaties zijn vertraagd met δ. (b) Vertraging propagatieproces. Resource agent 1 ontvangt een reservatievertragingnotificatie. Hierna worden recursief de vertragingen gepropageerd naar de buren van de resource agent, aangeduid met 2 en 3.

De gegokte vertraging δ moet ook geen exacte weergave van de werkelijke vertraging zijn. Bij voorkeur is δ wel representatief en zo dicht mogelijk bij de werkelijke vertraging. Voertuigagenten moeten de exacte vertragingstijd of de reden voor de vertraging niet kennen

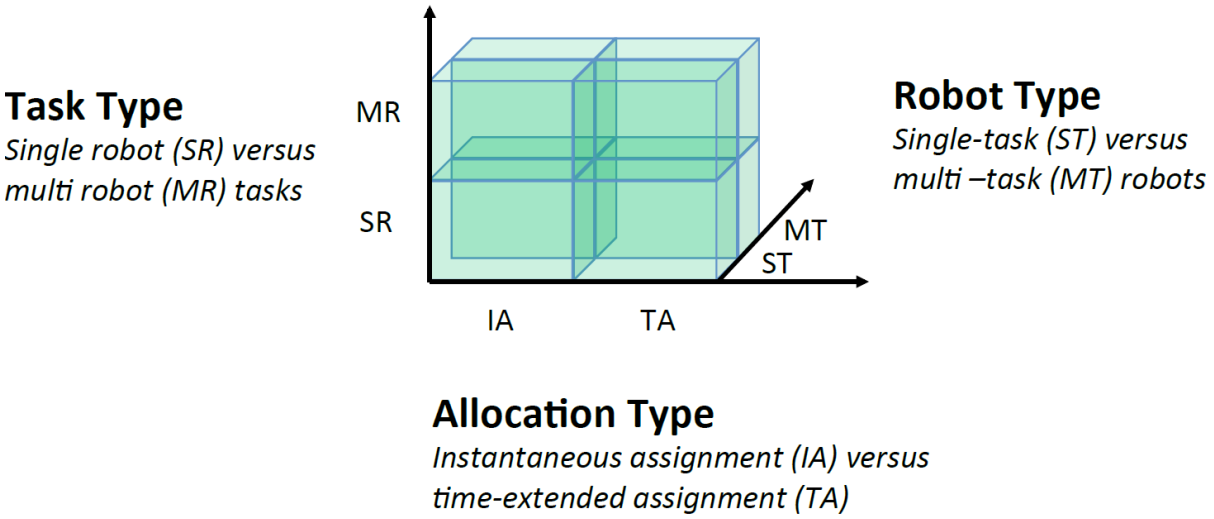
[7]. Als de vertraging kleiner is dan de verwachte δ, zal de oorspronkelijke vertraagde voertuigagent dezelfde navigatiemethode gebruiken als de wachtende voertuigagenten die zich door de vertraging navigeren. Er zullen regelmatig exploratiemieren worden rondgestuurd op zoek naar betere paden en de betere paden worden vervolgens verkozen boven de tragere. Als de vertraging langer is dan de verwachte δ dan zal de voertuigagent een nieuwe δ_2 doorsturen naar de resource agent die de voertuigagent op dat moment inneemt. De nieuwe δ_2 zal dan net zoals de oorspronkelijke δ doorheen het systeem worden gepropageerd. Als een andere voertuigagent een reservatie probeert te maken tijdens de vertragingspropagatie, zal die voertuigagent een reservatie mislukking terugkrijgen en na een wachttijd opnieuw exploratiemieren rondsturen op zoek naar een pad.

2.4 Taakallocatie overzicht van de state-of-the-art

Taakallocatie is een belangrijk aspect van veel multirobot-systemen. De eigenschappen en de complexe aard van mulitrobot-taakallocatieproblemen (MRTA) hangen af van het domein waarin het probleem zich afspeelt. In eerste instantie kan het probleemgebied worden ingedeeld in de volgende zaken. Probleemgebieden waarin alle robots van homogene aard zijn, waar alle robots dus dezelfde taken kunnen uitvoeren, net als probleemgebieden waarin de robots van heterogene aard zijn en ze niet allemaal dezelfde taken kunnen uitvoeren. Daarnaast kan er ook nog een indeling gebeuren op basis van de onafhankelijkheid van de taken en de soorten overlap in de schema's die de robots volgen. Dit deel gaat dieper in op de taakallocatie binnen de state-of-the-art. We doen hiervoor voornamelijk een beroep op het werk van Korsah et al. [24].

2.4.1 Nodige woordenschat

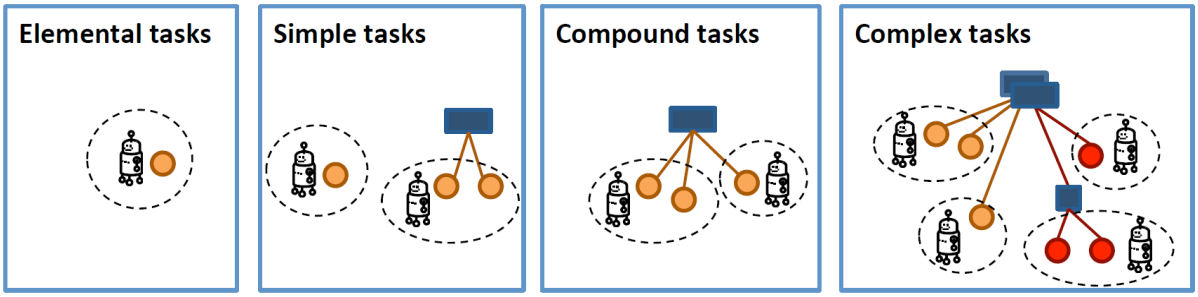
Om dieper in te gaan op de bestaande soorten taakallocaties en de state-of-the-art ervan, gaan we eerst dieper in op de nodige woordenschat en achtergrondinformatie. De taxonomie over multirobot-taakallocatie van Gerkey en Matarié [11] geven aan dat Single Task-Single Robot-Instataneous Assignment-problemen (ST-SR-IA) een onderdeel zijn van optimale toewijzingen in een combinatorische optimalisatie. Het ST-SR-IA-probleem is het enige MRTA dat kan worden opgelost in polinominale tijd. Alle andere MRTA-problemen zijn sterk NP-Hard. Gerkey en Matarié delen multirobot-taakallocaties in volgens drie assen, voorgesteld in Figuur 2.6.



Figuur 2.6: Weergave van de drie assen volgens Gerkey en Matarié [24].

De eerste as, Single-Task robots (ST) versus Multi-Task robots (MT), verdeelt de probleemruimte in robots die één enkele taak per keer kunnen uitvoeren en robots die er meerdere kunnen uitvoeren. De tweede as, Single-Robot-Tasks (SR) versus Multi-Robot-Tasks (MR), deelt de probleemruimte verder in in taken die exact één robot nodig hebben om de taak uit te voeren en taken die meer dan één robot nodig hebben. De derde as, Instantaneous Assignment (IA) versus Time-Extended assignments (TA), maakt een onderscheid in problemen die onmiddellijk toegewezen worden aan robots zonder planning voor toekomstige toewijzingen en toewijzingen die volgens een schema verlopen dat rekening houdt met de huidige taak en mogelijke latere taken.

ST-SR-TA-problemen (Single-Task robots, Single-Robot tasks, Time-Extended Assignment) kunnen worden beschouwd als het opmaken van een schema van een machine, wat overeenkomt met een schema van taken voor elke robot. Het ST-MR-IA-probleem (Single-Task robots, Multi-Robot tasks, Instantaneous Assignment) is hier aanzienlijk moeilijker om op te lossen en er wordt naar verwezen als coalitievorming. Coalitievorming wordt opgelost door taken te verdelen over meerdere niet-overlappende sets van robots. Gerkey en Matarié leggen ook uit dat een MT-SR-IA-probleem mathematisch overeenkomt met een ST-MR-IA-probleem maar met omgekeerde rollen voor de taken en de robots. Het ST-MR-TA-probleem maakt gebruik van zowel coalitievorming als tijdschema's om oplossingen te verkrijgen en is het mathematische equivalent van de minder frequente MT-SR-TA-problemen. De MT-MR-IA-problemen streven naar het bepalen van coalities waarin elke robot kan voorkomen in meerdere coalities en meerdere taken in één coalitie kan uitvoeren. Ten slotte beweren Gerkey en Matarié dat MT-MR-TA-problemen ingewikkeld zijn en dat ze kunnen worden vergeleken met scheduling problemen van een multi-processor taak en *multi-purpose* machines. Daarmee gaan Korsah et al. [24] niet akkoord. Korsah et al. halen aan dat de term *multi-purpose* niet aangeeft dat een machine meerdere taken tegelijkertijd kan uitvoeren en dat dit voorbeeld daardoor een belangrijk element weglaat.



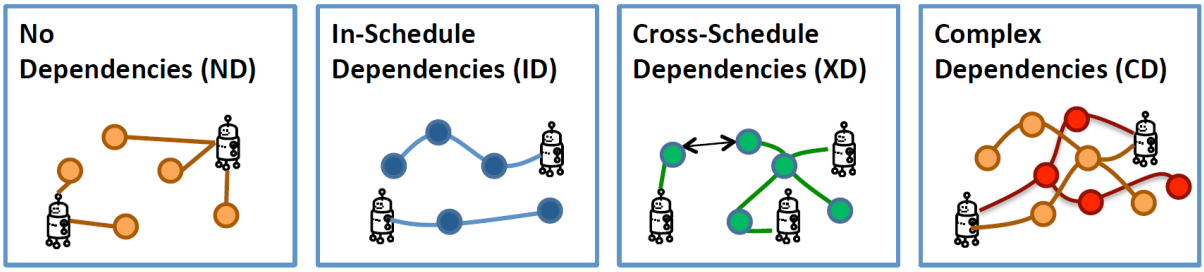
Figuur 2.7: Visualisatie van de taaktypes volgens Zlot [24]. Gestippelde cirkels tonen potentiële geldige taaktoekenningen voor robots aan. Gekleurde stippen tonen eenvoudige taken aan, gekleurde rechthoeken tonen eventuele verder verdeelbare taken aan. De verdere onderverdeling wordt voorgesteld door de boomstructuur. De overlappende bomen in de rechtse figuur staan voor de meerdere manieren om een complexe taak verder onder te verdelen.

Volgens Korsah et al. kunnen de verschillende taaktypes die door agenten worden uitgevoerd verder worden onderverdeeld. Sommige taken bestaan uit één enkele taak die niet verder kan worden opgesplitst en worden atomaire of elementaire taken genoemd. Simpele taken is een taak die ofwel een elementaire of een opsplitsbare taak is. Taken die kunnen verder worden opgebroken in meerdere stappen of deeltaken worden samengestelde taken (compound tasks) genoemd, op voorwaarde dat er een eenduidige manier bestaat om ze op te splitsen in deeltaken. Elk deel van een samengestelde taak kan aan verschillende agenten worden toegewezen. Bovendien moeten sommige onderdelen van samengestelde taken mogelijks door eenzelfde agent worden uitgevoerd. Als dat het geval is, spreken we van decomposable single tasks. Tot slot zijn er de complexe taken, taken die op meerdere mogelijke manieren kunnen worden gesplitst en verdeeld onder agenten. Bovenstaande taaktypes zijn geïllustreerd in Figuur 2.7, overgenomen uit Zlot [43]. Het grootste verschil tussen de samengestelde en complexe taken ligt bij het berekenen van een optimale splitsing in de taken. Doordat complexe taken op meerdere manieren kunnen worden gesplitst en er voor het berekenen geen optimale opsplitsing gekend is, moet er tijdens de berekening tegelijkertijd onderzocht worden hoe de taak kan worden opgesplitst en aan wie de opsplitsingen kunnen worden toegewezen. Het zoekgebied voor mogelijke taaktoewijzingen in multirobot-taaktoekenningsproblemen met simpele of samengestelde taken is exponentieel in het aantal agenten en taken. De mogelijke zoekruimte van alle mogelijke toekenningen voor hetzelfde probleem maar met complexe taken is echter exponentieel groter [43].

Restricties in taaktoekenningsproblemen zijn mogelijke willekeurige functies die de oplossingsruimte van mogelijke oplossingen van het probleem beperken. Dat zijn onder andere bekwaamheidsrestricties die definiëren welke taken een robot kan uitvoeren of capaciteit beperkingen die bepalen hoeveel taken een robot op een gegeven tijdstip kan uitvoeren. Om een samengestelde taak op te lossen, kunnen simpele taken eventueel via restricties gerelateerd zijn aan elkaar. Dat zorgt ervoor dat de taken, hoewel toegewezen aan verschillende robots, op hetzelfde moment of in een bepaalde volgorde worden uitgevoerd.

Dat betekent dat er een hechte relatie is tussen het splitsen van een complexe taak en de restricties tussen die splitsing. Een probleem dat kan worden voorgesteld als onafhankelijke samengestelde taken die met ongerelateerde restricties kunnen worden uitgevoerd, kan worden gelijkgesteld aan een probleem met simpele taken die verbonden zijn door inter-taak restricties. Dat houdt in dat sommige problemen zich op meerdere manieren kunnen voordoen, en dus ook opgelost worden. In optimalisatieproblemen, zoals taaktoekenning, wordt een bepaalde utilityfunctie geoptimaliseerd. Gerkey en Matarié defineëren de utilityfunctie voor de robot r en de taak t als $-\infty$ als robot r de taak t niet kan uitvoeren en anders over de gestandaardiseerde schaal $Q_{rt} - C_{rt}$ met Q_{rt} een beloning voor de robot en C_{rt} de kost om de taak uit te voeren. In sommige problemen is de utility van een agent over een taak onafhankelijk van de utility van een agent over een andere taak. Utilities kunnen worden beschouwd als functies met corresponderende waarden ten opzichte van de kenmerken van het probleem en restricties als binaire functies die de relevantie van de kenmerken van het probleem aanduiden. Beide hebben een impact op het niveau van onafhankelijkheid tussen agenten en de taken in een bepaald probleem.

iTax, de taxonomie voorgesteld door Korsah et al. [24], houdt rekening met de verschillende niveaus van onafhankelijkheid die voortkomen uit de utility en de restricties, net als de impact van de niveaus op het toewijzen van taken aan agenten. De taxonomie bestaat uit twee niveaus, het eerste is gebaseerd op de onafhankelijkheidsgraad van de agent-taak utility's. Het tweede niveau geeft verdere informatie over de configuratie van het probleem, door middel van de taxonomie van Gerkey en Matarié [11]. De onafhankelijkheidsgraad wordt opgesplitst in vier mogelijke categorieën. Elk wordt in de volgende paragrafen besproken en is afgebeeld in Figuur 2.8. De wiskundige modellen kunnen worden teruggevonden in [24].



Figuur 2.8: De vier high-level categorieën van de nieuwe taxonomie door Korsah et al. [24]. Gekleurde cirkels stellen taken voor en de lijnen ertussen stellen de routes van de agenten voor. De pijlen tussen de taken beelden restricties tussen die taken af en de overlappende routes in de rechter figuur staan voor de mogelijke taaksplitsingen.

2.4.2 No Dependencies

No Depedencies (ND) [24] taakallocatieproblemen zijn taakallocatieproblemen met simpele of samengestelde taken die onafhankelijke robot-taak-utility's hebben. Dat houdt in dat de uiteindelijke utility voor een robot of taak volledig onafhankelijk is van andere taken en robots in het systeem. De restricties in dit probleem gaan voornamelijk over één enkele robot, één enkele taak of één enkel robot-taak paar. Een veelvoorkomend voorbeeld is de

situatie waarbij enkel rekening wordt gehouden met de afstand tot een taak ten opzichte van de robot. Alle problemen in die categorie zijn per definitie ST- en SR-problemen. Als een robot namelijk meerdere taken kan uitvoeren, zullen er extra restricties toegevoegd worden die afhankelijk zijn van andere robots door de beperkte capaciteiten van de robot, die enkel ST-problemen toelaat. Daarnaast, als een taak meerdere robots nodig heeft om de taak uit te voeren, zullen er extra afhankelijkheden tussen deze robots aanwezig zijn, wat per definitie MR-problemen uitsluit.

ND ST-SR-IA

De subcategorie ND ST-SR-IA draait voornamelijk rond het toewijzen van één-op-één relaties tussen singlerobot-taken aan onafhankelijke singletaak-robots. Het gaat dan vaak om lineaire toewijzingsproblemen die kunnen worden opgelost in polynomiale tijd met behulp van algoritmen zoals, maar niet beperkt tot, het Hungarian algoritme of varianten zoals een gedistribueerde versie voorgesteld door Chopra et al. [4] en veilingmethodes, zoals voorgesteld door Gerkey en Matarié [11]. Mogelijke oplossingen worden bereikt door toewijzingen tussen robots en taken, met mogelijke inmenging van dummy robots als er te veel taken zijn ten opzichte van robots. Verder kunnen de utilityfuncties gedefinieerd worden met robot-taakrestricties door grote negatieve utility's te gebruiken.

ND ST-SR-TA

In de tijd verlengde toewijzingen versie van het ND ST-SR-IA-probleem kan elke robot meer dan één enkele taak toegewezen krijgen. Daarnaast wordt ook een schema gebouwd voor elke robot dat aangeeft wanneer welke taak wordt uitgevoerd. Dat treedt op als er meer taken dan robots zijn of meer optimale oplossingen bekomen worden waar een robot na het uitvoeren van een taak meteen een andere kan uitvoeren en zo de kosten van een andere robot beperkt. Door het tekort aan in-shedule dependencies maakt de volgorde van het uitvoeren van de taken geen verschil op de utility- of doelfunctie. Door het gebrek aan in-shedule dependencies kan een robot ook meerdere taken tegelijkertijd uitvoeren en nog steeds tot dit deelprobleem horen.

2.4.3 In-Schedule Dependencies

De utility van een robot voor een taak hangt in deze probleembranche af van welke andere taken ook aan de robot zijn toegewezen. Dergelijke problemen komen vooral voor in de tijd verlengde toewijzingen taakallocatieproblemen waarin de utilityfunctie afhangt van bijvoorbeeld routeringskosten of de tijdstippen waarop een taak wordt afgerond, PDP zijn daarvan een passend voorbeeld. In-Schedule Dependencies (ID) [24] komen ook voor in gevallen waar een robot meerdere taken tegelijkertijd kan afronden maar de tijdstippen waarop elke taak afgerond wordt van belang is. De ID-klasse kan per definitie geen subklassen bevatten van het ST-SR-IA door het gebrek aan In-Schedule Dependecies die de ST-SR-IA subklasse definieëren. Daarnaast hebben alle problemen die multirobot-taken bevatten per definitie ook Cross-Schedule Dependencies naast de In-Schedule Dependencies en vallen ze dus buiten de ID-klasse.

ST-SR-TA

Een ST-SR-TA-probleem is doorgaans een No Dependencies-probleem, zoals in subsectie 2.4.2, tenzij er een tijdslimiet wordt opgelegd waarbinnen alle taken moeten worden uitgevoerd of een taak inter-taak-restricties heeft waardoor er eerst andere taken moeten worden toegewezen of uitgevoerd. De taken die een robot kan uitvoeren, hangen dus af van alle andere toewijzingen die een robot al ontvangen heeft, gekoppeld met extra restricties. Veel methodes om dit soort problemen op te lossen, gebruiken gecentraliseerde methodes die ontworpen waren voor Traveling Salesman- (TSP) en Multi-Traveling Salesman (m-TSP)-problemen. Daarnaast worden veiling- of markt-gebaseerde-ontwerpen veel gebruikt voor het oplossen van multirobot-taken door de gedistribueerde natuur van robotteams. De veiling-gebaseerde-ontwerpen bereiken multirobot-coördinatie door het veilen van taken aan robots, die op de taken bieden om de taaktoekenning te bepalen. Daardoor wordt ook het berekenen van in de tijd verlengde toewijzingen van taken, doorgegeven aan de robots zelf. De robots bieden op de taken als ze vrijkomen terwijl ze rekening houden met hun eigen schema's. Robots kunnen ook hun eigen taken proberen te veilen, op voorwaarde dat ze nog niet uitgevoerd worden, om mogelijke lokale minima te vermijden. Een voorbeeld van een dynamische veilingmethode is DynCNET, voorgesteld door Weyns et al. [38]. Deze veiling-gebaseerde-methodes worden intensief onderzocht en er zijn meerdere variaties mogelijk.

MT-SR-IA

De subcategorie ID MT-SR-IA stelt problemen voor waar een onmiddellijke toekenning van een set van taken aan een robot gebeurt, waarna die taken tegelijkertijd uitgevoerd moeten worden. Elke taak heeft een enkele robot nodig maar een robot kan meerdere taken tegelijk uitvoeren. Dat kan worden voorgesteld door een ND-probleem waar de tijdsrestricties worden vervangen door capaciteitsrestricties. De capaciteitsrestricties illustreren het feit dat geen enkele realistische robot een oneindig aantal taken tegelijkertijd kan uitvoeren. Van die subcategorie is er geen duidelijk voorbeeld maar ze wordt toch vermeld voor later gebruik met betrekking tot multirobot-taken in de context van coalitievorming.

MT-SR-TA

In de subklasse ID MT-SR-IA moet een in de tijd verlengde toewijzing van singlerobot-taken gebeuren aan multitaak-robots. Sommige deelproblemen van VRP, zoals DARP en PDP, vallen onder deze categorie. Dit gaat meestal over voertuigen die meerdere pakketten kunnen vervoeren die afkomstig zijn van verschillende pickup- en leveringslocaties, dus van een vaste start- en stoplocatie van de robot (depots) met daartussen variabele taken. Toen Korsah et al. iTax voorstelden, bestond er nog weinig onderzoek over dit onderwerp.

2.4.4 Cross-Schedule Dependencies

Problemen in de categorie Cross-Schedule (XD) [24] gaan voornamelijk over het toewijzen van simpele of complexe taken in situaties waar de utilityfunctie van een robot afhangt van zijn geselecteerde taken maar ook van de taken die andere robots uitvoeren. Cross-Schedules Dependencies komen vooral voor in twee situaties. In de eerste situatie worden

twee of meer singlerobot-taken toegewezen aan verschillende robots. De taken zijn sterk gerelateerd aan intertaak-restricties, zoals bv. de volgorde van taken, het tegelijkertijd uitvoeren van taken of de afstand tussen elkaar. In het tweede geval worden multirobot-taken toegewezen aan een subset van robots door het aanmaken van coalities. Daardoor ontstaat er een coalitievormingsprobleem. Het grootste verschil tussen ID en XD ligt vooral in de manier waarop robots hun schema individueel kunnen optimaliseren of waar er groepscommunicatie nodig is voor het optimaliseren.

SR-taken

Het simpelste probleem in XD is waar er onmiddellijke toewijzingen plaatsvinden van singlerobot-taken, met mogelijke intertaak-restricties, aan singletaak-robots. Ten tweede zijn er ook Cross-Schedule Dependencies die voorkomen door intertaak-restricties waar de tijd verlengde toewijzingen gebeuren van taken aan robots. Er zijn meerdere multirobot-taakallocatiemethodes die XD ondersteunen. Het M+ systeem voorgesteld door Botelho en Alami [3], gebruikt vooral een marktsysteem met onmiddellijke toewijzingen. Daarnaast laat het ook prioriteitsrestricties toe door onderhandelingen op uitvoerbare taken. Er bestaan ook nog enkele andere methodes op basis van veilingen. Bovendien worden er ook methodes voorgesteld die werken met een gecentraliseerde planner die gebruikmaakt van heuristische algoritmen om conflictvrije schema's op te stellen. Deze methode gebruikt gecentraliseerde doelallocatie, gevolgd door een gedecentraliseerde gedetailleerde planning en een schema voor elke robot. De gedecentraliseerde planning kan ook vervangen worden door methodes gebaseerd op veilingen en individuele biedingen van de robots. Lemaire et al. [26] stellen simpele orderrestricties voor tussen taken in de vorm van 'taak x moet a seconden plaatsvinden voor taak y'. Om de orderrestricties uit te voeren, wordt de eerste taak toegewezen aan een masterrobot die de volgende taken veilt aan andere robots. De robot die een van de volgende taken toegewezen krijgt, wordt een slave genoemd. De master-slaverelatie blijft bestaan tot alle gerelateerde taken uitgevoerd zijn.

MR-taken

Problemen met multirobot-taken met XD worden meestal opgelost door coalitievorming. Deze subcategorie, gericht op multirobot-taken, lost dezelfde problemen op zoals in de vorige subcategorie van XD maar dan met multirobot-taken. Coalitievorming is een veel onderzocht onderwerp in taakallocatie en veel mogelijke manieren om de coalities te vormen zijn al voorgesteld. Sommige maar niet allemaal zijn hieronder kort opgesomd:

- Greedy, gedistribueerde-set-opdelingsalgoritmen die gebruikmaken van vectoren die de nodige capaciteiten voorstellen. Elke coalitie heeft een dergelijke vector. De coalitiewaarden voor een taak worden berekend via de utilityfunctie die de coalitie kan bereiken door het uitvoeren van de taak. De eerste fase van het algoritme bestaat uit een gedistribueerde berekening van de coalitiewaarden, gevolgd door een fase waarin iteratief beslist wordt welke coalities voorgenomen worden voor bepaalde taken. Dit werd voorgesteld door Shehory en Kraus in [33].

- Guerrero en Oliver stellen in hun werk [13] een coalitievormingsprobleem voor dat met veiling-achtige mechanismen van een robot een leider van een taak maakt. De leider veilt de taak vervolgens naar andere robots. De veiling vormt de coalitie om

de taak uit te voeren. Andere algoritmen bouwen verder op deze veiling-achtige manier om coalities te vormen door het toevoegen van combinatorische biedingen om samen een coalitie te vormen.

- Shiroma en Campos stellen in [34] CoMutaR voor. CoMutaR is een framework voor taakallocaties met gedeelde middelen. Elke taak is een serie van acties die moeten worden uitgevoerd. Om de coalitie te bouwen, wordt een serie van vragen naar de robots gestuurd met informatie over de acties. Op basis van die vragen wordt een veilingsysteem opgezet waarna de nodige robots geselecteerd worden om in de coalitie deel te nemen.

- Koes et al. stellen een in de tijd verlengde toewijzingen van taken coalitievormingsprobleem als een mixed interger linear programming formulatie voor in [23]. Naast die formulatie ontwierpen ze ook COCoA (Constraint Optimization Coordination Architecture) om een iteratieve combinatie te maken van een linear programming solver en een heuristische methode die de beginwaarden van de solver aanpast.

2.4.5 Complex Dependencies

De subcategorie Complex Denpendencies (CD) [24] spitst zich vooral toe op complexe taken in domeinen waar de utility van een robot voor een taak afhangt van de schema's van de andere robots. Complexe taken hebben meerdere manieren om te worden opgesplitst en daardoor tot minstens één robot te worden toegewezen met meerdere opties om de taken efficiënt op te delen. De vraag bij deze complexe taken is dus om zowel een efficiënte opdeling te maken voor de taken en ze dan aan een subset van robots toe te wijzen die deze efficiënte opdeling kan uitvoeren. Complexe taken bestaan uit een set van simpele taken die kunnen worden gecombineerd tot een complexe taak doordat er restricties aanwezig zijn op die simpele taken. CD hebben twee grote probleemdomeinen. In het eerste domein draait het voornamelijk rond singlerobot-taken (SR) die door restricties of disjuncties verbonden worden tot complexe taken. In het tweede domein bevinden zich problemen met multirobot-taken die complexe taken zijn.

SR-taken

Jones et al. stelden in [20] twee methodes voor om een set van ST-SR-TA problemen op te lossen. De eerste gebruikt gelaagde veilingen gecombineerd met clusters en opportunistische padbepaling om een begrensde zoektocht uit te voeren over de mogelijke in tijd verlengde schema's en toekenningen. De tweede methode gebruikt genetische algoritmen.

MR-taken

Parker en Tang stelden een methode voor om coalities te vormen door een proces dat zij beschreven als *"automated task solution synthesis"* in [28]. Dat proces wordt bereikt doordat de oplossing gebouwd wordt door dynamische verbindingen tussen een netwerk van schema's die op de robots zelf aanwezig zijn. Die schema's zijn gedefinieerd door input- en exportpoorten, lokale variabelen en het gedrag van de robots. Gegeven de informatie, worden die schema's vervolgens via de poorten verbonden om het gewenste gedrag te bereiken. In de gedistribueerde versie van dit proces beslist elke robot zelf

welke informatie hij nodig heeft om beslissingen te nemen en vraagt hij die informatie zelf op aan andere robots. Dit proces werd voorgesteld voor het oplossen van SR-MR-IA-problemen. Daarnaast werd er opnieuw een veilingmethode voorgesteld om te bieden op taakbomen in plaats van op simpele taken. Een taakboom stelt een mogelijke splitsing van een complexe taak voor en wanneer de taakboom geveild wordt, kan een robot zowel bieden op de splitsing die de veilingmeester heeft gemaakt of op een splitsing die de robot zelf maakt. Er kan ook geboden worden op een deel van de boom in plaats van op de volledige boom zelf. Daarna beslist de veilingmeester over de best mogelijke splitsing van de complexe taak, gebaseerd op alle mogelijke veilingen om een zo laag mogelijke kost te bereiken om de taak op te lossen.

2.4.6 Taaktoekenning in MAS

Multirobot-taakallocatie in MAS telt twee mogelijke manieren voor de taaktoekenning, gecentraliseerd en gedecentraliseerd [22]. Sommige oplossingen gebruiken een hybride van de twee waarin bijvoorbeeld elke robot gedecentraliseerd biedt op een veilingitem en een centraal algoritme die de best mogelijke taakverdelingen uit de veiling bepaalt. Het voordeel van gecentraliseerde systemen is onder andere het vermijden van dubbel werk. Dat heeft ook meerdere nadelen; zo worden de robuustheid van het systeem aangetast en wordt de schaalbaarheid op grote netwerken moeilijk. Gedecentraliseerde algoritmen zijn één van de meest gerapporteerde oplossingen voor multirobot-taakallocatieproblemen. Daarnaast is de dynamische aard van veel multirobot-taakallocatiesproblemen makkelijker op te lossen met gedistribueerde systemen. Daarbij wordt het werk gesplist over meerdere robots en valt het knelpunt in de communicatie naar de centrale eenheid weg. Informatie-uitwisseling tussen meerdere robots is nog steeds nodig om tot een goede oplossing te komen. De vraag wordt dan hoe efficiënt informatie kan gedeeld worden op een lokaal niveau. De meestvoorkomende methodes om multi-robot taakallocaties uit te voeren zijn de volgende. Als eerste zijn er de op markt/veiling gebaseerde methodes waarbij een taak geveild wordt. De robots bieden elk op een bepaalde taak waarna een centrale eenheid mogelijks nog verdere beslissingen neemt. Ten tweede zijn er de optimalisatiemethodes waarbij een functie gemaximaliseerd of geminimaliseerd wordt. Die functie hangt af van mogelijke restrictie en utilityfuncties. Beide methodes komen zowel voor in gecentraliseerde, gedecentraliseerde en hybride vormen met verdere methodes en technieken, zoals neurale netwerken [41], mierenkolonie-optimalisaties [42] en genetische algoritmes [21] voor het optimaliseren van de bekomen oplossingen. Rizk et al. [31] tonen aan dat het gelijktijdige vormen van coalities en toewijzen van taken tot meer optimale oplossingen kan leiden. Hoewel die werkwijze veelbelovende resultaten had, zijn er volgens Rizk et al. weinig werken die dit verder onderzocht hebben.

2.5 Deadlocks in MAS

In veel van de state-of-the-art-methodes, besproken in sectie 2.4, is de kans op spontane deadlocks klein. In delegate MAS wordt de kans op het vormen van spontane deadlocks een stuk groter doordat de beslissingskracht in de robots zelf zit. In specifieke scenario's zijn deadlocks bijna onvermijdelijk. Een voorbeeld daarvan is terug te vinden in de parkeerlanes voorgesteld in hoofdstuk 3. Dergelijke scenario's bevatten vaak bidirectionele

segmenten [25].

2.5.1 Deadlocks

Deadlocks komen voor in processen waar meerdere spelers dezelfde middelen willen gebruiken terwijl die middelen beperkt zijn in toegang. Deadlocks komen voor als aan de volgende voorwaarden wordt voldaan [25].

- Mutual exclusion: een middel kan maar door één proces opgeroepen worden, vaak ter bescherming van middelen die niet-verdeelde operaties aankunnen.
- Hold and wait: een proces mag de toegewezen middelen bijhouden terwijl het wacht op de nodige volgende middelen.
- No pre-emption: een proces kan niet geforceerd worden om zijn toegelaten middelen vrij te geven.
- Circular wait: een gesloten ketting van processen waar ieder proces zich op zijn minst aan één middel vasthoudt.

In de MAS-situatie waar elk proces een fysieke AGV is en een deel van de middelen een fysieke plaats is waarop die AGV zich bevindt, is het makkelijk in te zien dat een AGV zijn middelen niet zomaar kan vrijgeven en dat elke AGV zich aan enkele middelen vasthoudt. Als meerdere AGV's zich vervolgens willen verplaatsen naar middelen die al ingenomen zijn door een andere AGV en die AGV's zich in een cirkelvormige beweging willen verplaatsen, is een deadlock onvermijdelijk. Deadlocks vermijden en oplossen is een kernprobleem voor sommige scenario's. Het oplossen van deadlocks in gedistribueerde MAS is bovendien vrij ingewikkeld. In [25] wordt een manier geïntroduceerd om deadlocks op te lossen op een weg waar slechts plaats is voor één voertuig maar beide rijrichtingen toegestaan zijn. Er wordt dan één AGV geselecteerd die onderdeel is van de mogelijke deadlock en die beslist wie en wanneer het beperkte pad kan kruisen. Door de implementatie van de parkeerlanes wordt het oplossen van de deadlocks via een manier waar de AGV's onderling onderhandelen een stuk moeilijker. Als vijf parkeerlanes naast elkaar in een deadlockprobleem geraken, moeten alle AGV's die deze vijf parkeerlanes willen inrijden met elkaar onderhandelen. Dit kan snel tot grote onderhandelingen uitlopen.

2.5.2 Edge-chasing algoritme

In het edge-chasing algoritme wordt een graaf bijgehouden of gemaakt als er zich mogelijks een deadlockprobleem voordoet. Constante monitoring van een grootschalig systeem is namelijk kostelijk. Door een bericht te sturen over de eventueel geblokkeerde processen, worden de gevraagde middelen bekeken. Via die middelen wordt het proces dat op dat moment met die specifieke middelen bezig is, opgezocht. Dat proces stuurt dan nogmaals een bericht naar zijn middelen waarop het wacht voor het zijn handeling kan afronden. Als er tijdens het uitvoeren van het edge-chasing algoritme een cirkelvormige wachtgraaf wordt gevonden, is er een mogelijke deadlock gevonden. Het oplossen van die deadlocks kan gebeuren door enkele processen af te sluiten, vaak gebaseerd op prioriteiten of door het systeem terug te draaien tot op een punt waar geen deadlocks aanwezig waren. Dat

laat het systeem dan toe om uit de deadlocksituatie te geraken.

In robotgebaseerde MAS is het oplossen van deadlocks een stuk moeilijker. Een proces kan namelijk niet zomaar gestopt worden en de middelen vrijgegeven omdat een AGV te allen tijde een locatie inneemt [25]. Het edge-chasing algoritme kan wel geïmplementeerd worden in robot-MAS door de robots gelijk te stellen aan de processen.

2.6 RinSim Simulator

RinSim is een simulator ontwikkeld in de Imec-Distrinet-onderzoeksgroep van KU Leuven, door van Lon en Holvoet [37]. RinSim is specifiek ontwikkeld voor onderzoek rond het General PDP-probleem met tijdslots in academische kringen. Zoals besproken in subsectie 2.1.3 kan een PDPTW-probleem worden omgezet in een Dial-a-Ride-probleem met deadlines. RinSim is gericht op het testen van logistieke problemen met behulp van Multi-agent systemen. RinSim laat toe om GPDP-problemen te configureren en verschillende oplossingsmethodes te testen en via statistieken te vergelijken. Het belangrijkste kenmerk van RinSim is het onderscheid tussen het probleemgebied en de oplossing. De belangrijkste elementen uit de scenario's worden gemodelleerd als agenten. De standaardmodellen, zoals besproken in van Lon en Holvoet [35], kunnen worden uitgebreid gebaseerd op de noden van de te testen oplossingsmethodes. De volgende modellen worden gebruikt of uitgebreid voor het configureren van de casestudy-omgeving, beschreven in hoofdstuk 3.

- **TimeModel**
  Het **TimeModel** vormt de basis van RinSim. Dit model laat de simulator toe om op basis van een ticklength de verschillende elementen die op dit model geabonneerd zijn een actie te laten uitvoeren. Met andere woorden is het TimeModel verantwoordelijk voor de verstreken tijd in de simulator. Het TimeModel kan ook gebruikmaken van een realtimeclock om de omgeving in echte tijd te simuleren.

- **RoadModel**
  Een **RoadModel** is een interface die verschillende voorstellingen van de omgeving voorstelt in bijvoorbeeld een graaf of een vlak. In een vlak kunnen de agenten zich verplaatsen op het volledige vlak in rechte lijnen, terwijl een graafvoorstelling zich meer op een traditioneel stratennetwerk gedraagt. Alle RoadModels hebben ook maximale snelheden die moeten worden gerespecteerd door de **MovingRoadUser** agenten.

- **PDP-Model**
  De **PDP-Model** bevat de implementaties van de Pickup-and-Delivery-Problemen met tijdslots en verzorgt de ophaal- en afleveroperaties van de agenten, gekoppeld met de tijdsduur en capaciteiten van de agenten.

- **ComModel**
  Het **ComModel** laat communicatie toe tussen verschillende agenten die dit model implementeren. Om een realistische simulatie te maken, kunnen klasses die van dit model gebruikmaken een maximumafstand voor communicatie instellen. Die afstand is consistent met de afstanden in het RoadModel.

- **StatsTracker**
  De **statsTracker**module is het centrale informatieverzamelingspunt van RinSim. De verkregen informatie van RinSim kan door het uitbreiden van de module verwerkt en opgeslagen worden.

Daarnaast wordt er gebruikgemaakt van het GUI-model van RinSim om een voorstelling van het probleem te maken. De voorgestelde oplossing, zoals voorgesteld in hoofdstuk 4, wordt vrij ontwikkeld naast de standaardmodellen die door RinSim worden geleverd.

Hoofdstuk 3

Casestudy: AGV's in de haven

In deze masterproef bestuderen we DMAS in de context van een grootschalige autoterminal in een haven. Die terminal is onderdeel van meerdere homogene terminals die samen het volledige import-exportgedeelte behandelen van een bedrijf in België. De soorten taken en problemen zijn in alle terminals gelijkaardig, maar de terminals hebben allemaal een andere plattegrond die onderhevig is aan veranderingen. Op de terminals staan duizenden wagens geparkeerd op verschillende soorten opslaglocaties. Die wagens moeten worden verplaatst naar workshops, laad-losplaatsen of er kunnen interne verplaatsingen plaatsvinden naar andere opslaglocaties. Een terminal wordt in detail bekeken en gemodelleerd als onderdeel van het experiment, zoals beschreven in sectie 5.1. Als de voorgestelde oplossing in hoofdstuk 4 gunstig is in deze terminal, kunnen we ook besluiten dat de oplossing gunstig zal zijn in de andere terminals. Verplaatsingen van wagens vinden op dit ogenblik enkel plaats tussen 8h en 18h, met pieken rond de vroege namiddag.

3.1 YardSolver

De YardSolver is een onderdeel van het softwaremanagementsysteem dat al aanwezig is op de site. De YardSolver is gefocust op de berekeningen van welke wagens op welk tijdstip moeten worden verplaatst. In essentie zijn dat restrictie problemen. De gevonden oplossing wordt uitgeprint en aan de dockmedewerkers doorgegeven. Daardoor kunnen nieuwe opdrachten die doorheen de dag binnenkomen moeilijk onmiddellijk uitgevoerd worden. De taken worden in slechts twee groepen doorgegeven, een groep in de ochtend en een groep in de avond met de prioritaire taken die doorheen de dag zijn toegekomen. De YardSolver rekent op accurate data van de site maar door menselijke fouten (zoals vergeten scannen of verkeerde locaties) komt de data in de YardSolver niet noodzakelijk overeen met de werkelijke situatie.
Zoals aangetoond in 2.1.3 kunnen de PDPTW door middel van een restrictie processor omgevormd worden naar DARP met deadlines waardoor er ook minder berekeningen nodig zijn op de AGV's. Door het veranderen naar een automatisch systeem kunnen dynamische taken met prioriteiten ogenblikkelijk doorgegeven worden naar de omgeving van de site, wat het onmiddellijk ophalen door de AGV's toelaat. Het automatiseren voorkomt ook menselijke fouten. In combinatie met het makkelijker uitdelen van taken naar de omgeving kan de YardSolver meer accurate, snellere en meer beslissingen nemen, wat de efficiëntie en doorvoer ook ten goede zou moeten komen.

3.2 Op te lossen problemen

Er zijn verschillende DARP die zich kunnen voordoen in deze case. Hieronder worden de belangrijkste opgesomd, samen met de moeilijkheden die zich kunnen voordoen bij het oplossen van het probleem.

- **Normale parkingplaatsen**
  Dit zijn normale parkeerplaatsen, zoals langs de kant van de weg. Van hieruit moeten de wagens enkel opgepikt en vervoerd worden naar een nieuwe locatie die aangegeven werd door de YardSolver. Deze kunnen ook worden vergeleken met visgratenpatronen als er maar één wagen per parkeerplaats kan worden geplaatst.

- **Visgratenpatroon**
  Dit zijn parkeerplaatsen waarin tot vier wagens geplaatst kunnen worden. Ze dienen als voorbereiding van een vrachtwagenlading en daarna moeten de wagens niet meer verplaatst worden door de AGV's. We gaan ervan uit dat de wagens in het visgratenpatroon in de correcte volgorde aankomen in de parkeerplaatsen door de YardSolver die de opdrachten in de correcte volgorde vrijstelde.

- **Parkeerlanes**
  Parkeerlanes zijn lange rijen waar tientallen wagens achter elkaar geparkeerd staan. Als een wagen moet worden weggehaald voor een levering, moeten eerst alle wagens die zich voor de betreffende wagen bevinden uit de lane worden opgepikt en na de verplaatsing van de doelwagen worden teruggeplaatst. Parkeerlanes kunnen in sommige gevallen langs beide kanten benaderd worden. Daardoor kan een kleinere hoeveelheid wagens verplaatst worden als de wagen benaderd wordt langs de kant waar zich de minste wagens bevinden voor de doelwagen.

- **Batch jobs**
  Batch jobs vinden plaats na het lossen van schepen. In dat geval moet een grote hoeveelheid wagens zo snel mogelijk verplaatst worden van de kade naar de parkeerplaatsen. De kades bestaan uit meerdere parkeerlanes die kunnen geledigd worden in de volgorde waarin ze staan.

- **Onderbreking wegen**
  Er kunnen ook onderbrekingen plaatsvinden op de interne wegen. Dat kunnen voetgangers zijn die oversteken, wagens die rondrijden, ... Die worden allemaal gemodelleerd als onderbrekingen die de voortgang van de AGV's tijdelijk stoppen.

- **Over publieke wegen**
  Tussen de verschillende terminals liggen ook publieke wegen. Soms moet een wagen over een publieke weg vervoerd worden naar een andere terminal om daar te worden behandeld. Publieke wegen zijn een stuk moeilijker voor automatische wagens of AGV's door obstakels die zich niet voordoen op de terminal zelf. Dit valt echter buiten het bestek van deze masterproef.

3.3 Requirements

In dit onderdeel worden de vereisten voor de bovenstaande casestudy globaal overlopen. Er word voornamelijk naar de niet-functionele vereisten gekeken. Er wordt geen aandacht besteed aan de specifieke vereisten die over de DMAS oplossing gaan en ook niet aan de vereisten voor de AGV's zelf of aan de infrastructuur agenten die eigen zijn aan DMAS.

3.3.1 Functionele vereisten

De functionele vereisten moeten vooral de mogelijke problemen oplossen die eerder in de casestudy werden aangehaald. Om die problemen op te lossen wordt er gebruikgemaakt van DMAS, zoals besproken in sectie 2.3. De AGV's moeten een taak kunnen vinden op het netwerk, de beste taak selecteren, zich ernaar verplaatsen en zich naar de eindlocatie van de taak begeven om de wagen af te leveren. De beste taak hangt af van meerdere factoren, voornamelijk de prioriteit van de taak en of de AGV na de taak een oplaadstation kan bereiken om op te laden, of op zijn minst nog genoeg batterijniveau heeft om na de taak nog een aantal honderden meters te rijden. Na het filteren zal elke AGV zo weinig mogelijk tijd willen doorbrengen zonder vracht en zal hij dus de dichtste taak die nog vrij is, oppikken. De AGV's volgen een BDI-systeem, besproken in subsectie 2.2.3, voor hun keuzes. Hoe onderling sequentiële afhankelijke taken opgelost worden, is uitgelegd in hoofdstuk 4. Daarnaast moet een AGV ook een pad kunnen bepalen naar een oplaadstation, een AGV mag immers nooit een lege batterij hebben. Als de AGV na het bepalen van een taak of pad een betere oplossing vindt, dan zal hij die betere oplossing proberen te volgen. De AGV's mogen ook andere voertuigen die zich op de weg bevinden voor de veiligheid nooit inhalen. Een honderdtal AGV's moeten zich tegelijkertijd veilig en zo snel mogelijk over de terminal kunnen bewegen waar zich ook andere wagens en mensen op begeven. Als er een probleem optreedt bij een AGV, mag de getroffen AGV geen andere AGV's tot stilstand brengen en moeten er zo weinig mogelijk storingen plaatsvinden op de terminal. Afgaande op het vorige kan de volgende onderzoeksvraag worden opgesteld:

Kunnen onderling sequentiële afhankelijke taken efficiënt in een DMAS-omgeving geïmplementeerd worden?

Met als deelvraag:

Wat is het effect van de oplossingsmethode op de AGV's en hun coördinatie?

3.3.2 Niet-functionele vereisten

De belangrijkste niet-functionele vereisten focussen vooral op snelheid, robuustheid en een zo groot mogelijke doorvoer van wagens. Een grote doorvoer van wagens zorgt namelijk voor een competitieve voorsprong op de concurrenten. Een AGV die een probleem ondervindt terug werkende krijgen, zorgt voor kostbaar verlies van efficiëntie, zeker als dat gebeurt tijdens het vervoer van een wagen. Niet alle wagens bereiken het einde van de route zonder schade, wat kostelijk is en zoveel mogelijk moet worden vermeden. Er mag zeker geen schade of hinder zijn naar de omlopende dockmedewerkers die met honderden aanwezig zijn op de site. Door de mogelijkheden om taken doorheen de dag uit te

delen, wordt ook de dienstverlening naar de klanten verbeterd omdat een dringend order efficiënter en sneller uitgevoerd kan worden. Het wegvallen van menselijke fouten zorgt ook voor een beter zicht op de werkelijke situatie en een vermindering van schade aan de wagens. Een AGV kan ook 's nachts werken zonder bijkomende kosten. Het zou een interessante piste zijn om AGV's dag en nacht te laten werken en zo de doorvoer nog verder op te drijven.
De focus in deze masterproef ligt vooral op het onderzoek naar het toewijzen van taken en routes aan AGV's, gecombineerd met het tijdig opladen ervan. Er wordt vooral gefocust op de snelheid en doorvoer van de DMAS-oplossing. Daarnaast wordt ook onderzocht wat de invloed is van de onderling afhankelijk taken op de AGV's en de invloed van een AGV die zijn beslissing herziet. We kunnen geen rekening houden met menselijke interactie, mechanische problemen of schade die de AGV's kunnen oplopen. Het ontwerpen van de AGV's en het uittesten van de voorgestelde oplossing op de site vallen buiten de bestek van deze masterproef.

Hoofdstuk 4

Implementatie van taak allocatie met DMAS

4.1 Voorafgaande opmerkingen

In dit hoofdstuk wordt de voorgestelde implementatie van de sequentiële taakallocatie besproken. Die wordt deel per deel opgebouwd en begint bij de implementatie van de feromonen en mieren die kenmerkend zijn voor een DMAS-oplossing gebaseerd op mieren. Verder worden de taakallocatie en de beslissingslogica gebaseerd op een BDI-systeem van de AGV's besproken. Zoals besproken in hoofdstuk 3, worden niet alle vereisten van de casestudy bekeken. Sommige onderdelen, zoals het vervoeren van de auto's over de publieke wegen en mogelijke onderbrekingen van het besliste pad van de AGV's, vallen buiten deze masterproef. Het verplaatsen van de AGV's over publieke wegen is namelijk een stuk ingewikkelder. Ook de hinder die de AGV's kunnen oplopen tijdens het uitvoeren van hun taken werd niet bekeken. Uit eerdere onderzoeken, zoals dat van Holvoet et al. [18], blijkt dat DMAS goede resultaten oplevert in dynamische omgevingen. Daarom werd besloten om niet verder te testen hoe DMAS zich gedraagt in omgevingen met mogelijke storingen. Het implementeren van een state-of-the-art-oplossing viel ook buiten dit project.

De AGV's zijn tijdens het schrijven van deze masterproef nog niet ontworpen en getest. Hierdoor is veel voertuig specifieke data niet beschikbaar, zoals onder andere de snelheid en benodigde tijd om te laden van een auto. Dat wordt in deze masterproef opgelost door het abstraheren van die data. We stellen de AGV's ook voor als klein genoeg om altijd tussen twee infrastructuurknooppunten te passen en nooit op meerdere infrastructuurknooppunten tegelijkertijd aanwezig te zijn. We gaan er ook van uit dat een constante snelheid volstaat en dat de versnelling en vertraging van de AGV's onmiddellijk is. De snelheid stelt de gemiddelde snelheid van een AGV voor. Alles wordt geïmplementeerd bovenop RinSim, uitgelegd in sectie 2.6, met een roadmodel dat geen overlapping of aanraking van de AGV's toelaat.

Door het bestaan van de YardSolver werd besloten om te focussen op een dial-a-ride probleem met deadlines. De YardSolver houdt zich voornamelijk bezig met het uitrekenen en optimaliseren van wanneer een taak kan worden uitgevoerd om altijd aan de deadlines te voldoen. Daardoor begint het pickuptijdslot op het moment waarop de taak vrijkomt in

het systeem. De reactietijd, zoals beschreven in subsectie 2.1.2 is dus niet aanwezig maar we spreken over een reactietijd die loopt van het vrijkomen van de taak en het eindtijdstip waarop een taak moet worden uitgevoerd voordat de rij- en laadtijdstippen langer zijn dan de toegelaten tijd. Door het gebrek aan een eindpunt van het pickuptijdslot, wordt er voornamelijk gekeken naar de leveringstraagheid, de pickuptraagheid wordt achterwege gelaten. Verdere optimalisatie met de YardSolver en het voorgesteld DMAS-systeem is mogelijk, maar er was geen toegang tot de YardSolver en mogelijke datasets tijdens het schrijven van deze masterproef. Het gebruik van de YardSolver wordt wel aangeraden door de mogelijkheid om de AGV's te laten samenwerken met het manuele werk dat op het moment van schrijven plaatsvindt. Er is een mogelijkheid om slechts bepaalde taken door te sturen naar de AGV's en andere taken aan de dockwerkers te overhandigen.

De infrastructuuragenten hebben kleine computationale mogelijkheden en kunnen geen intensieve berekeningen doen. Ze dienen enkel voor het doorsturen van mieren en het bijhouden van reservaties. De infrastructuuragenten op de parkeerplaatsen houden naast de reservaties ook nog hun taken en auto's bij die op hun locatie aanwezig zijn. De BDI-structuur van de AGV's wordt gebruikt om alle beslissingen te nemen die de AGV's nodig hebben om hun taken te volbrengen. Buiten de YardSolver die de taken uitgeeft, is er nog een centrale server nodig om administratieve informatie mee te delen aan de AGV's, zoals de locaties van taken. Daarnaast kan er worden beslist om het A*-algoritme te laten uitvoeren op een centrale server om computationeel werk weg te nemen van de infrastructuuragenten. Dit zorgt wel voor een nieuw knelpunt in de informatie of het netwerk. De centrale server krijgt ook de opdracht om mogelijke deadlocks op te lossen. Een parkeerplaats heeft namelijk maar één enkele ingang, terwijl het mogelijk is dat er in een visgraatparkeerplaats meerdere auto's geparkeerd worden. Dat betekent dat er, terwijl een AGV een auto aan het plaatsen is, een tweede AGV kan aankomen om zijn vracht in dezelfde parkeerlane te plaatsen. Er is namelijk geen centraal algoritme dat de AGV's aanstuurt en de AGV's hebben geen informatie over de locatie van andere AGV's. Dit probleem wordt verder besproken in sectie 4.6.

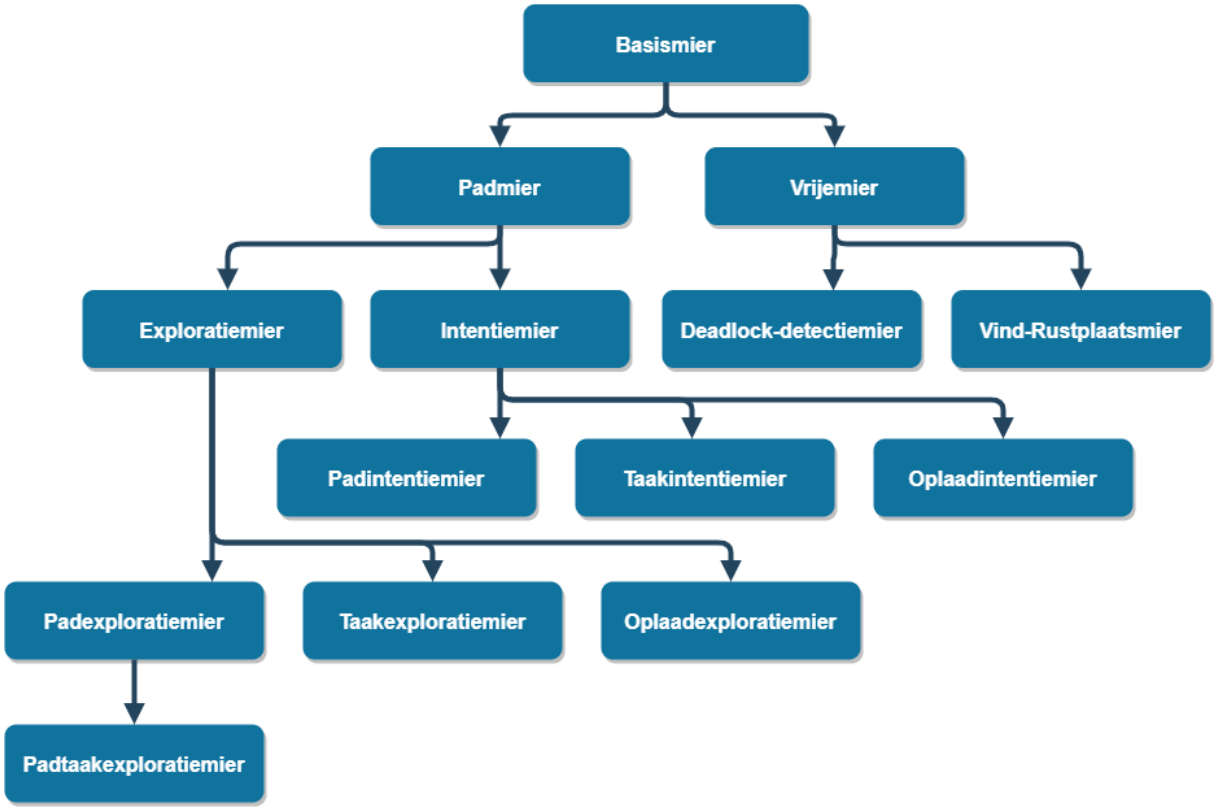
Alle AGV's hebben dezelfde bouw en kunnen dezelfde taken uitvoeren, ze zijn dus homogeen. Doordat alle taken ook enkel verplaatsingen van gelijkaardige auto's inhouden, worden ze ook als homogeen beschouwd. De implementatie van de sequentiële taakallocaties met DMAS gaat ervan uit dat alle taken door alle AGV's kunnen worden uitgevoerd. Een combinatie tussen de voorgestelde oplossing, mogelijke optimalisatie en veilingtechnieken, zoals besproken in subsectie 2.4.6, wordt niet verder bekeken.

De twee grote problemen, zoals omschreven in hoofdstuk 3, zijn de batchjobs en het uithalen van een specifieke auto uit een parkeerlane. Een AGV is in staat om één auto te verplaatsen en het is de enige taak die hij kan uitvoeren. Daaruit volgt dat we in een Single-Task robotsituatie (ST) zitten als we de door Korsah voorgestelde taxonomie, beschreven in sectie 2.4, gebruiken. Het verplaatsen van een auto kan beschreven worden als een simpele taak. Het verplaatsen van een reeks auto's om een bepaalde auto op te heffen of het verplaatsen van een volledige batchjob wordt dan weer omschreven als een samengestelde taak. De taken kunnen op twee manieren bekeken worden. In de eerste manier gaan we ervan uit dat er een restrictie aanwezig is om elke auto te verplaatsen waardoor elke andere auto die zich ervoor bevindt, moet worden verplaatst. Daardoor kan er zich

telkens maar één AGV bezighouden met een taak en pas als die eerste taak is afgerond, kan de volgende auto worden bereikt en dus ook een volgende taak vrijgegeven worden naar de DMAS-omgeving en de robots. Er wordt hier verder naar verwezen als de enkele-consumentmethode. De eerste manier komt daardoor neer op een ST-SR-IA-probleem. In de tweede manier spreken we over coalitievorming waar er n AGV's toegelaten worden om op hetzelfde moment met een taak bezig te zijn. Daaronder beschouwen we het verplaatsen van alle auto's in een batchjob of het verplaatsen van één enkele specifieke auto in een parkeerlane als een multirobot-taak die moet worden uitgevoerd door meerdere AGV's. In dit geval laten we meerdere AGV's naar de parkeerlane komen maar door de werking van DMAS is de aankomsttijd van iedere individuele AGV niet gegarandeerd. Dit wil zeggen dat elke AGV die zich voor deze coalitie aanmeldt, elke mogelijk taak moet kunnen uitvoeren die op die taaklocatie aanwezig is. Dankzij de homogene aard van de taken en de AGV's is dat een stuk makkelijker. In dit geval spreken we over een ST-MR-IA-probleem en wordt er in deze masterproef verder naar verwezen als de coalitiemethode. Beide vallen onder Cross-Schedule Dependencies, zoals beschreven in subsectie 2.4.4, door de invloed van de beslissingen die andere robots maken op zowel taakniveau als op het niveau van padbepaling. Als een robot beslist om zich toe te voegen aan een coalitie, wordt het volledige uitvoeren van de coalitietaken veel interessanter voor andere robots.

Als een robot een taak uitvoert waardoor een nieuwe taak beschikbaar wordt voor de andere robots, kunnen die andere robots beslissen om de nieuwe taak uit te voeren in plaats van een taak die ze daarvoor al hadden gekozen. Cross-schedule ST-SR-IA wordt opgelost door inter-taak restricties, bijvoorbeeld het vrijgeven van taken pas nadat aan bepaalde restricties voldaan is, toe te passen bovenop een taakallocatiesysteem zoals markt- of veiling-gebaseerd-systemen. Die markt- of veilingsystemen proberen we te vervangen door een DMAS-systeem waar de beslissingslogica aanwezig is in het BDI-systeem van de AGV's zelf. Cross-schedule ST-MR-IA-problemen worden standaard opgelost door coalitievorming, zoals besproken in subsectie 2.4.4. Om te zorgen dat de AGV's zo weinig mogelijk moeten wachten op een andere AGV die nog een vorige taak aan het uitvoeren is, voegen we een langzaam groeiende coalitiemethode in. We gebruiken hier opnieuw een DMAS-allocatiesysteem met beslissingslogica in de AGV's om de AGV's zichzelf te laten toewijzen aan de coalities. De vorming van de coalities is dynamisch doordat een AGV kan beslissen dat een andere taak een betere keuze is dan de huidige. Daardoor kan een AGV een coalitie verlaten voor een andere coalitie. De coalities zijn niet strict over het toewijzen van een specifieke taak aan een specifieke AGV en de taakallocatie is dynamisch en gebaseerd op het aankomen van de AGV op de locatie waar de taak aanwezig is. Dat kan er natuurlijk voor zorgen dat het uitvoeren van een taak langer duurt doordat AGV's van taak wisselen of er veel tijd zit tussen het uitvoeren van taak 1 door AGV 1 en taak 2 door AGV 2. Zulke situaties moeten worden vermeden. In subsectie 2.4.6 wordt de stelling van Rizk et al. in [31] besproken dat het simultaan vormen van coalities en het toewijzen van taken tot optimalere oplossing kan leiden. Door het dynamische toewijzen van taken aan de AGV's op het moment waarop ze aankomen en de gedistribueerde aard van DMAS, hopen we op een competitief alternatief met de state-of-the-art-algoritmen, besproken in sectie 2.4. Hierbij worden de dynamische aard van de casestudy, de reactietijd voor de prioritaire taken, de grote hoeveelheid AGV's op een groot domein en de hinder voor de robots door het verkeer op de site, in rekening genomen.

4.2 De mieren



Figuur 4.1: Relatiediagram van de gebruikte mieren.

In dit onderdeel worden de gebruikte mieren besproken en hoe ze geïntegreerd worden in de voorgestelde oplossing. Ook de feromonen die ze achterlaten, worden voorgesteld. Er zijn twee grote subsecties voor de gebruikte mieren, de vrijemieren die geen pad aflopen en de padmieren die een pad volgen dat bij creatie is meegegeven. Hoe de mieren zich verhouden tot elkaar is zichtbaar op Figuur 4.1. De vrijemieren worden onderverdeeld in deadlock-detectie- en vind-rustplaatsmieren die beide worden gebruikt in paragraaf 4.6. De vind-rustplaatsmieren worden ook gebruikt voor het zoeken naar mogelijke rustplaatsen zodat een AGV in rust zich niet op middelen (resources) bevindt die nodig zijn om andere processen uit te voeren. Dit wordt verder besproken in subsectie 4.7.5. De vrijemieren zijn voor lokale doeleinden bestemd en verplaatsen zich niet over grote afstanden over het netwerk, in tegenstelling tot de padmieren. De padmieren bestaan opnieuw uit twee grote onderverdelingen, die kenmerkend zijn voor DMAS, namelijk de intentie- en exploratiemieren. De exploratiemieren worden gebruikt om informatie te verzamelen in het netwerk die de keuzes van de AGV's beïnvloedt. Hoewel haalbaarheidsmieren (feasibility ants) ook in DMAS-oplossingen kunnen worden gebruikt, gebruiken we ze niet in de voorgestelde oplossing. Haalbaarheidsmieren kunnen zich afhankelijk van het doel in de subcategorie van zowel de pad- als de vrijemieren bevinden. Mogelijke verdere onderzoeken en optimalisaties die haalbaarheidsmieren kunnen bieden, worden in de volgende hoofdstukken kort toegelicht. De exploratiemieren die in de oplossing gebruikt worden zijn de volgende:

- Padexploratiemier: wordt verder besproken in sectie 4.4 en dient voor het opzoeken van paden en zo het beste pad te vinden van punt A naar punt B. Dit is dan ook de meest gebruikte mier en wordt voornamelijk door de AGV's zelf verzonden.

- Oplaadexploratiemier: wordt verder besproken in sectie 4.5 en heeft als hoofddoel informatie te verzamelen in de oplaadstations om de AGV's naar een vrij oplaadstation te leiden.

- Taakexploratiemier, wordt gebruikt om een taak te zoeken op het netwerk en de informatie van de taak terug te brengen naar de AGV om verdere beslissingen te nemen. Deze mieren worden verder besproken in sectie 4.7.

- Padtaakexploratiemier is direct verbonden met de padexploratiemieren maar wordt niet verzonden door de AGV's maar door infrastructuuragenten. De padtaakexploratiemieren bekijken of er een pad bestaat tussen het begin- en eindpunt van een taak. In sectie 4.7 wordt het gebruik van deze mier verder verduidelijkt.

De intentiemieren geven de intenties van de AGV's door aan het globale netwerk door het gebruik van feromonen zoals beschreven in sectie 2.3. Intentiemieren worden, in tegenstelling tot de exploratiemieren, enkel verzonden door de AGV's omdat alle beslissingen enkel en alleen door de AGV's kunnen worden genomen. De intentiemieren worden verder opgedeeld in drie subklassen, padintentie-, oplaadintentie en taakintentiemieren, die elk een specifieke intentie van de AGV propageren door het netwerk. Verdere toelichting op de werking en toepassing van elke intentiemier gebeurt in dezelfde paragrafen als de respectievelijke exploratiemier. In de verdere toelichting wordt ook specifieker vermeld welke feromonen elke intentie mier achterlaat en de impact van deze feromonen op het globale systeem.

4.3 Infrastructuuragenten

De verschillende soorten infrastructuuragenten die zich op het wegennetwerk bevinden, worden in deze paragraaf kort besproken. We definiëren vier grote groepen van infrastructuuragenten, zijnde:

- De ware infrastructuuragenten die zich op de wegen bevinden. Die hebben als hoofdfunctie het verwerken van mogelijke mieren die langs de agenten voorbijkomen en het begeleiden van de AGV's over de wegen door gebruik te maken van feromoneninfrastructuur en reservaties die de mieren achterlaten tijdens hun tocht. De infrastructuuragenten bouwen een lijst van reservaties op via de intentiemieren. De reservaties houden de zone met een tijdstip van aankomst en verlaten bij en een ID voor de AGV die de reservatie heeft gemaakt.

- De oplaadstationsagenten die de ingangen van oplaadstations voorstellen. Naast de functies van de infrastructuuragenten dienen deze agenten ook nog als aanspreekpunt voor de oplaadstations die ze voorstellen. De extra functies worden verder besproken in sectie 4.5.

- De opslagagenten die de verschillende mogelijke opslagplaatsen voorstellen. Er bestaan twee klassen van opslagagenten. De opslagagent stelt een parkeerplaats voor

die de AGV's slechts langs één kant kunnen benaderen. Het gaat dan voornamelijk om de import/exportpunten van het havenbedrijf en het visgraatpatroon voor de opslag van auto's. De dual-opslagagenten stellen ingangen van parkeerlanes voor waar AGV's de opslagplaats langs twee kanten kunnen benaderen. Elke dual-opslagagent staat in contact met een tweede dual-opslagagent die de andere ingang van zo'n parkeerlane voorstelt. De opslagagenten voeren de functies van de infrastructuuragenten uit maar dienen ook als aanspreekpunt voor het opnemen van taken, en dus het maken van coalities. Daarnaast houden ze de taken, voertuigen en coalitievoortgang bij. In paragraaf 4.7 wordt hier in meer detail op ingegaan.

- De laatste klasse is een groep rustplaatsagenten. Deze stellen plaatsen voor waar een AGV kan rusten, in ijdele modus, zonder dat deze het verkeer op de werkwegen stoort.

Een infrastructuuragent is verantwoordelijk voor het punt waar de agent zich op bevindt en alle mogelijke inkomende wegen naar dit punt. Dat heeft als gevolg dat als er geen overlappende reservaties zijn op een infrastructuuragent, er zich altijd slechts één AGV op het verantwoordelijke punt bevindt, of op weg is naar dat punt. Dit verhindert dat meerdere AGV's zich naar een bepaald punt kunnen begeven om dan een obstructie te vormen voor elkaar, waardoor menselijke tussenkomst noodzakelijk is.

4.4 Padbepaling en verplaatsingen van AGV's

Padbepaling en het uitvoeren van de geplande paden zijn cruciaal om de taken te kunnen uitvoeren. De padbepaling voor de AGV's steunt op het padbepalingsalgoritme, voorgesteld in subsectie 2.3.2, waar we gebruikmaken van padexploratiemier om interessante paden te verkennen, eventueel aangevuld met padintentiemier om het geselecteerde pad te reserveren. De AGV's houden zowel hun intenties als hun eigenlijke routes bij en ze bepalen hun routes via een BDI-structuur. De beliefs die een agent heeft is de grafenstructuur van het netwerk. Die hoeft niet volledig up-to-date te zijn, als bepaalde paden wegvallen door onderbrekingen kan er een ander pad gekozen worden zelfs als er geen antwoord verkregen wordt op een deel van de verzonden mieren. De beste routes kunnen worden gegarandeerd door een flooded broadcast van de mieren over het netwerk. Bij een flooded broadcast worden berichten naar ieder naburig knooppunt verstuurd omdat de weg naar het eindpunt niet gekend is. Dat zorgt voor veel netwerkverkeer, zeker in grote netwerken, en vaak zonder aanzienlijke verbeteringen. Om dit probleem op te lossen, wordt er een variant op het A*-algoritme gebruikt, zoals voorgesteld in [7] en geïllustreerd in Algoritme 2.

Op die manier worden mogelijke kandidaatsoplossingen gegenereerd op de graaf. Slecht gegenereerde kandidaatsoplossingen leiden tot slechte plannen. Alle mogelijke routes tussen twee punten bekijken, is onpraktisch door het schaalprobleem dat de flooded broadcast ook veroorzaakte. Anderzijds, als enkel de kortste paden tussen twee punten berekend worden, is er een grotere kans dat het verkeer op die paden ophopingen veroorzaakt. De set van mogelijke oplossingen voor de route tussen twee punten moet dus verschillend zijn voor de verschillende AGV's om ophopingen te vermijden en tegelijkertijd goed genoeg zijn om AGV's efficiënt te verplaatsen. Het eerste gegenereerde pad wordt berekend op de standaardgraaf met het standaard A*-algoritme en heeft dus het kortste pad als gevolg,

Algoritme 2 Alternatief padzoek algoritme gebasseerd op A* met boetes gebaseerd op kansen

Require: Infrastructuurgraaf G = (V,E), aantal gewenste alternatieve paden N, huidige locatie start, bestemming dest, standaardboete boete

```
1:  S ← ∅
2:  numFails ← 0
3:  maxFails ← 3
4:  α ← Uniforme willekeurige waarde tussen 0 en 1
5:  while sizeOf(S) < N ∧ numFails < maxFails do
6:      p ← getShortestPath(G,start,dest)
7:      if p ∈ S then
8:          numFails ← overlap + 1
9:      else
10:         numFails ← 0
11:         S ← S ∪ p
12:     for all vertex V_i ∈ p do
13:         β ← Uniforme willekeurige waarde tussen 0 en 1
14:         if β < α then
15:             for all edge E_i direct verbonden met V_i do
16:                 weight(E_i) ← boete
    return S
```

zichtbaar op lijn 6. Voor verdere paden wordt een uniform willekeurig nummer α tussen 0 en 1 gegenereerd. Voor elke sprong op het gevonden pad wordt een nummer β gegenereerd, opnieuw uniform willekeurig tussen 0 en 1, zoals zichtbaar op lijn 12. Als $\beta < \alpha$ dan wordt er op het gewicht van deze sprongen een boete bijgerekend, zoals voorgesteld in lijn 14-16. Die boete is een standaardwaarde meegegeven aan het algoritme. Grotere boetes zorgen voor een kleinere kans dat een sprong herkozen wordt, maar kan ook slechtere paden als gevolg hebben. Bij een kleine waarde voor α zijn de gevonden paden redelijk gelijkaardig, bij een grote waarde zijn de ze redelijk verschillend. Als er volledige overlappingen zijn tussen de berekende paden wordt de numFails verhoogd, als $numFails \geq maxFails$ dan worden er geen paden meer gevonden die niet volledig overlappend zijn en stopt het algoritme. Via deze stochastische methode kunnen de AGV's zelf paden berekenen die verschillend zijn voor elke iteratie van het algoritme, de kansen op de boetes leiden namelijk tot niet deterministische oplossingen. Die verschillende paden zorgen er dan voor dat het netwerk niet overbelast raakt in bepaalde efficiënte knooppunten. Al bij al zorgt deze manier van paden zoeken ervoor dat niet steeds dezelfde paden geselecteerd worden en dus ook dat die paden niet overbelast raken door de grote hoeveelheid AGV's en mieren.

Na het berekenen van een vooraf ingesteld aantal paden creëert de AGV de padexploratiemier, waarbij elke mier exact één pad toegewezen krijgt. Bij de reservatie die de AGV bij het begin van het zoeken heeft, wordt de eindtijd vertraagd om tijd te geven aan de mier om zich over het netwerk te verplaatsen. Als dat niet zou gebeuren, zou het verkregen pad van reservaties correct zijn, mocht de AGV ook effectief vertrekken op het moment waarop de mieren worden verstuurd, het duurt namelijk even voor de mieren terugrapporteren. De vertraging geeft de AGV en mier de tijd om alle berekeningen door

te voeren en en beslissingen te nemen, gebaseerd op geldige informatie. De padexploratiemieren worden eerst naar de infrastructuuragent gestuurd waar de AGV zich op dat moment bevindt of naar de agent waar de AGV zich naartoe beweegt als de AGV zich aan het verplaatsen is tussen twee infrastructuuragenten. De padexploratiemier verplaatst zich dan van infrastructuuragent naar infrastructuuragent, die de mier naar de volgende sprong doorsturen, zoals in de basis DMAS-oplossing zichtbaar op Figuur 2.3. Op elke infrastructuuragent verzamelt de mier de tijdstippen waarop de AGV kan toekomen en vertrekken in de vorm van een tijdelijke reservatie, gebaseerd op de al aanwezige reservaties op de infrastructuuragent. De voorgestelde tijdstippen voor de tijdelijke reservatie worden berekend op de vorige voorgestelde reservatie op de vorige infrastructuuragent. Daarnaast wordt er ook een mogelijke maximale eindtijd toegevoegd waarop de AGV zich moet verplaatsen van de vorige infrastructuuragent. Als de AGV zich niet zou verplaatsen voor die maximale eindtijd, is er overlapping met andere gereserveerde tijdstippen door andere AGV's. Er wordt ook verwacht dat er een tijdsbestek zit tussen twee AGV-reservaties. Als er geen tijdsbestek blijkt te zijn, laat de infrastructuuragent de mier vallen. Niet alle mogelijke tijdstippen zijn geldige tijdstippen en om een volledig correcte verbonden reservatiereeks te berekenen, is mogelijks een niet ondersteunde terugkerende beredenering (backtracking) nodig, wat vermeden wordt door de mier te laten vallen. Het laten vallen van mieren gebeurt ook als een infrastructuuragent de mier niet kan doorsturen naar de volgende infrastructuuragent. Door het sturen van meerdere mieren kan het laten vallen van een onmogelijk pad geen kwaad. Er wordt een andere oplossing gevonden en het pad kan nog verder worden geoptimaliseerd tijdens het verplaatsen van de AGV. Als de infrastructuuragenten alle mieren laten vallen, zal de AGV geen antwoord krijgen op zijn verstuurde mieren tijdens een geassocieerde wachtperiode. Na de wachtperiode zal de AGV, als er geen antwoord kwam op zijn verstuurde mieren, opnieuw een pad proberen te bepalen door paden te herberekenen, op een mogelijks geüpdate graaf, en vervolgens nieuwe mieren te versturen. Als een mier de laatste sprong bereikt, en dus aankomt op de eindbestemming, wordt er een oplaadexploratiemier aangemaakt. De reden en het doel van de tweede mier wordt beschreven in sectie 4.5. Als het antwoord op de tweede mier dan door de verzendende infrastructuuragent ontvangen wordt, worden beide mieren teruggestuurd naar de zender van de padexploratiemier, de AGV.

Als de wachttijd van AGV verstreken is of hij een aantal mieren heeft ontvangen, sorteert de AGV de correcte mieren gebaseerd op de begintijd van de laatste reservatie, die aantoont wanneer de AGV op zijn bestemming aankomt. De infrastructuuragenten beslissen zelf hoe lang en hoe snel de AGV over het stuk van de weg rijdt waar zij verantwoordelijk voor zijn. Doordat de opeenvolging van reservaties rekening houdt met de mogelijke begin- en eindtijden van elke reservatie, is de laatste begintijd van de opeenvolging van reservaties representatief voor de aankomst van de AGV als dit pad wordt gevolgd. De AGV selecteert vervolgens het mogelijk kortste pad in tijd en begint zijn reservatieproces. De tijdelijke reservaties en het gevolgde pad worden meegegeven met een padintentiemier. Die padintentiemier volgt hetzelfde pad en licht de gekozen tijdelijke reservatie toe aan de infrastructuuragenten. Dit werd al eens toegelicht en dit proces is zichtbaar op Figuur 2.3. Als de tijdelijke reservatie nog steeds geldig is, wordt ze opgenomen in de infrastructuuragent en ontvangt ze een feromoneninfrastructuur. Als de feromoneninfrastructuur vervalt of als de eindtijd van de reservatie verlopen is, wordt de reservatie verwijderd. Als de tijdelijke reservatie niet meer geldig is, wordt de mier teruggestuurd naar de AGV met

een verwerpingsvlag. Als de eindtijd van de huidige vertraagde reservatie het nog toelaat, probeert de AGV het volgende beste pad te reserveren. Mocht dat niet meer zo zijn, start de AGV opnieuw met het zoeken naar een pad en voegt hij een nieuwe vertraging toe op zijn huidige reservatie. Als enkel een latere reservatie verworpen wordt, dan worden de vorige aangemaakte reservaties verwijderd doordat ze niet hernieuwd worden door de AGV's en komen hun tijdslots terug vrij. Als alle reservaties aangenomen worden door de infrastructuuragenten, worden ze teruggestuurd naar de AGV waarna die de reservaties volgt om tot zijn eindbestemming te komen. Op regelmatige tijdstippen stuurt de AGV een nieuwe padintentiemier die ditmaal niet de reservaties vastzet maar de feromonen van de reservaties hernieuwt. In geval van vertragingen, zoals toegelicht in subsectie 2.3.3, wordt de nieuwe informatie teruggebracht naar de AGV via de mieren om de reservaties die de AGV bijhoudt op die manier te updaten. Als daarbij een padintentiemier terugkeert met de niet-geaccepteerde vlag, rijdt de AGV naar de volgende infrastructuuragent en start hij een zoektocht naar een nieuw pad.

Om de paden verder te optimaliseren, stuurt de AGV tijdens het rijden ook voortdurend nieuwe padexploratiemieren naar zijn bestemming. Voor hen worden de paden op dezelfde manier berekend als voor de normale padexploratiemier. Bij het terugkeren van die mieren worden de paden eerst gefilterd op basis van de vraag of ze nog van toepassing zijn. Sommige paden vragen echter een mogelijke andere afslag dan de AGV al gemaakt heeft tijdens het volgen van het huidige pad. Het duurt namelijk even voor er antwoord komt op de verstuurde mieren. De gefilterde lijst wordt dan opnieuw gesorteerd op zoek naar een beter pad. Als een beter pad wordt gevonden, stuurt de AGV opnieuw een padintentiemier. Enkel als die terugkomt met een nog steeds mogelijk pad waarvan alle reservaties geaccepteerd zijn, neemt de AGV het nieuwe pad volledig over. De gemaakte reservaties die nu niet meer gevolgd worden, worden opnieuw door de feromoneninfrastructuur langzaamaan van het netwerk weggewerkt en de tijdslots die deze reservaties innamen, komen opnieuw vrij. Andere AGV's kunnen de vrijgekomen tijdslots dan ook tijdens het zoeken naar betere paden gebruiken voor het optimaliseren van hun routes.

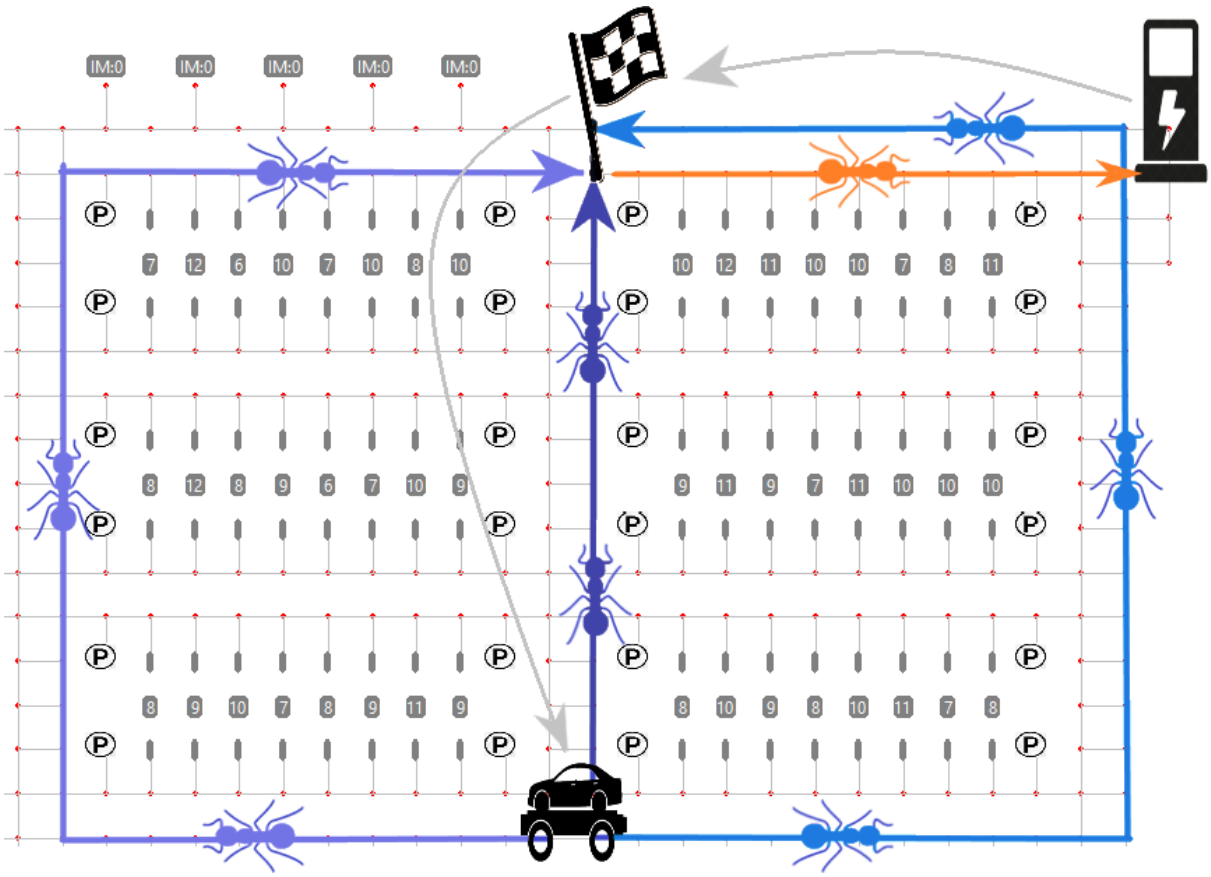
4.5 Opladen van de AGV's en hun bereik

In de vorige sectie werden de padexploratie- en padintentiemieren besproken en hun bijdrage aan de oplossing. In deze sectie worden die mieren nu uitgebreid met de oplaadexploratiemieren voor het vinden van lokale informatie over de totale afstand die de AGV's afleggen als ze dit pad volgen en zich daarna naar een oplaadstation begeven. Dit is het eerste doel van de oplaadmieren. Daarnaast wordt ook het tweede doel van de oplaadintentiemieren besproken en hoe de AGV's bij oplaadstationsagenten een tijdsslot kunnen reserveren om hun batterij op te laden.

De oplaadexploratiemieren worden ten eerste verzonden wanneer een AGV een padexploratiemier verstuurd om een pad te zoeken met een infrastructuuragent als eindbestemming van de die mier. De eindbestemming infrastructuuragent zal in dit geval een oplaadexploratiemier aanmaken voor het verkrijgen van extra informatie. Die oplaadexploratiemier wordt verstuurd naar de dichtste oplaadstationsagent, dit proces wordt voorgesteld in Figuur 4.2. Tijdens het volgen van dit pad verzamelt de oplaadexploratiemier ook mogelijke

reservaties, net zoals de padexploratiemier. Het is ook mogelijk dat een oplaadexploratiemier geen correcte toewijzing kan krijgen en dat de infrastructuuragenten hem laten vallen. Daarom worden er meerdere paden berekend, net zoals bij de padexploratiemier, die worden meegegeven aan verschillende oplaadexploratiemieren. Als de oplaadstationsagent de oplaadexploratiemier ontvangt, wordt die laatste teruggestuurd naar de zender van de oplaadexploratiemier. De eerste terugkomende oplaadexploratiemier wordt dan gekoppeld aan de padexploratiemier. Als er geen oplaadexploratiemier wordt ontvangen, dan wordt er een vlag geplaatst op de padexploratiemier, waarmee de AGV rekening houdt. De combinatie van oplaadexploratie- en padexploratiemier dient als een filter die de AGV gebruikt om te meten of een pad mogelijk is, omdat we niet geïntresseerd zijn in het snelste pad maar wel of een pad mogelijk is. De afstanden en tijden van zowel de padexploratie- en de oplaadexploratiemier worden samengekoppeld tot een waarde die de mogelijke impact op de batterij vertegenwoordigt. Als $\gamma * BatterijImpact > HuidigEnergie$, dan wordt de padexploratiemier verworpen. Die waarde γ schrijft voor dat het geplande pad langer kan zijn dan oorspronkelijk voorzien. Een AGV met een lege batterij is een kost die moet worden vermeden. Naast herschaalde batterij-impact moet er dus ook altijd bepaald batterijpercentage overblijven. Als de vlag geplaatst is op de padexploratiemier die aangeeft dat er geen pad naar een dichtbijgelegen oplaadstation gevonden is, neemt de AGV enkel het pad aan als er garantie is dat er geen batterijtekort kan zijn. Om dezelfde reden kan de AGV een vlag plaatsen op de padexploratiemier zodat de infrastructuuragenten de oplaadexploratiemier niet versturen.

Ten tweede worden oplaadexploratiemieren ook gebruikt om te kijken of een oplaadstationsagent een reservatie kan maken om op te laden. Het aanmaken van een reservatie op die manier gebeurt enkel door de AGV's als hun batterijniveau te laag komt te staan. Dat laat toe dat er een realistische schatting kan worden gemaakt om te weten of er op het moment van aankomst een plaats vrij is in een oplaadstation. Om een oplaadstation te vinden, stuurt de AGV een oplaadverzoek naar verschillende oplaadstations via de oplaadexploratiemieren. Die mieren worden dan teruggestuurd met het eerstvolgende tijdslot waarop de AGV zich zou kunnen opladen. Gebaseerd op de terugkerende oplaadexploratiemieren, kiest de AGV een mogelijk oplaadstation. Als er geen oplaadstations beschikbaar zijn, wacht de AGV een bepaalde tijd voor hij opnieuw probeert. Als een AGV bijna geen batterijniveau meer heeft, kan hij beslissen om zich naar de dichtstbijzijnde oplaadstation te begeven en te accepteren dat een queue mogelijk is. Het bevestigen van de reservatie gebeurt door een oplaadintentiemier te creëren met de voorgestelde reservatie en die dan naar de oplaadstationsagent te sturen. Als de reservatie verworpen wordt door de oplaadstationsagent, begint dit proces opnieuw. Bij acceptatie start de AGV met een padbepaling naar de oplaadstationsagent terwijl hij ook periodiek opnieuw oplaadintentiemieren naar de oplaadstationsagent stuurt om de feromoneninfrastructuur te hernieuwen. De AGV kan ook beslissen dat dat opladen niet meer de beste beslissing is, bijvoorbeeld bij het vrijkomen van een nieuwe prioritaire taak. Dan verwijdert de feromoneninfrastructuur de reservatie opnieuw en laat ze andere AGV's toe de tijdslots in te nemen. Tegenovergesteld kan een AGV ook inzien dat tijdens het rijden naar een punt, de AGV een batterijtekort zal hebben en zich eerst zal moeten opladen. Tijdens het uitvoeren van taken is het voor de AGV verboden om zich te gaan opladen, dit houdt in dat voor het aannemen van een taak de AGV moet zeker zijn dat hij de taak kan uitvoeren, zelfs met onvoorziene problemen. De oplaadfunctie van de oplaadstations wordt



Figuur 4.2: Samenwerking van padexploratiemieren en oplaadexploratiemieren. Padexploratiemieren en hun paden worden voorgesteld door de verschillende blauwe paden, elke tint stelt een ander pad van één padexploratiemier voor. De oplaadexploratiemieren worden voorgesteld door het oranje pad. De oplaadexploratiemieren worden verstuurd als een padexploratiemier toekomt op de eindbestemming. Als de oplaadexploratiemier toekomt bij het oplaadstation wordt die direct teruggerapporteerd aan de infrastructuuragent. De infrastructuuragent stuurt daarna meteen alle verzamelde informatie direct terug naar de AGV. De grijze pijlen staan voor het direct terugrapporteren van de mieren als ze toekomen op hun eindbestemming.

in deze masterproef geabstraheerd naar een lineaire functie gebaseerd op tijd. Door het gebrek aan informatie over de batterij van de AGV, de oplaadinfrastructuur en dus ook de oplaadsnelheid, is het noodzakelijk om deze gegevens te abstraheren. Door die abstracties valt het implementeren van een ingewikkeldere en realistischere oplaadfunctie buiten dit werk.

4.6 Deadlocks

In de vorige secties werd besproken hoe de AGV's hun pad bepalen en hoe de AGV's verzekeren dat ze nooit een batterijtekort zullen hebben tijdens het uitvoeren van een pad. In deze sectie gaan we even terug naar het padbepalingsalgoritme en bekijken we in detail hoe deadlocks vermeden worden tijdens het zoeken van paden. In deze sectie bespreken we ook hoe de deadlocks die voorkomen opgelost worden en welke speciale

maatregelen er genomen worden om te zorgen dat de parkeerlanes vlot verkeer toelaten en geen deadlock veroorzaken. Deadlocks komen voornamelijk in twee gevallen voor, die allebei te maken hebben met tweerichtingsverkeer op plaatsen waar maar één enkele AGV voorbij kan. Deadlocks ontstaan door de manier waarop de coördinatie gebeurt in DMAS, elke AGV beslist namelijk over zijn eigen paden, en door het tekort aan een centraal algoritme dat de besloten paden controleert op mogelijke deadlocks. Dat houdt dus in dat er een anti-deadlocksysteem moet worden geïmplementeerd en indien mogelijk ook een systeem dat controleert of er deadlocks ontstaan.

4.6.1 Deadlockpreventie in padbepaling

Om deadlocks in paden te vermijden, wordt tijdens het bepalen van mogelijke reservaties in de infrastructuuragenten bekeken of er een deadlock zou kunnen ontstaan. Dit gebeurt door in de reservaties ook bij te houden wat de vorige en volgende mogelijke posities zijn. Door naar de vorige en volgende reservatie te kijken waarbinnen de mogelijke nieuwe reservatie plaatsvindt, kan er worden bekeken of er een mogelijke deadlock ontstaat, gebaseerd op het pad van de mier. Deadlocks treden voornamelijk op als de vorige en volgende positie bij reservatie 1 elk overeenkomen met respectievelijk de volgende en vorige positie van reservatie 2. In dat geval wordt er een deadlockreservatie aangemaakt en meegegeven met de mier in plaats van een normale reservatie. Als de AGV de mier ontvangt terwijl hij op zoek is naar een mogelijk pad, worden alle mieren met deadlockreservaties weggefilterd. In het geval dat er enkel een mogelijk pad wordt gezocht, zoals bij de oplaadexploratiemieren die gestuurd worden door de infrastructuuragenten na het ontvangen van een padexploratiemier, dan dienen de mieren enkel ter inlichting en hebben de mogelijke deadlocks geen invloed op de beslissingen van de AGV.

Bij het in- en uitrijden van parkeerlanes is er maar één mogelijk pad. Wachten tot een pad zonder deadlocks mogelijk is, is echter ook geen efficiënte oplossing omdat het lang kan duren voor er een parkeerplaats bereikbaar is door de tijdspanne van het laden en lossen door de AGV's. Daardoor worden de opslagagenten geconfigureerd om geen deadlockreservaties aan te maken. Als een deadlock optreedt bij het in- en uitrijden van de parkeerplaatsen, moet die anders worden opgelost om efficiëntie te bewaren.

4.6.2 Edge-chasing implementatie in DMAS

Het edge-chasing algoritme, zoals besproken in subsectie 2.5.1, stuurt berichten tussen middelen en processen om een mogelijke deadlock te detecteren. Door de bestaande infrastructuur voor de mieren is het aangeraden om de berichten voor te stellen door een andere soort mieren. De deadlock-detectiemieren worden aangemaakt door een AGV als een pad dat de AGV verwachtte correct te kunnen uitvoeren, plots wordt onderbroken. De deadlock-detectiemieren worden dan naar de infrastructuuragent gestuurd waar de AGV zich op bevindt. De deadlock-detectiemieren wordt dan vanaf die infrastructuuragent naar het proces gestuurd, of dus de AGV die de infrastructuuragent op dat moment verwacht te ondersteunen. De AGV die die mieren ontvangt, stuurt hem door naar het middel waartoe hem de toegang is geweigerd. Bij elke stap houdt de mier ofwel de coördinaten van de infrastructuuragent, ofwel het ID van de AGV bij. Als een duplicaat-ID of coördinaat wordt gevonden, is er sprake van een deadlock omdat er een

cirkel in de verwijzingen gevonden is. Als er geen cirkel wordt gevonden, worden de deadlock-detectiemieren beëindigd en eindigt het proces hier. In het andere geval licht een infrastructuuragent of een AGV een centrale deadlock-oplossingsserver in, met behulp van de deadlock-detectiemier. Die centrale deadlock-oplossingsserver stuurt commando's naar een deel van de AGV's, namelijk naar diegene die de deadlocks veroorzaken, die hen verplichten om zich naar een rustplaats te begeven en hen daar laat wachten. Als de centrale server ziet dat de deadlock mogelijk opgelost is door te kijken naar de middelen die nog in gebruik zijn, laat de deadlock server de AGV's weten dat hun wachtperiode voorbij is en dat ze hun acties mogen voortzetten. De deadlock-oplossingsserver zal de AGV's die geladen zijn en dus een taak bevatten, waar mogelijk verkiezen om hun taken verder te zetten. Andere AGV's die geen taken hebben, worden verzocht om zich naar een rustplaats te begeven en hun mogelijk gekozen taken te verlaten om de deadlock op te lossen.

4.6.3 Rustplaatsen

Rustplaatsen zijn ofwel specifieke plaatsen waar een AGV kan wachten, ofwel opslagagenten die geen taken bevatten en geen inkomend verkeer verwachten. Oplaadstations, het wegennetwerk en opslagagenten die taken bevatten of een taaklevering verwachten, staan geen AGV's in rust toe. Dat is om het mogelijke verstoren van de geplande activiteiten te vermijden. AGV's zullen zich dus altijd eerst naar een rustplaats verplaatsen voor ze over een nieuwe taak of oplaadlocatie beslissen. Het zoeken van paden vanop niet-rustplaatsen wordt wel toegelaten, net zoals het zoeken naar betere taken of oplaadstations door een AGV in beweging. De rustplaatsen worden door de AGV's verkend door het sturen van een haalbaarheidsmier, namelijk de vind-rustplaatsmier. Die overloopt het netwerk met behulp van een flooded broadcast met een incrementele maximale hop teller. Als een sprong niet mogelijk is, laat de infrastructuuragent de mier vallen. Elke rustplaats die nog niet in gebruik is, stuurt een antwoord op de vind-rustplaatsmier met zijn coördinanten en ID. De AGV sorteert die antwoorden gebaseerd op de afstand van de coördinaten en zijn huidige locatie, met de dichtste eerst. De AGV gaat de lijst af tot hij een positief antwoord krijgt op de vraag of de locatie nog steeds beschikbaar is. Dan start de AGV het padbepalingsalgoritme en verplaatst hij zich naar de rustplaats. Eenmaal aangekomen, laat de AGV zijn locatie weten aan de deadlock-oplossingsserver als hij van hem de verplaatsingsopdracht kreeg. Daarop geeft de deadlock-oplossingsserver de AGV een wachttijd. Als een rustplaats niet meer geldig is, zal de AGV zich naar een andere rustplaats verplaatsen als hij nog steeds geen beslissing heeft genomen.

4.6.4 Optimalisatie door HoldIA

Om mogelijke deadlocks te vermijden op de in- en uitrit van de parkeerlanes, worden Holdinfrastructuuragenten (HoldIA) voorgesteld. Die bevinden zich op het wegennetwerk en net naast een opslagagent. De HoldIA's gebruiken haalbaarheidsmieren om de opslagagent te vinden aan wie ze verbonden zijn. Om een HoldIA te betreden, moet een AGV zijn eindbestemming meegeven in de aanvraag of hij mag verder rijden. Als de eindbestemming overeenstemt met de verbonden opslagagent, zal de HoldIA eerst via een direct bericht bekijken of er plaats is op de opslagagent. Is dit het geval, dan laat de HoldIA de AGV passeren en naar de opslagagent rijden. Is dit niet het geval, dan staat hij niet

toe dat een AGV zijn zone inrijdt en een mogelijke deadlock veroorzaakt. Wanneer de opslagagent vrijkomt doordat de AGV die hem bezette, is vertrokken, staat de HoldIA toe dat de AGV zich verder beweegt. Treedt er wel een deadlock op door de vertraging, of door meerdere AGV's die dezelfde opslagagent willen betreden, dan start de HoldIA een speciaal geval op in de deadlock-oplossingsserver. Die geeft de AGV een commando om een rustplaats op te zoeken tot de opslagagent terug beschikbaar is om te betreden. Doordat de AGV's die zich op de opslagagent bevinden op die manier niet vertraagd worden door een mogelijke deadlock, hopen we dat dit een efficiëntere oplossing toelaat. Een HoldIA laat in tegenstelling tot de andere knooppunten op het wegennet wel toe dat een AGV op hen wacht. Op voorwaarde dat de AGV op weg is naar een taak en er een queue gevormd is naar het punt van die taak.

4.7 Toewijzen van taken

In dit deel worden de toewijzingen van de taken aan de AGV's in detail besproken. In sectie 4.4 werd besproken hoe de AGV's hun paden beslissen. Dit werd uitgebreid met secties 4.5 en 4.6 om te zorgen dat AGV's kunnen opladen, een lege batterij vermijden en hoe deadlocks worden vermeden en opgelost. Nu dat de AGV's correct paden kunnen vinden en uitvoeren, bekijken we hoe de AGV's hun taken vinden en uitvoeren.

Als eerst bespreken we hoe de taken aan de opslagplaatsen worden toegewezen en hoe de parkeerlanes bepalen hoe de taakverdeling tussen de verschillenden ingangen wordt verdeeld. De twee verschillende methodes, de enkele-consument- en coalitiemethodes, worden hier verder toegelicht, net als hun integratie met de totaaloplossing en de uitwerking van het verschil tussen de twee methodes voor de rest van de beslissingen tussen de AGV's. In de enkele-consumentmethode worden alle taken als individuele taken aanzien en dus ook individueel uitgevoerd. In de coalitiemethode spreken we van een langzaam groeiende coalitiemethode. Om de coalitie te laten groeien laten we maar een gelimiteerd aantal AGV's zich aan dit groeiproces koppelen. Als teveel AGV's tegelijkertijd zich aan de coalitie zouden koppelen en dus ongeveer terzelfder tijd toekomen aan de opslagagent, zullen er grotere wachttijden geïntroduceerd worden. De taak van de taakexploratie- en taakintentiemieren wordt hier ook in detail aangekaart en hun rol in het ontdekken en beslissen van de taken door de AGV's.

4.7.1 Taken in opslagplaatsen

De opslagagenten ontvangen hun taken via een centrale server die via de YardSolver een taakschema doorkrijgt. De opslagagenten houden, naast de centrale server, zelf ook hun taken en alle informatie bij die bij die taken hoort. Bij opslagagenten die maar één enkele inkom hebben, wordt een lijst opgemaakt met mogelijke taken. Als er voor een taak eerst een andere auto moet worden verplaatst, maken de opslagagenten ook een vasthoudtaak aan per auto die moet worden verzet. Die vasthoudtaak kan tot op het moment van de pickup gewijzigd worden naar een leveringstaak. Vasthoudtaken zijn taken waarbij de auto niet verplaatst moet worden maar enkel tijdelijk moet worden bijgehouden door een AGV tot alle taken op de opslagagent opgepikt zijn. Een leveringstaak is een taak waarbij een AGV een taak of auto oppikt en die naar een andere locatie vervoert. We

gaan ervan uit dat alle taken die op een opslagagent moeten worden uitgevoerd op een korte tijdsperiode naar de opslagagent verstuurd worden. Het is immers niet optimaal als er een hoeveelheid auto's meermaals moet worden verzet. We gaan er dus vanuit dat de YardSolver dit optimaliseert.

Bij dual-opslagagenten stuurt de centrale server alle taken door naar de dual-opslagagenten die hen verbinden met de andere ingang van de parkeerlane. De dual-opslagagenten berekenen dan op basis van alle taken wat de beste taakverdeling is, algoritme 3 beeldt dit proces uit.

Algoritme 3 Optimale toewijzing van taken aan dual-opslagagenten

Require: dual-opslagagenten left, dual-opslagagenten right, TaskList tasks

1: *nextNodes* $\leftarrow \emptyset$
2: *currentNodes* $\leftarrow \emptyset$
3: *tasksLeftNode* $\leftarrow \emptyset$
4: *tasksRightNode* $\leftarrow \emptyset$
5: *currentNodes* $\leftarrow$ *currentNodes* $\cup$ *generateStartNode()*
6: **for all** *Task* $T_i \in$ *tasks* **do**
7: **for all** *Node* $N_i \in$ *currentNodes* **do**
8: *nextNodes* $\leftarrow$ *nextNodes* $\cup$ *generateRightNode*(T_i,N_i) $\cup$ *generateLefttNode*(T_i,N_i)
9: *currentNodes* $\leftarrow$ *nextNodes*
10: *nextNodes* $\leftarrow \emptyset$
11: *lowestScoreNode* $\leftarrow$ *getLowestScore(nextNodes)*
12: **while** *hasParent(lowestScoreNode)* **do**
13: **if** *isLeftNode(lowestScoreNode)* **then**
14: *tasksLeftNode* $\leftarrow$ *tasksLeftNode* $\cup$ *lowestScoreNode*
15: **else**
16: *tasksRightNode* $\leftarrow$ *tasksRightNode* $\cup$ *lowestScoreNode*
17: *lowestScoreNode* $\leftarrow$ *parent(lowestScoreNode)*
18: *allocateTasks(tasksLeftNode, left)*
19: *allocateTasks(tasksRightNode, right)*

Ten eerste wordt er op lijn 5 een beginknooppunt (rootnode) aangemaakt die de huidige situatie voorstelt. Per taak wordt er dan een nieuw niveau toegevoegd aan de boomstructuur, het rechterkind van elke knooppunt stelt de hoeveelheid verplaatsingen voor die de taak nodig heeft als hij wordt toegewezen aan opslagagent 1 en elke linkerknooppunt stelt de hoeveelheid verplaatsingen voor als de taak toegewezen wordt aan opslagagent 2. Dit proces is te zien in de for-lus op lijn 7. Elke knooppunt in de boom krijgt ook het ID van de taak mee, het vorige knooppunt en of hij een linker- of rechterkind is van het vorige knooppunt. Elke taak die dient te worden toegewezen, wordt één per één verwerkt in een nieuwe laag in de boom, dit gebeurt in de for-lus op lijn 6. De nieuwe laag stelt dan alle verplaatsingen voor die nodig zijn om alle taken die tot die laag of hoger behoren, uit te voeren. Als alle taken toegewezen zijn in de boom, wordt in de onderste laag voor elk knooppunt een score berekend en wordt het knooppunt met de laagste score opgezocht. Hier is dat enkel de hoeveelheid verplaatsingen die dienen te worden uitgevoerd. Hoe

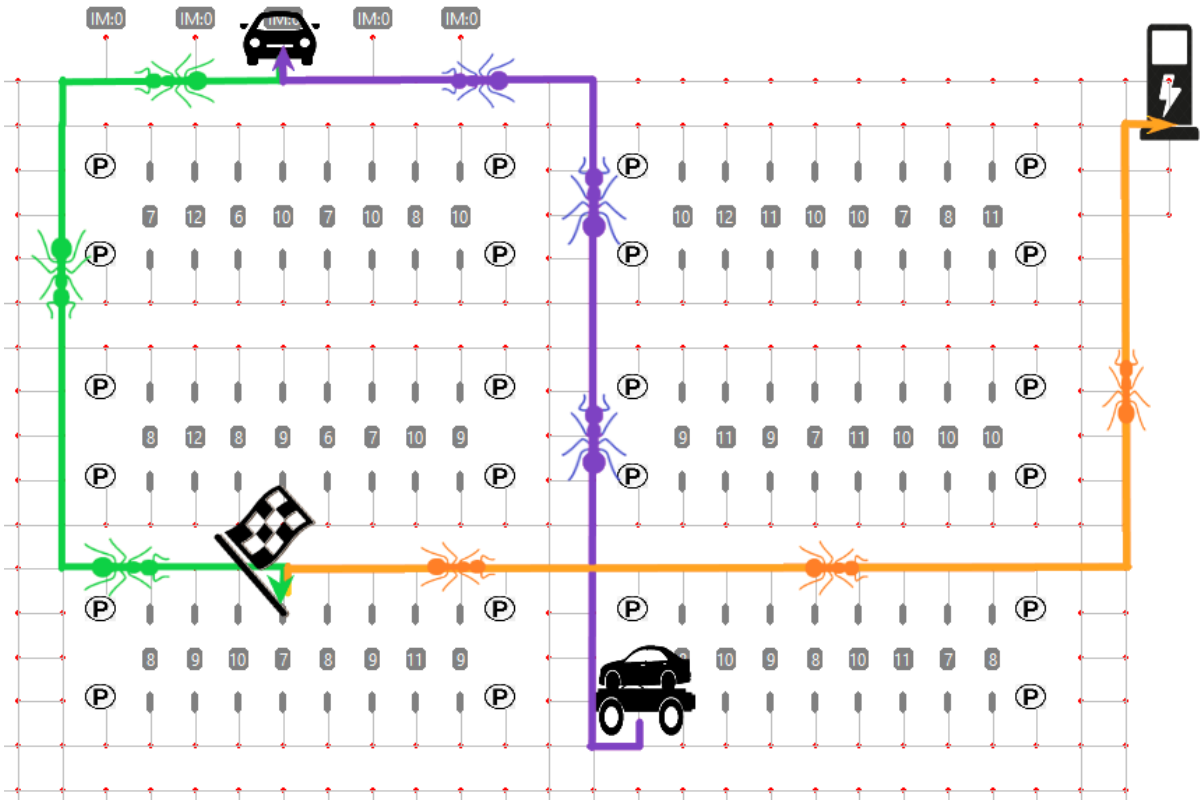
minder verplaatsingen, des te beter en dus des te lager de score. Het knooppunt met de laagste score wordt dan gebruikt om de taken toe te wijzen. De verwante taak van de knooppuntlaag wordt toegewezen aan de opslagagent en het ouderknooppunt wordt opgeroepen. Dit proces is te zien op lijn 12, en herhaalt zich tot het beginknooppunt bereikt is en alle taken aan een knooppunt zijn toegewezen. Deze methode zorgt ervoor dat er een minimaal aantal auto's moet worden verzet door de AGV's of dat het verplaatsen van die auto's een minimale kost met zich meebrengt. Als alle auto's moeten worden verplaatst, wordt aan elke kant de helft van de taken toegewezen.

4.7.2 Taakontdekking

Zoals eerder vermeld werd, worden de AGV's voor de initiële zoektocht naar een taak eerst verplaatst naar een mogelijke rustplaats. Dat is zo om tijdens het zoekproces zo weinig mogelijk andere AGV's te storen en om ervoor te zorgen dat AGV's niet langer blijven wachten op locaties waar andere AGV's moeten zijn. Dit taakontdekkingsproces voor de AGV's gebruikt een centrale server om alle mogelijk taken op te vragen. Het duurt namelijk lang en het is een netwerkintensieve opdracht om via haalbaarheidsmieren alle mogelijke taaklocaties op te zoeken. Nadat de AGV de lijst van mogelijke taken heeft opgevraagd, wordt er een lijst met taakexploratiemieren aangemaakt, meerdere voor elke taaklocatie. Dit taakontdekkingsproces wordt gecombineerd met het opzoeken van mogelijke paden. Op Figuur 4.3 wordt de werking van een enkel taakexploratieproces afgebeeld met slechts één beschouwd pad. Er is namelijk nood aan accurate en actuele informatie over de rijtijd tot een punt. Dat betekent dat de taakexploratiemieren zich op dezelfde manier als de padexploratiemieren verplaatsen over het netwerk en initieel dezelfde informatie verzamelen. Bij het aankomen bij een opslagagent met een taak wordt ofwel informatie verzameld voor de enkele-consument- of coalitiemethode. Als een taakexploratiemier aankomt bij een opslagagent die geen taak heeft of al een taakconsument heeft in de enkele-consumentmethode, of waarvan de langzaam groeiende coalitie al haar maximale groeigrootte heeft bereikt, wordt er een vlag gezet op die mier om dat aan te duiden en wordt die mier leeg teruggestuurd. Als uit het filterproces, besproken in de volgende onderdelen, blijkt dat een AGV geen enkele taak kan uitvoeren door zijn batterijniveau, start de AGV een oplaadcyclus.

Enkele-consumentmethode

In de enkele-consumentmethode wordt het oplossen van sequentiële onafhankelijke taken pragmatisch opgelost. Tijdens het taakontdekkingsproces betekent dat dat de AGV steeds enkel rekening houdt met de eerste beschikbare taak en niet met alle taken die op dat punt aanwezig zijn. Daarvoor wordt de eerstvolgende taak meegedeeld aan de taakexploratiemieren bij de ontvangst ervan. Als de taak een vasthoudtaak is, stuurt de opslagagent oplaadexploratiemieren op dezelfde manier als bij padexploratiemieren door naar het dichtstbijzijnde oplaadstation. Bij het ontvangen van een antwoord op de oplaadexploratiemieren worden beide mieren gecombineerd en teruggestuurd naar de AGV. Als de taak niet mogelijk is, wordt de taakexploratiemier gedropt. Bij een leveringstaak wordt er een padtaakexploratie naar het eindpunt van de taak gestuurd om informatie te verzamelen over de batterij-impact en om te controleren of een pad mogelijk is. De mier werkt opnieuw zoals een padexploratiemier. Die mieren worden dan gecombineerd



Figuur 4.3: Samenwerking van mieren voor taakexploratie. Taakexploratiemier en zijn pad wordt voorgesteld door de het paarse pad. De padtaakexploratie wordt voorgesteld door het groene pad en de de oplaadexploratiemieren worden voorgesteld door het oranje pad. Elke nieuwe mier wordt verstuurd bij ontvangst van de vorige mier bij zijn eindlocatie.

in de taakexploratiemier en direct teruggestuurd naar de AGV. Dat alles samen laat de AGV beslissingen nemen gebaseerd op lokale informatie over grote oppervlaktes en laat de AGV correct inschatten wat de kost voor de taak inhoudt. In beide gevallen filtert de AGV de taken die niet mogelijk zijn door zijn bereik. Bij vasthoudtaken wordt een waarde bijgerekend die de kleine verplaatsing voor een vasthoudtaak voorstelt.

Langzaam groeiende coalitievorming

We laten de coalitie aan een vaste en trage snelheid vormen door het aantal benaderende AGV's te beperken. Hierdoor zullen de wachttijden van de AGV's aan de opslagagenten worden beperkt. Met DMAS kunnen de AGV's inschatten wanneer ze op een bepaalde locatie zouden aankomen. Die schatting kan zowel te vroeg of te laat zijn. Dat betekent echter dat er in het langzaam groeiende deel van de coalitiemethode geen zekerheid is over welke AGV als eerste zal aankomen. Het kan dat een AGV een veel beter pad vindt, terwijl een andere AGV een aanzienlijke vertraging oploopt. Elke AGV die aankomt op een punt moet dus alle taken kunnen uitvoeren op dat punt omdat het niet zeker is welke taak de AGV zal toegewezen krijgen. Als in de coalitievormingsmethode een opslagagent een taakexploratiemier ontvangt, wordt er voor alle leveringstaken een padtaakexploratie gestuurd. Het eindpunt dat die padtaakexploratie ontvangt, stuurt naar analogie met

de enkele-consumentmethode dan opnieuw een oplaadexploratiemier om alle informatie vervolgens te verzamelen in één enkele mier. De AGV beslist of er aan een coalitie kan deelgenomen worden op basis van de mogelijke batterij-impact van de grootste taak. De coalities waar de AGV's niet aan kunnen deelnemen, worden weggefilterd. Aangezien een vasthoudtaak een minimale verplaatsing en wachttijd bevat, wordt die niet in rekening gebracht tijdens het filterproces als alle andere taken kunnen worden uitgevoerd. Het is wel belangrijk om te vermelden dat de coalitiemethode ervan uitgaat dat er met homogene taakverzamelingen en homogene AGV-verzameling wordt gewerkt. Elke AGV kan namelijk elke taak uitvoeren. Als dat niet het geval is, moet de coalitievormingsmethode anders verlopen. Door de schattingen van het aankomen van de AGV's, is er geen zekerheid dat als de derde taak enkel door de derde AGV in de coalitie kan worden uitgevoerd, die derde AGV ook als derde aankomt. Het is mogelijk om de enkele coalitie te splitsen in meerdere kleine coalities om dat probleem deels op te lossen.

Vastzetten van de taakintentie

In beide voornoemde gevallen is er een lijst verzameld van mogelijke taken of coalities die de AGV wil uitvoeren. De AGV stuurt dan een taakintentiemier naar de locatie waarop de taak zich bevindt. Als de opslagagent nog steeds geen consument heeft in de enkele-consumentmethode of er een plaats is in het langzaam groeiende deel van zijn coalitie in de coalitiemethode, dan ontvangt de AGV een taakintentiemier met een geaccepteerde vlag en extra bijgewerkte informatie. De AGV bewaart die informatie om te helpen beslissen over mogelijke betere taken. Doordat de taakexploratiemieren ook al de paden hebben bepaald en het mogelijks beste pad hebben gevonden, wordt er over het beste pad ook al een padintentiemier gestuurd. Daardoor moeten de AGV's na het beslissen over een taak niet nogmaals een pad zoeken. Als de AGV niet geaccepteerd is door de opslagagent, ontvangt de AGV de taakintentiemier zonder de geaccepteerde vlag. Daarna probeert de AGV de volgende beste taak aan te nemen. Dit proces herhaalt zich tot er ofwel een taak gevonden is, ofwel een termijn is verstreken. Als de termijn is verstreken, wordt de takenlijst opnieuw opgehaald en worden er nieuwe taakexploratiemieren verstuurd. Als voor het verstrijken van de tijdspanne blijkt dat alle mogelijke taken bezet zijn, gaat de AGV op in een wachtperiode. Doordat de AGV enkel naar taken zal zoeken op een mogelijke rustplaats, heeft het wachten geen impact op de schema's van de andere AGV's. Door de vluchtige aard van de feromonen, moet een AGV regelmatig opnieuw taakintentiemieren versturen naar het eindpunt en verkrijgen ze steeds geüpdatete informatie, wat opnieuw helpt bij het selecteren van mogelijke betere taken. Bij het vastlopen van de AGV of bij het verkiezen van een andere taak, wordt de taak door de feromoneninfrastructuur opnieuw vrijgegeven.

4.7.3 Taak besluitsvormingproces

In dit deel staan we stil bij de elementen die helpen beslissen welke taak de AGV aanneemt. Doordat de taken DARPWD voorstellen, heeft elke taak een leveringstijdslot, die loopt vanaf het moment dat de taak verschijnt op het wegennetwerk tot een bepaalde eindtijd. Om taken interessanter te maken als ze dichter bij hun eindtijden komen, heeft de resterende tijd van het leveringstijdslot een prioriteit die in bepaalde intervallen stijgt, in deze situatie elke vijftien minuten. Het is ook mogelijk om de intervallen dynamischer

te maken en ze te baseren op enerzijds de eventuele nog op te pikken taken die zich voor de taak in kwestie bevinden en anderzijds op de duur van het oppikken die voor die taak verwacht wordt. Een taak heeft daarnaast ook altijd een eigen prioriteit die voorstelt hoe dringend de taak is, naast mogelijke tijdsbeperkingen. Dat kan gebruikt worden om taken die absoluut niet te laat mogen zijn, nog verder naar boven te duwen tijdens het selectieproces. Een probleem met de prioriteiten is het mogelijks optreden van prioriteitsverhogingen. Het is namelijk mogelijk om steeds hogere prioriteiten toe te wijzen aan taken, wat ervoor zorgt dat een hoge taakprioriteit op tijdstip 1 mogelijks een lage is tegen tijdstip 10. Om dit probleem tegen te gaan, wordt er aan taken enkel een eigen prioriteit toegewezen tussen 1 en 5. In het geval van vasthoudtaken wordt de prioriteit van de vasthoudtaken gelijkgesteld aan de prioriteit van de leveringstaak waarvoor de vasthoudtaken werden aangemaakt. Omdat die vasthoudtaken enkel aangemaakt zijn voor het uitvoeren van de leveringstaken, zullen die altijd een prioriteit hebben van 0 indien ze niet worden gelijkgesteld. Dat wil ook zeggen dat de prioriteit van de leveringstaken niet correct zou worden doorgerekend naar de belangrijkheid van het uitvoeren van de vasthoudtaak. Op dezelfde manier wordt de tijdprioriteit van de leveringstaak ook doorgegeven aan de vasthoudtaken. Daarnaast wordt in zowel de enkele-consument- als de coalitiemethode ook rekening gehouden met het aantal wachtende AGV's. De wachtende AGV's voeren een vasthoudtaak uit en bevinden zich dus op een rustplaats met een auto. Die AGV's kunnen tijdens de vasthoudtaak geen andere taken uitvoeren. Dat betekent dat het aantal AGV's die een vasthoudtaak uitvoeren voor een optimale oplossing altijd zo klein mogelijk moet zijn. Om de prioriteit van een onderling afhankelijke taak voor te stellen, worden bij de door de tijd beïnvloede en de eigen prioriteit ook het aantal al opgepikte taken of AGV's die een vasthoudtaak uitvoeren voor die specifieke pickup tot de prioriteit gerekend.

Om een gewogen som voor de prioriteiten voor te stellen, wordt er ook telkens een factor toegevoegd die de prioriteiten belangrijker of minder belangrijk kan maken. Daaruit volgt dan de gewogen som voor de prioriteit van een taak bij enkele-consumentmethode:

$$\begin{aligned} Priority = {} & PickedUpTaskFactor * PickedUpTasks \\ & + TaskPriorityFactor * TaskPriority \\ & + TimeFactor * TimePriority \end{aligned} \tag{4.1}$$

waar de prioriteiten voor leveringstaken van zichzelf zijn en waar de prioriteiten van de vasthoudtaken doorgegeven zijn door de leveringstaken waarvoor ze gecreëerd zijn. Het proces verloopt anders bij het zoeken naar coalities. Doordat de prioriteit van een coalitie afhangt van de hoogste taak in de coalitie, worden alle taakprioriteiten aan de taakexploratiemier meegegeven. Daarnaast wordt ook het aantal geplande AGV's die onderdeel van de coalitie zijn, meegeteld. Als er een coalitie wordt gevormd, moet ze zo snel mogelijk alle taken uitvoeren. In dit geval is de gewogen som voor prioriteiten de volgende:

$$\begin{aligned} Priority = {} & PickedUpTaskFactor * PickedUpTasks \\ & + TaskPriorityFactor * HighestTaskPriority \\ & + TimeFactor * HighestTimePriority \\ & + CoalitionsSizeFactor * CoalitionSize \end{aligned} \tag{4.2}$$

Het aantal AGV's die op weg zijn naar de coalitie (coalitieSize), en dus deel zijn van het langzaam groeiend deel van de coalitie, en het aantal AGV's die een vasthoudtaak uit-

voeren (pickedUpTask), wordt gesplist om meer controle te krijgen over de gewogen som. Het is ook mogelijk om tijdens het beslissingsproces een gewogen som te maken tussen de rijtijd en de prioriteit van een taak. In de implementatie wordt een gelaagd filterproces gebruikt, waarop eerst de hoogste prioriteiten verkozen worden. Daarna worden de taken met gelijkaardige prioriteiten verder gesorteerd op de mogelijke rijtijd naar de verkozen taak.

4.7.4 Zoeken naar betere taken door AGV's

Een AGV zal tijdens het verplaatsen naar een gekozen taak ook constant taakexploratiemieren sturen naar andere mogelijke taaklocaties. De taakexploratiemieren zullen daarna aankomen bij een knooppunt waar een taak aanwezig is. Bij het aankomen op dat knooppunt, zullen er opnieuw taakexploratiemieren en daarna oplaadexploratiemieren worden gestuurd. Dat gebeurt op dezelfde manier als het zoeken naar een mogelijke taak maar er wordt slechts één pad in aanmerking genomen in plaats van meerdere zoals bij de initiële zoektocht. Vervolgens worden de mieren opnieuw samengevoegd en naar de AGV gestuurd. De verkregen informatie is in dit geval niet volledig correct omdat de huidige locatie van de AGV en de locatie wanneer de mieren verstuurd zijn, gewijzigd is. Een AGV kan namelijk bewegen terwijl hij op antwoord wacht van de gestuurde mieren en die mieren vragen ook wat tijd om rond het netwerk te worden verstuurd. Bij het beslissen of een andere taak beter is, moet er met die aanpassing rekening gehouden worden. Het sturen van meerdere mieren naar dezelfde locatie om met de verkeersdrukte en aankomsttijd te optimaliseren, is hier niet meer van toepassing.

De AGV filtert agressief in de nieuwe mogelijke taken die zich niet per se in het bereik van de AGV bevinden. De afstand tussen de locatie van waarop de AGV de mieren verstuurde en het laadpunt van de taak, wordt samengevoegd met de afstand tussen het laad- en lospunt en de afstand tussen het lospunt en het dichtste oplaadstation. Die volledige afstand wordt vervolgens vermenigvuldigd met een factor. Als de uitkomst niet haalbaar is voor de AGV, wordt de taak verworpen. De factor, ingesteld op $1,5$, is toegevoegd om het mogelijke verschil op te lossen tussen de werkelijke locatie van de AGV en de locatie waarop de mieren werden verstuurd. De factor kan worden aangepast op basis van de grootte van het wegennet en hoe rampzalig mogelijke lege batterijen zijn voor een AGV. De niet gefilterde taken worden dan gesorteerd op basis van de prioriteit die de taak heeft en bij dezelfde prioriteit de mogelijke aankomsttijd. Enkel taken met een hogere prioriteit dan die van de huidige taak komen in aanmerking om een mogelijke betere taak te zijn. Als de set van taken met een hogere prioriteit leeg is, eindigt het proces hier. Door het feit dat de AGV al verplaatst kan zijn, kan er niet worden ingeschat of een taak met dezelfde prioriteit zich dichterbij bevindt. Als de taak zich dichterbij zou bevinden, zou hij namelijk beter zijn dan de huidige taak. Dat betekent ook dat het sorteren van de taken op aankomsttijd niet volledig correct is met de huidige locatie van de AGV. Er kan eventueel worden gebruikgemaakt van een gewogen som die zowel de aankomsttijd als een mogelijke afstand van de huidige locatie en het oppikknooppunt gebruikt om het probleem te verkleinen. Het is echter onvermijdelijk dat de informatie niet volledig overeenstemt met de werkelijkheid.

De meest prioritaire taak met de dichtste rijtijd wordt na het filterproces gecontacteerd

en als hij nog beschikbaar is, wordt hij door de AGV geselecteerd. Pas nadat de taak geaccepteerd is, laat de AGV de oude taak vallen. Daarna wordt er een nieuw padbepalingsproces gestart om een pad te zoeken en de nieuwe taak uit te voeren. Als de taak niet meer beschikbaar is, wordt de taak uit de set van mogelijke taken verwijderd en wordt de volgende mogelijk taak gecontacteerd. Als de set leeg is en er geen betere taak beschikbaar is, stopt het proces. Wel belangrijk om te vermelden is dat de AGV , in het geval van coalitievorming, een eerstgekozen taak in prioriteit heeft doen stijgen door hem aan te nemen. Voor die taak steeg het aantal AGV's dat zich heeft aangemeld om de coalitie op te lossen, wat de taak interessanter maakt voor andere AGV's en dus ook voor zichzelf. Het proces waarbij er naar betere taken wordt gezocht, betekent dus vooral dat er naar taken met hogere prioriteiten wordt gezocht. Het verschijnen van een taak die meer prioritair is, door bijvoorbeeld wegvallen van een AGV in een coalitie of een taak die zeer snel moet worden volbracht, wordt door het zoeken naar betere taken snel opgevangen.

4.7.5 Uitvoeren van taken

In dit onderdeel wordt beschreven hoe de AGV's de toegewezen taken uitvoeren. De AGV's zullen de taken uitvoeren na het oppikken ervan. Tijdens dit proces zullen AGV's ook niet zoeken of een andere betere taak beschikbaar is. Zoals eerder uitgelegd, zijn er twee soorten takenverzamelingen, namelijk de batch-jobs en een specifieke pickup. De batch-jobs bestaan uit een verzameling van taken waar alle taken leveringstaken zijn. Dat wil zeggen dat elke taak naar een specifieke andere opslagplaats moet worden verplaatst. In de casestudy treedt dat op na het lossen van een schip of trein waarbij alle auto's in het systeem moeten worden gebracht of waar een workshop een volledige lading auto's vrijgeeft aan het systeem. De andere taakverzamelingen zijn de specifieke pickups en de tijdelijke verplaatsingen die daarvoor nodig zijn. Het gaat daarbij om een verzameling die bestaat uit leverings- en vasthoudtaken. Beide taken vragen een licht verschillend gedrag van de AGV's om ze correct uit te voeren.

Batch-jobs

In batch-jobs krijgt elke taak een specifieke eindlocatie waarop de auto moet worden afgeleverd. Dat versimpelt het probleem tot een padbepalingsprobleem na het oppikken van de taak. Als een AGV eenmaal een leveringstaak heeft opgenomen, en dus de auto heeft geladen, zal een reeks padexploratiemieren worden verstuurd op dezelfde manier als hoe padbepaling gebeurt zonder aanwezige taak. Eenmaal aangekomen, zal de AGV de auto op de nieuwe opslagagent lossen en wordt de taak beëindigd. Na het lossen wordt de AGV wel verplicht om zich naar een rustplaats te begeven om een nieuwe mogelijke taak te zoeken. Het verplaatsen van AGV's naar een rustplaats en de werkwijze ervan werd al gedetailleerd besproken in subsectie 4.6.3.

Specifieke pickup

Als er een specifieke pickup wordt aangevraagd en de auto als eerste in de rij staat en er dus geen sprake is van een vasthoudtaak, dan wordt er een reeks padexploratiemieren verstuurd en wordt er een pad bepaald. Als er echter een vasthoudtaak wordt toegewezen aan een AGV, dan zoekt die een plaats op in het systeem waar hij tijdelijk uit de weg

staat. De rustplaats wordt opgezocht met vind-rustplaatsmieren, zoals besproken in de deadlock solving methode in subsectie 4.6.3. De AGV verplaatst zich vervolgens naar de gevonden rustplaats en wacht op een bericht van de opslagagent die aangeeft dat de AGV kan terugkeren naar de opslagagent om de taak en auto terug te plaatsen. Als de gekozen rustplaats tijdens de wachtperiode haar restricties om een rustplaats te zijn verliest, zoekt de AGV een nieuwe rustplaats en wacht hij daar. Pas als de auto terug op zijn originele plaats staat, is de taak afgerond. Als een volgende AGV daarna een leveringstaak oppikt, wordt die uiteraard als leveringstaak uitgevoerd. Dat betekent ook dat de opslagagenten bijhouden welke AGV met welke auto en taak bezig is. Als alle leveringstaken afgerond zijn, roept de opslagagent de AGV's één voor één terug in de omgekeerde volgorde van pickups zodat de rij van auto's terug in hun oorspronkelijke volgorde staan. Die volgorde kan eenvoudig worden aangepast door extra restricties op het terugroepingsproces toe te voegen die de volgorde wijzigt. Als een AGV de auto heeft teruggeplaatst en de taak dus is afgewerkt, verplaatst de AGV zich opnieuw naar een dichtbijgelegen vrije rustplaats zodat de volgende AGV zich vlot naar de opslagagent kan begeven zonder te moeten wachten tot de eerste AGV een mogelijke taaklocatie gevonden heeft en het padbepalingsalgoritme heeft uitgevoerd.

4.8 Bereikgebonden zoeken

Het hoofddoel van de taakallocatie met DMAS is om een gedistribueerd en schaalbaar taakallocatie-algoritme te implementeren. De voorgestelde oplossing omvat echter nog een probleem dat de schaalbaarheid vermoeilijkt, namelijk de hoeveelheid mieren. Bij een grotere hoeveelheid AGV's, die een grotere hoeveelheid taken wou verzorgen, trad er een probleem op omtrent de performantie. Er werden namelijk veel mieren verstuurd omdat er meerdere mieren naar elke taak werden gestuurd, gebaseerd op de verschillende paden. Die mieren werden elk nog vermenigvuldigd door het bevestigen van bestaande paden. De AGV's stuurden namelijk naar alle taken om de beste taak te vinden, zodat elke AGV de best mogelijke beslissing nam maar dat veroorzaakte een grote hoeveelheid netwerkverkeer. Om dit op te lossen, werd er voor de taken een bereikgebonden zoekmethode ingesteld. Er werd een minimum aantal taken geïntroduceerd die een AGV moet contacteren om een taak te zoeken. Een AGV berekent per taak hoe ver de laadlocatie gelegen is. Als de afstand zich onder een bepaalde waarde ρ bevindt, wordt de taak opgenomen in de lijst van mogelijke taken. Als het minimumaantal taken niet wordt bereikt, wordt de afstand ρ verdubbeld. Die waarde ρ wordt zo vergroot tot het minimumaantal taken wordt bereikt, of tot alle taken zich in de lijst van mogelijke taken bevinden. Op het einde van de iteraties worden alle taken waarvan de startlocatie zich op minder dan afstand ρ bevindt, als mogelijke taken beschouwd en worden ze door de AGV gecontacteerd via de taakexploratiemieren. De AGV's gaan er op die manier vanuit dat taken die zich verder bevinden, opgepikt zullen worden door een andere AGV. Dat is ook logisch omdat taken die zich ver van een AGV bevinden vaak niet de beste toewijzingen zijn en dat zelfs als ze prioritair zijn, de grote afstand betekent dat een andere AGV zich dichterbij kan bevinden.

Hoofdstuk 5

Resultaten en bespreking

In dit hoofdstuk zullen we eerst de opstelling van het experiment bekijken en de keuzes die zijn gemaakt op vlak van de taken en de abstracties. Vervolgens worden er enkele oplossingscriteria besproken. Daarna volgen enkele experimenten en hun resultaten met een korte bespreking en tot slot wordt dit hoofdstuk afgesloten met de conclusies die uit die experimenten volgen.

5.1 Design

De experimenten zijn gebaseerd op de casestudy. Vanwege de nodige abstracties door het ontbreken van bepaalde waarden, besloten we slechts een deel van de volledige site te modelleren. Door de abstracties kunnen de experimenten geen doorslaggevende besluiten vormen over de ideale oplossing en met welke heuristische waarden de site de beste resultaten zal behalen. Zoals te zien is in Figuur 5.1, wordt een deel van de site gemodelleerd als een rechthoekig wegennetwerk. De meeste parkeerplaatsen bestaan uit parkeerlanes waarvan enkele slechts bereikbaar zijn via één zijde. Die eenzijdige parkeerlanes stellen het visgratenpatroon voor en zijn aanwezig aan de buitenkanten van de site. Aan één kant van de site komen, via importknooppunten, enkel auto's aan. Aan de andere kant van de site bevinden zich de exportknopen die laadplaatsen voorstellen. Vanop die exportknopen worden de auto's van de site verwijderd. Elke rij parkeerlanes is tien stroken breed, waarna een tweerichtingsstraat ze afsplitst van een nieuwe rij parkeerplaatsen. Elke rij parkeerlanes bevat ook twee rustplaatsen waar de AGV's zowel kunnen wachten als kunnen zoeken naar nieuwe acties. Verticaal zijn er tien rijen parkeerplaatsen. Horizontaal wordt de site opgedeeld in de verschillende rijen parkeerplaatsen door verschillende tweerichtingsstraten die verbonden zijn met de ingangen van de parkeerplaatsen. In totaal bevat de site negen rijen verticale parkeerplaatsen. Dit wil zeggen dat er 72 parkeerlanesblokken aanwezig zijn, waarin meer dan 1600 auto's kunnen worden opgeslagen. Er zijn ook zes oplaadstations aanwezig die zich in groepen van drie dicht bij de importknooppunten bevinden. Verticaal bestaan de wegen uit honderd knooppunten met een afstand van twee meter tussen elke twee verbonden knooppunten. Horizontaal bestaan de wegen uit vijftig knooppunten met opnieuw een afstand van twee meter tussen elke verbonden knoop. Sommige knooppuntverbindingen stellen echter wel afstanden voor die langer zijn dan die twee meter. Door de vaste snelheid van de AGV worden die verbindingen ook als twee meter voorgesteld, hoewel de AGV's zich op de echte site sneller over die speciale verbindingen verplaatsen. Die verbindingen bevinden zich vooral parallel met de parkeer-

lanes en hebben geen kruispunten. Door dit gebrek aan kruispunten zouden de AGV's zich ook sneller kunnen verplaatsen op de werkelijke site in vergelijking met straten met veel kruispunten, zoals tussen de rijen parkeerlanes.

De gemaakte abstracties, zoals eerder vermeld in hoofdstuk 4, hebben voornamelijk betrekking op de capaciteiten van de AGV. Onder andere de snelheid, de versnelling, het bereik, de breedte en de lengte zijn nog allemaal onbekend. Om toch met de AGV's te kunnen werken, werden de volgende abstracties gemaakt. Ten eerste neemt een AGV altijd één infrastructuuragent in. Ten tweede kan een AGV een gemiddelde snelheid van twee m/s halen op straten met veel kruispunten, zoals tussen de rijen parkeerstroken. Op andere banen waar minder kruispunten aanwezig zijn, gaan we ervan uit dat een AGV een gemiddelde snelheid van vier m/s haalt. Ten derde gaan we ervan uit dat een volgeladen AGV een bereik heeft van een twintigtal kilometer. Dat betekent dat een volgeladen AGV tot honderdmaal de volledige breedte van de verkleinde site kan afleggen. De AGV's laden hun batterijen ook op via een versimpelde lineaire functie. We gaan ervan uit dat het volledige opladen van een batterij één uur duurt. Als de specificaties van de AGV's bekend zijn, kunnen er realistische cijfers worden gebruikt.

Bij het starten van de simulatie is de parking voor 75 procent volgeladen. Elke parkeerlane, die tot twaalf auto's kan bevatten, bevat tussen de zes en twaalf auto's uniform verbeeld. De parkeerplaatsen met visgraatpatroon kunnen maximaal zes auto's tellen. De import- en exportknooppunten, die zich elk respectievelijk bovenaan en onderaan de graaf bevinden, afgebeeld in Figuur 5.1, zijn respectievelijk vol of leeg. Als er een auto naar een exportknooppunt wordt gebracht, wordt een nieuwe auto toegevoegd in een willekeurig importknooppunt. Op die manier blijft de site evenveel auto's bevatten.

Om taken te genereren, wordt er een uniforme willekeurige waarde tussen 0 en 1 opgevraagd. Als die waarde kleiner of gelijk is aan 0.5, wordt er een batchjob gegenereerd, startende vanuit een willekeurig importknooppunt. De taken van die batchjob kunnen als eindpunt een exportknooppunt of een opslagplaats hebben. Als de waarde groter is dan 0.5, wordt er een specifieke pickup aangevraagd vanuit een opslagpunt. Die pickup neemt uniform verdeeld één tot vier auto's mee uit de locatie en bestempelt ze als pickup met als eindbestemming een exportknooppunt. Die werkwijze laat toe een verdeeld aantal taken aan te maken van beide soorten. De toegevoegde taken hebben allemaal hun eigen prioriteit en eindtijdstip. Dat eindtijdstip is willekeurig bepaald en bedraagt ofwel een kwartier, een halfuur, drie kwartier of een uur meer dan het begintijdstip. De taken zijn niet gegarandeerd mogelijk om uit te voeren voor hun eindtijdstip bereikt is. Dit laat toe om te bekijken hoe de DMAS-configuratie zich gedraagt als er plots een zeer prioritaire taak op het netwerk verschijnt. Elke seconde kan er een taak worden toegevoegd aan het netwerk, gebaseerd op een kans. Als er al twintig leveringstaken aanwezig zijn op het netwerk wordt dat tegengehouden. Opdat de AGV's te allen tijde taken zouden hebben om uit te voeren, is er altijd een minimumaantal van vijftien leveringstaken aanwezig op het netwerk.

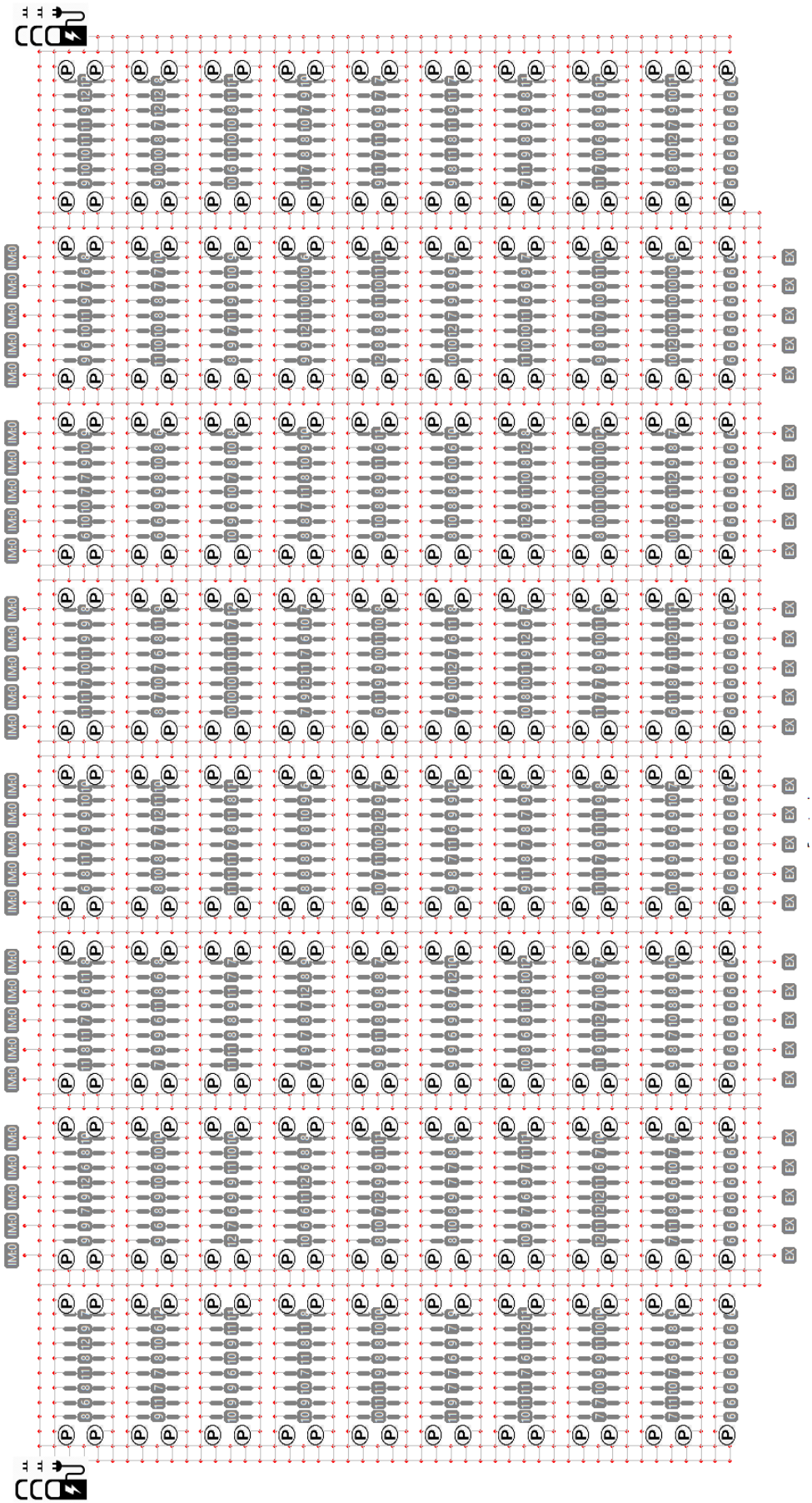
Elk experiment wordt twintig keer uitgevoerd in vier reeksen van vijf. De simulaties zijn van dynamische aard door de veranderingen die voorvallen tijdens het uitvoeren van de simulatie. Daarnaast is de simulatie van stochastische aard door de kansen geïntroduceerd

in hoofdstuk 4. Er werd ook geen standaardwaarde meegegeven om de willekeurige waarden te bepalen. De variabelen van de simulatie worden vastgelegd bij het begin van de simulatie waardoor ze van discrete aard is. De simulatie gebruikt het TimeModel van RinSim om zo een dag van acht uur te simuleren voordat ze afgerond wordt. Na die acht uur worden de niet afgeronde taken niet meegeteld bij het totale aantal uitgevoerde taken. Via het TimeModel van RinSim wordt de simulatie opgedeeld in ticks van 250 milliseconden. Elke agent krijgt een kans om tijdens zo'n tick een actie uit te voeren. Dat betekent echter dat de mieren slechts om de 250ms door de agent kunnen worden verstuurd. Het uitvoeren van de berekeningen voor de mieren duurt slechts enkele milliseconden. Het rondsturen van een mier in die opstelling betekent dat een mier, hoewel ze slechts enkele milliseconden nodig had voor de uitvoering en klein is qua grootte, enkele minuten nodig had om een pad uit te voeren. Om dat te versnellen, werden de mieren pragmatisch doorgestuurd over de agenten met vertragingen na het bereiken van de eindagent, in plaats van ze door te sturen per tick van de simulatie.

In de standaardopstelling gaan we ervan uit dan een agent de hiervoor vermelde snelheid kan aanhouden en dertig seconden nodig heeft voor het oppikken van een auto. Er zijn ook telkens twintig AGV's aanwezig op het wegennet met een batterijpercentage tussen twintig en honderd percent. De andere standaardwaarden voor de experimenten zijn terug te vinden in Tabel 5.1.

Tabel 5.1: Standaardwaarden voor de experimenten.

Type	Waarde
Alternatieve paden voor exploratie	3
Tijd tussen reservaties	10 seconden
CoalitieSizeFactor	2
PickedUpTaskFactor	4
TijdFactor	1
TijdPrioriteit 60+ Minuten	2
TijdPrioriteit 30+ Minuten	4
TijdPrioriteit 15+ Minuten	8
TijdPrioriteit 0+ Minuten	16
Laadtijd	30 seconden



Figuur 5.1: Versimpelde site van het experiment.

5.2 Evaluatiecriteria

Om de invloed van de verschillende methodes en aanpassingen op de experimenten in te schatten, worden hier de interessantste criteria die we gebruiken om ze te vergelijken, opgesomd. Het eerste en voornaamste criterium is de hoeveelheid uitgevoerde taken, dat de doorvoer voorstelt die de AGV's bereiken. Een optimale oplossing zal namelijk meer taken uitvoeren dan een niet optimale oplossing in dezelfde tijd. Daarnaast wordt ook de totale leveringstraagheid bekeken die een experiment opleverde. Die leveringstraagheid komt overeen met de leveringstraagheid voorgesteld in subsectie 2.1.3. Niet alle taken kunnen worden uitgevoerd in de toegelaten tijd maar in een ideale situatie blijft de leveringstraagheid zo laag mogelijk. De hoeveelheid deadlocks die optreden en de tijdsduur om ze op te lossen zijn ook interessant om de performantie van het systeem in te schatten. In de beste situatie worden deadlocks zoveel mogelijk vermeden. Door de mogelijke queues die voorkomen als meerdere AGV's zich aan een parkeerstrook bevinden, zijn de deadlocks die door die queus gecreëerd werden, moeilijk te vermijden. Ten slotte wordt ook bekeken hoe vaak een AGV van pad of taak verandert doordat hij een beter pad vond. Dit laat ons toe om in te schatten hoe de AGV's hun intenties herevalueren en in welke mate. Om de gemiddelden te vergelijken, wordt er ook gebruikt gemaakt van Median Absolute Deviation (MAD) om de spreiding van de data te bekijken. Door de aanwezigheid van uitschieters in de data, vooral op vlak van deadlocks, gaf de vaker gebruikte standaarddeviatie een slechte weergave. De uitschieters in de data, waar MAD meer resistent tegen is, en de niet normaal gedistribueerde natuur van de experimentresultaten raden allebei het gebruik van de vaker gebruikte standaarddeviatie af.

Voor de experimenten definiëren we vijf verschillende methodes. Als eerste hebben we de enkele-consumentmethode, die in de volgende paragrafen wordt vermeld als 1-CONS. Dit stelt de enkele-consumentmethode voor zoals besproken in de voorgaande hoofdstukken. Als tweede hebben we de coalitiemethode, die verder wordt opgesplitst in vier submethodes. De eerste drie nemen de vorm van x-coalities aan. Die x stelt de maximale grootte van het groeiende deel van een coalitie voor. In het geval van 2-coalitie wordt toegelaten dat het groeiende deel van de coalitie, dus het aantal nieuwe benaderende AGV's, zich aan een maximale grootte van twee moet houden. Bij de laatste submethode spreken we van inf-coalitie. In dit geval laten we de coalitie onbeperkt groeien tot evenveel AGV's de taaklocatie benaderen als de taaklocatie nodig heeft om al haar taken uit te voeren en dus geen beperkingen oplegt over hoe snel die coalitie mag groeien.

5.3 Invloed van coalitievorming

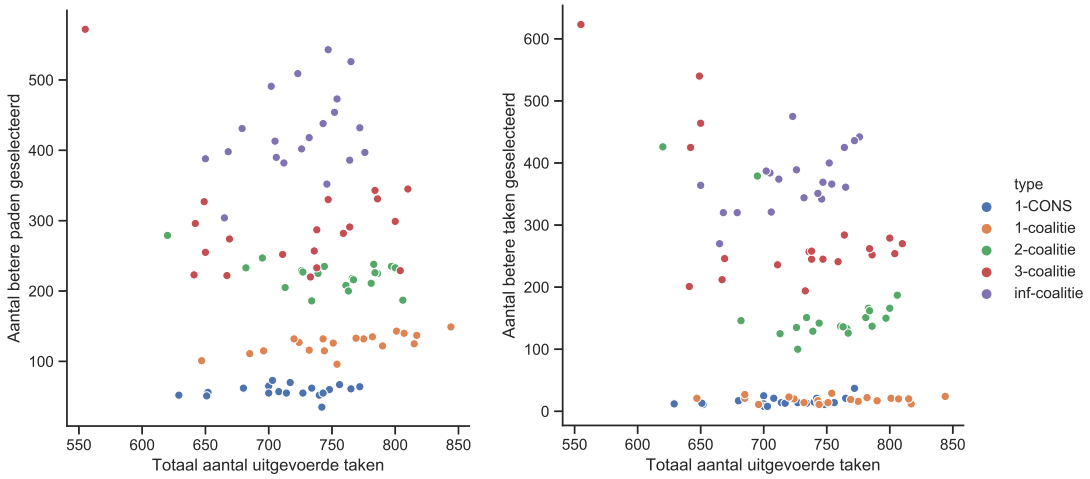
In dit experiment ligt de focus op de verschillen tussen de verschillende oplossingsmethodes. Dit experiment gebruikt alle standaardwaarden die zijn vermeld in Tabel 5.1. De AGV's behouden ook hun vermelde gemiddelde snelheid. Tabel 5.2 toont voor de verschillende methodes hoeveel taken zij gemiddeld verwerkten in de acht uur durende simulatie. Voor elke methode werden, zoals eerder vermeld, twintig iteraties uitgevoerd. Uit Tabel 5.2 is af te leiden dat slechts een deel van de coalitiemethodes het beter doet dan de enkele-consumentmethode. In het beste geval van 1-coalitie is slechts ongeveer een viertal percent meer performant dan de enkele-consumentmethode. Opmerkelijk is wel de

Tabel 5.2: Experimentresultaten: invloed coalitievorming op taken.

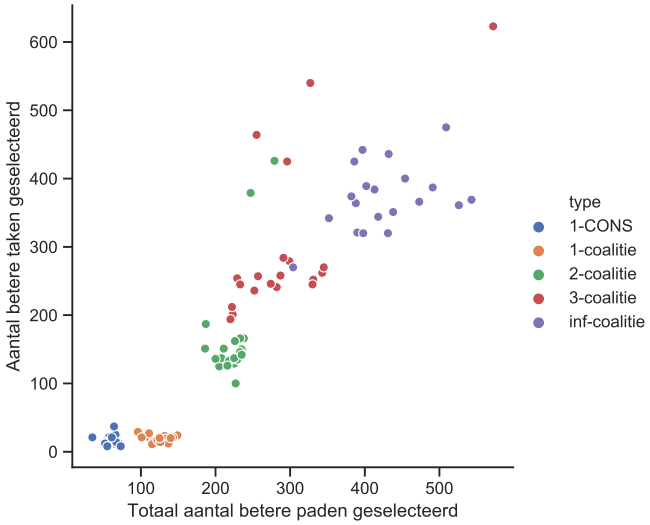
	Gemiddeld aantal uitgevoerde taken	MAD
1-CONS	724.35	30.92
1-coalitie	754.05	41.36
2-coalitie	748.70	35.83
3-coalitie	719.15	56.92
inf-coalitie	724.35	30.92

kloof tussen de 1- en 2-coalitiemethodes en de 3- en inf-coalitiemethodes. Waar de 1- en 2-coalitiemethodes betere resultaten behalen dan de enkele-consumentmethode is dit voor de 3- en inf-methode niet het geval, die waren allebei ongeveer even performant als de enkele-consumentmethode. De reden waarom de 3-coalitiemethode het slechtste scoort, kan deels worden toegeschreven aan meer variatie in de experimenten door uitschieters, wat ook aangeduid wordt door de hogere MAD die de 3-coalitiemethode heeft versus de andere methodes. Het aanmaken van te grote coalities blijkt dus onproductief te zijn. Dit is toe te schrijven aan het feit dat een AGV die wacht om een pakket op te pikken, op hetzelfde moment geen andere taak kan uitvoeren. In de 3- en inf-coalitiemethodes zullen er dus meer AGV's wachten om een taak op te pikken. In gevallen met een hogere laadtijd maar met een gelijkaardige rijtijd als in deze situatie, kunnen de wachttijden van de AGV's nog meer oplopen. In gevallen met een langere rijtijd is het misschien nuttiger om meer AGV's op tijd te laten vertrekken omdat de verhouding tussen rijden en laden op die manier daalt. Beide gevallen worden verder bekeken in sectie 5.4.

In Figuur 5.2 wordt de verhouding tussen de totale hoeveelheid taken, de methodes en het aantal betere paden of taken afgebeeld. Zowel de linker- als rechtergrafiek tonen een sterk gelaagde structuur aan, waar elke methode zijn eigen laag inneemt. In het geval van betere paden is er een gedeeltelijke overlapping tussen de 2- en 3-coalitiemethodes. In het geval van betere taken is er een overlap aanwezig tussen verschillende methodes maar een duidelijke splitsing tussen de 1-CONS- en 1-coalitiemethodes in vergelijking met de andere methodes. In Figuur 5.3 is er een duidelijke clustering te zien in de verschillende methodes met enkele uitschieters. Dat wijst op een verband tussen de optimalisatie van de paden en de optimalisatie van de taken. Elk pad kan verder worden geoptimaliseerd tijdens het uitvoeren ervan. In het geval waar vaker van taak wordt gewisseld, zullen er meer paden moeten worden geoptimaliseerd. Het kiezen van een nieuwe taak, en dus ook een nieuw pad, betekent dat het geoptimaliseerde pad vervangen wordt door een mogelijks niet geoptimaliseerd pad, waarop weer betere paden kunnen worden geselecteerd. De hogere coalitiegrootte van de 2-,3- en inf-coalitiemethodes voegen meer taken toe op de omgeving waarvoor een AGV zich kan aanmelden. Door de grotere hoeveelheid beschikbare taken, kunnen AGV's vaker een betere taak vinden. De prioriteiten worden ook verhoogd naargelang de grootte van de coalities en de al opgepikte taken, waardoor een eerst niet-prioritaire taak plots voor een AGV wel prioritair kan worden.



Figuur 5.2: Links: de verhouding tussen het aantal uitgevoerde taken in een experiment en hoe vaak een AGV een beter pad selecteerde. Rechts: de verhouding tussen het aantal uitgevoerde taken in een experiment en hoe vaak een AGV een betere taak selecteerde. Beiden afkomstig uit het experiment rond de invloed van coalitievorming. Op de horizontale as wordt in beide grafieken de totale hoeveelheid uitgevoerde taken afgebeeld. Op de verticale as wordt op de linkse grafiek het aantal keer dat een beter pad geselecteerd werd afgebeeld. Op de verticale as van de rechtse grafiek , het aantal keer dat een betere taak geselecteerd werd. Op beide figuren worden de verschillende methodes afgebeeld door de kleuren van de datapunten.



Figuur 5.3: Verhouding tussen het aantal betere paden die geselecteerd werden in een experiment en hoe vaak een AGV een betere taak selecteerde, afkomstig uit het experiment rond de invloed van coalitievorming. Op de horizontale as wordt het aantal keer dat een beter pad geselecteerd werd afgebeeld, op de verticale as het aantal keer dat een betere taak geselecteerd werd. De verschillende methodes worden afgebeeld door de kleuren van de datapunten.

Ook opmerkelijk is de locatie van de datapunten bovenaan links op de linker- en rechter-

grafiek van Figuur 5.2 en het datapunt rechtsboven op Figuur 5.3. Dit zijn datapunten met een kleine hoeveelheid uitgevoerde taken en een grote hoeveelheid betere paden en taken. Dit is mogelijks een situatie van een te nerveus systeem zoals beschreven in subsectie 2.2.3. Verdere tests zullen uitwijzen of dat wel degelijk het geval is en of er dus een aanpassing in de heuristische waarden van de AGV's nodig is om dit mogelijks te nerveuze gedrag weg te werken.

Tabel 5.3: Verhouding tussen het aantal deadlocks en de uitvoeringsmethodes. Het kopje "Aantal deadlocks" stelt de hoeveelheid deadlocks voor die zich in een experiment voordoen, terwijl "Totale deadlocktijdspanne" de som voorstelt van de duur voor het oplossen van die deadlocks, in seconden. Het gaat om gemiddelden over alle experimenten. De rechterkolom stelt de MAD voor van de gemiddelde oplossingstijd van de deadlocks over alle experimenten.

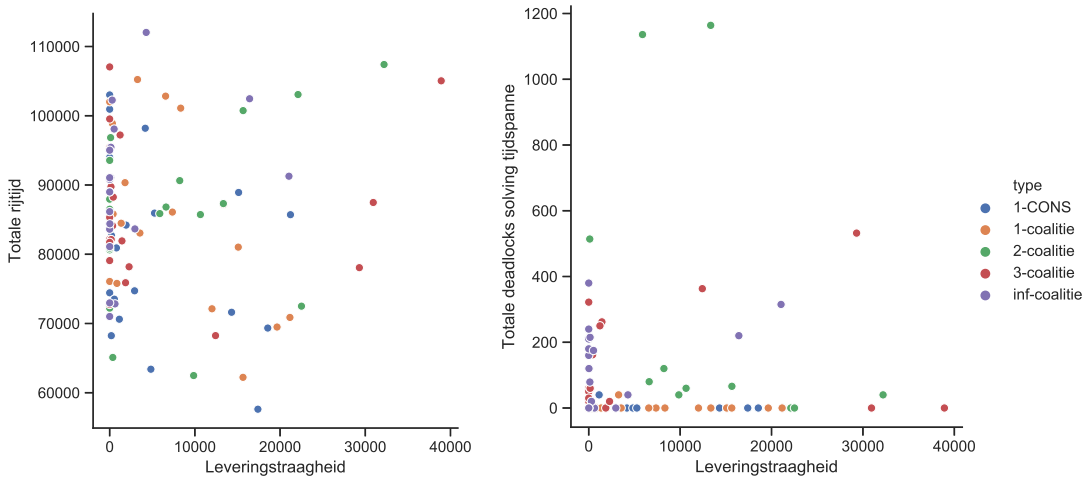
	Aantal deadlocks	Totale deadlocktijdspanne	MAD
1-CONS	0.05	2.00	3.80
1-coalitie	0.10	2.00	3.80
2-coalitie	1.55	171.00	231.00
3-coalitie	2.50	109.55	123.37
inf-coalitie	3.80	140.20	100.28

In Tabel 5.3 is duidelijk te zien dat het vergroten van de coalities meer deadlocks introduceert in het systeem. Die extra deadlocks gaan ook gepaard met een veel hogere oplossingstijd dan bij de enkele-consument- en 1-coalitiemethodes. Het oplossen van deadlocks geeft duidelijk ook een grote verspreiding aan in de MAD-waarden. Sommige deadlocks worden heel snel opgelost terwijl andere er veel langer over doen. Het wegwerken van die deadlocks en de verdere optimalisatie van oplossingen voor de deadlocks zouden de performantie van de grotere coalitiemethodes kunnen verbeteren en ze mogelijks meer competitief maken met de andere methodes. De geïntroduceerde deadlocks bevinden zich vooral in gebieden met grote hoeveelheden AGV's zoals rond knooppunten die een druk exportluik zijn of meerdere taakknooppunten die zich op plaatsen bevinden waar meerdere AGV's naartoe gaan. Het tweede geval komt vaker voor met grotere hoeveelheden AGV's in de gevallen met grotere coalities.

Tabel 5.4 stelt de leveringstraagheid van de verschillende methodes voor. In dit geval scoren de grotere coalitieoplossingen beter dan de kleinere en de enkele-consumentmethode. Dat is ook logisch omdat er zich door de grote coalities meer AGV's aanmelden om dringende taken uit te voeren waardoor die ook sneller zullen worden uitgevoerd. De traagheid is echter enorm verspreid. Sommige experimenten ondervinden geen traagheid terwijl andere experimenten een grote traagheid vertonen voor sommige taken. De MAD is hier het bewijs van. Figuur 5.4 toont aan dat de grotere leveringstraagheid niet afkomstig is van grotere afstanden die de AGV's afleggen of van deadlocks die de leveringen ophouden. Waarschijnlijk lopen de AGV's achterstand op hun uit te voeren taken op terwijl een aantal taken niet mogelijk zijn om ze tijdig uit te voeren, zoals vermeld in de vorige paragrafen. Die achterstand kunnen de AGV's niet inhalen waardoor sommige taken zeer laat worden uitgevoerd.

Tabel 5.4: Verhouding tussen de leveringstraagheid en de uitvoeringsmethodes. De te late leveringen zijn de gemiddelde aantal leveringen die gebeuren na hun eindtijdstip. De leveringstraagheid, zoals besproken in 2.1.3, is de gemiddelde totale traagheid van alle experimenten, in seconden, met de MAD van die gemiddelde traagheid.

	Te late leveringen	leveringstraagheid	MAD
1-CONS	5.45	5442.85	5939.88
1-coalitie	6.70	6542.10	6029.01
2-coalitie	5.60	7376.50	7554.50
3-coalitie	4.25	5974.60	8768.46
inf-coalitie	2.40	2322.30	3542.08



Figuur 5.4: Links: de verhouding tussen de leveringstraagheid en de totale rijtijd. Rechts: de verhouding tussen de leveringstraagheid en de totale oplossingstijd van de deadlocks, allebei afkomstig uit het experiment rond de invloed van coalitievorming. Op de horizontale as wordt in beide grafieken de leveringstraagheid afgebeeld, in seconden. Op de verticale as wordt op de linkse grafiek de totale rijtijd per experiment afgebeeld in seconden, op de rechtse grafiek is dit de oplossingstijd van de deadlocks in seconden. Op beide figuren worden de verschillende methodes afgebeeld door de kleuren van de datapunten.

5.4 Invloed van de laad- en verplaatsingstijd verhouding

Dit experiment legt de nadruk op de verhouding tussen de laad- en verplaatsingstijd en de impact van die verhouding op de verschillende methodes. In dit experiment wordt er niet dieper ingegaan op de leveringstraagheid van de verschillende methodes. Er werden geen aanpassingen gedaan aan het taak-generatie algoritme voor dit experiment hoewel de taken in beide gevallen aanzienlijk langer zullen duren. De leveringstraagheid zal dus niet representatief zijn door het onmogelijk uitvoeren van de taken in de aangevraagde tijd. Door de aangepaste waarden, kunnen de waarden afgeleid uit de experimenten in deze subsectie niet onderling of met de waarden uit sectie 5.3 vergeleken worden.

In onderdeel één van dit experiment wordt er gefocust op langere laad- en lostijden. De laadtijd van een pakket wordt ingesteld op 300 seconden of tien keer de standaardwaarde die gebruikt werd in sectie 5.3. De invloed van die aanpassingen op de totale hoeveelheid uitgevoerde taken is terug te vinden in Tabel 5.5. In dit geval is het duidelijk dat de enkele-consument-, de 1- en 2-coalitiemethodes opnieuw de beste resultaten halen. Ze bevinden zich alle drie relatief dicht bij elkaar met een vergelijkbare MAD. Het drastische vergroten van de laadtijd schijnt dus geen grote impact te hebben op welke methode de beste resultaten haalt of op de afstand tussen de resultaten.

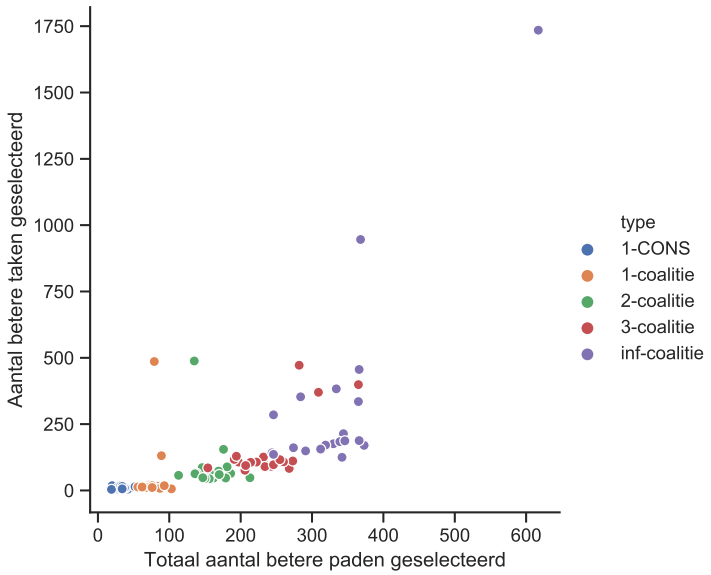
In het tweede onderdeel van dit experiment worden de langere rijtijden bekeken. Dit wordt verwezenlijkt door het gebruik van de standaardwaarden voor de experimenten maar met de gemiddelde AGV-snelheid gehalveerd naar slechts 1 m/s. In Tabel 5.6 wordt het aantal uitgevoerde taken weergegeven voor dit experiment. Bij het vergroten van de rijtijden blijkt het interessanter te zijn om te werken met grotere coalitiemethodes. Vooral de 2- en 3-coalitiemethodes blijken betere resultaten te verkrijgen dan de andere methodes. Dat die twee methodes beter scoren is ook logisch. Door de grotere verhouding tussen laad- en verplaatsingstijden is het interessant om meerdere AGV's naar een belangrijkere taak te sturen. De kans is namelijk groter dat als er zich meerdere AGV's naar een coalitie begeven, de parkeerlane vrij zal zijn als er een AGV aankomt. Dat is voornamelijk toe te schrijven aan de langere rijtijd en de verhouding tussen de laad- en rijtijd. Hierdoor kan de nu net aangekomen AGV onmiddellijk oprijden en moet hij niet wachten tot een andere AGV zich van de parkeerlane verplaatst. Bij de enkele-consument- en 1-coalitiemethodes is er door de tragere AGV's veel extra tijd tussen het laden van de verschillende auto's, waardoor die methodes lager scoren. Op Figuur 5.5 zijn opnieuw dezelfde clusters te zien zoals in sectie 5.3. Opvallend is wel dat er opnieuw sprake is van te nerveuze agenten die, net als bij sectie 5.3, te veel van taak en pad veranderen. Dat is zichtbaar in de twee grote uitschieters in de rechterbovenhoek van Figuur 5.5, allebei zijn afkomstig uit de inf-coalitie experimenten. Die inf-coalitie experimenten blijken daar heel gevoelig aan te zijn door de grote hoeveelheid uitvoerbare taken die altijd op het netwerk aanwezig zijn.

Tabel 5.5: Experimentresultaten: invloed van de verhouding tussen laad- verplaatsingstijd op taken met een grote laadtijd.

	Gemiddeld aantal uitgevoerde taken	MAD
1-CONS	353.05	14.00
1-coalitie	345.40	11.04
2-coalitie	347.85	14.47
3-coalitie	329.95	17.57
inf-coalitie	323.90	18.03

Tabel 5.6: Experimentresultaten: invloed van de verhouding tussen laad- verplaatsingstijd op taken met een grotere verplaatsingstijd.

	Gemiddeld aantal uitgevoerde taken	MAD
1-CONS	600.30	21.70
1-coalitie	617.90	27.20
2-coalitie	656.00	29.70
3-coalitie	660.30	35.30
inf-coalitie	632.70	45.69



Figuur 5.5: De verhouding tussen het aantal betere paden die geselecteerd waren in een experiment en hoe vaak een AGV een betere taak selecteerde, afkomstig uit het experiment rond de verhouding tussen laad- verplaatsingstijd. Op de horizontale as wordt het aantal keer dat er een beter pad geselecteerd wordt, afgebeeld. De verticale as toont het aantal keer dat er een betere taak geselecteerd werd. De verschillende methodes zijn afgebeeld door de kleuren van de datapunten.

5.5 Invloed van de heuristische waarden op taakbeslissingen

In deze sectie bekijken we de invloed van de heuristische waardes voor de prioriteiten van de taken, en dus ook hun invloed op het taakbeslissingsproces van de AGV's. We bekijken vier verschillende opties voor de heuristische waardes, naast de standaardwaarden. Alle vier verschillende opties spelen zich af in de 1-coalitiemethode. De eerste optie, vermeld als 1-coalitie-lage-groep-prioriteit, heeft de standaardwaardes zoals eerder beschreven, maar CoalitionSizeFactor en PickedUpTasksFactor werden allebei ingesteld op 1. In dit scenario worden alle factoren uit subsectie 4.7.3 ingesteld op 1 en dus weggewerkt. Voor de tweede optie, waar we voortaan naar verwijzen als 1-coalitie-geen-groep-prioriteit, worden de CoalitionSizeFactor en PickedUpTasksFactor ingesteld op 0. Er wordt dus voor de prioriteiten van de taken geen rekening meer gehouden met de keuzes van de andere AGV's. In de derde optie wordt de tijdprioriteit in een kleine mate geschaald, daarom verwijzen we voortaan naar deze optie als 1-coalitie-lage-tijd-prioriteit. Bij de laatste optie wordt de tijdprioriteit heel hoog geschaald en wordt en er wordt verder naar verwezen als 1-coalitie-hoge-tijd-prioriteit. De twee laatste opties hebben allebei de standaardwaardes en gebruiken voor de tijd heuristische waardes, vermeld in Tabel 5.7.

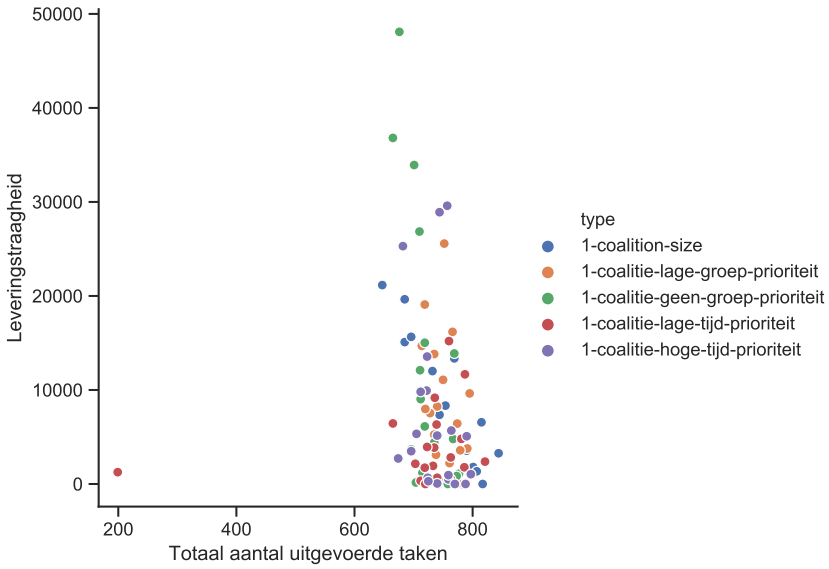
Tabel 5.7: Tijdwaarden voor de heuristische experimenten.

	Standaard	lage-tijd-prioriteit	hoge-tijd-prioriteit
TijdPrioriteit 60+ Minuten	2	1	1
TijdPrioriteit 30+ Minuten	4	2	4
TijdPrioriteit 15+ Minuten	8	3	16
TijdPrioriteit 0+ Minuten	16	4	64

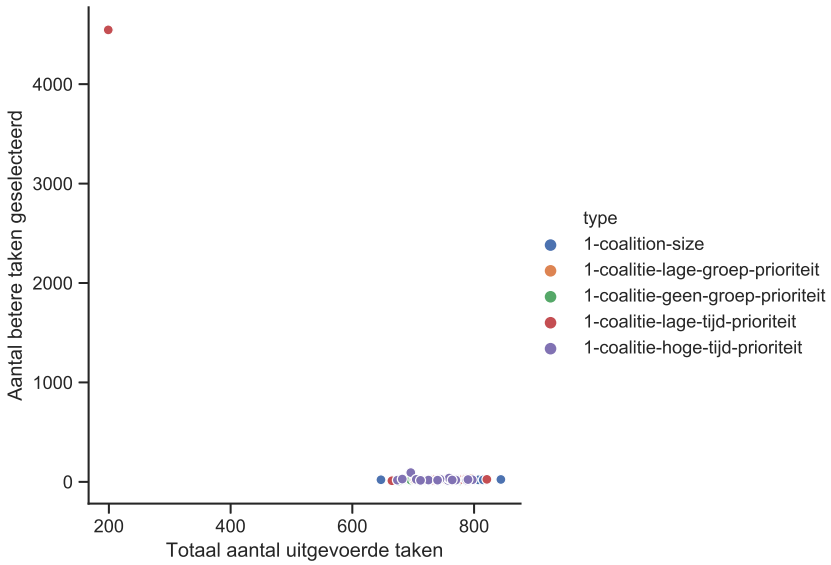
In Tabel 5.8 worden het aantal uitgevoerde taken per optie vermeld samen met de 1-coalitiemethode. Uit die tabel blijkt dat de lage-groep-prioriteit en de standaardwaardes gelijkaardige resultaten behalen. De geen-groep-prioriteit en de lage-tijdprioriteit voeren beiden een stuk minder taken uit dan de standaardwaardes. In Figuur 5.6 wordt afgebeeld hoe de heuristische waardes de leveringstraagheid beïnvloeden. Het is duidelijk zichtbaar dat de geen-groep-prioriteit en de lage-tijd-prioriteit duidelijke uitschieters hebben in vergelijking met de rest van de cluster. Bij de lage-tijd-prioriteit is het ook logisch dat sommige taken te laat zullen worden uitgevoerd doordat er geen hoge prioriteiten meer worden toegewezen aan taken die al te laat zijn. Dat zorgt ervoor dat die taken niet voorgenomen worden op andere tijden waardoor de leveringstraagheid enkel maar kan stijgen. De uitlopers in de geen-groep-prioriteit zijn toe te schrijven aan het feit dat AGV's zich niet gaan aansluiten aan coalities door het gebrek aan motivatie door het ontbreken van de prioriteiten die coalities promoten. Dat zorgt ervoor dat sommige AGV's lang wachten vooraleer een taak volledig wordt uitgevoerd. De uitloper van de lage-tijd-prioriteit is opnieuw een voorbeeld van te nerveuze agenten, zoals bevestigd wordt in Figuur 5.7. De uitloper in Figuur 5.7, linksonder is namelijk dezelfde als de uitloper in de lage-tijd-prioriteit van Figuur 5.6, linksboven. Die te nerveuze agenten kunnen geen taak kiezen en maken elk voortdurend een andere keuze voor de beste taak.

Tabel 5.8: Experimentresultaten: invloed van de aanpassingen aan de heuristische waarden.

	Gemiddeld aantal uitgevoerde taken	MAD
1-coalitie	754.05	41.36
1-coalitie-lage-groep-prioriteit	750.20	19.82
1-coalitie-geen-groep-prioriteit	724.20	24.14
1-coalitie-lage-tijd-prioriteit	718.70	59.58
1-coalitie-hoge-tijd-prioriteit	738.55	28.40



Figuur 5.6: Visualisatie van de verschillende heuristische opties en hun impact op de leveringstraagheid, met de verschillende opties afgebeeld door de kleuren van de datapunten. Op de horizontale as staat het totale aantal uitgevoerde taken, terwijl op de verticale as de totale leveringstraagheid wordt afgebeeld.



Figuur 5.7: Verhouding tussen het aantal uitgevoerde taken in de experimenten met heuristische waardes en hoe vaak een AGV een betere taak selecteerde. Op de horizontale as wordt de totale hoeveelheid uitgevoerde taken afgebeeld. De verticale as toont het aantal keer dat er een betere taak geselecteerd werd. Ook hier worden de verschillende opties van de heuristische waardes afgebeeld op basis van de kleur van het datapunt.

5.6 Invloed van de HoldIA's

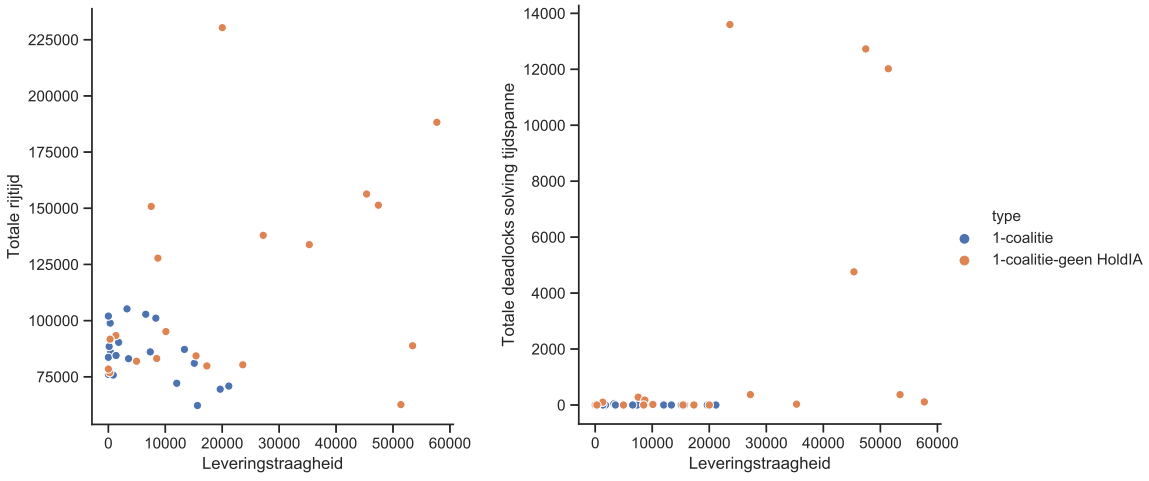
In dit laatste experiment wordt de invloed van de HoldIA's bekeken op het uitvoeren van de taken. De HoldIA's verhinderen AGV's om een mogelijke deadlock te maken door een parkeerlane te benaderen waar al een AGV aanwezig is. In dit experiment vergelijken we het standaard 1-coalitie experiment en een experiment met dezelfde standaardwaardes waar de HoldIA's vervangen zijn door standaard infrastructuuragenten. In Tabel 5.9 worden alle uitgevoerde taken van het experiment afgebeeld. Zoals verwacht, voert het standaardexperiment meer taken uit. Het experiment zonder de HoldIA's blijkt heel wat slechter te werken in taken en met een veel grotere MAD. Dit toont aan dat het gebrek aan HoldIA's niet in elke situatie slechtere resultaten veroorzaakt. De HoldIA's blijken toch een groot aantal deadlocks tegen te houden, zoals Tabel 5.10 aantoont. Het oplossen van de deadlocks in de oplossing zonder HoldIA's vraagt tijd, zoals te zien is in de rechtergrafiek in Figuur 5.8, waar de standaardoplossing bijna geen leveringstraagheid vertoont in vergelijking met de oplossing zonder HoldIA's. Voor de oplossing zonder de HoldIA's is in de linkergrafiek in Figuur 5.8 duidelijk te zien dat die oplossing een veel grotere rijtijd vergt. Doordat de AGV's tot toe rijden en dan merken dat ze de parkeerlane door de aanwezigheid van een andere AGV niet kunnen inrijden, moeten ze rondrijden of zich naar een rustplaats verplaatsen om de deadlocks op te lossen. Dit zijn onnodige en vermijdbare verplaatsingen die het uitvoeren van de taken vertragen.

Tabel 5.9: Experimentresultaten rond de invloed van de HoldIA's.

	Gemiddeld aantal uitgevoerde taken	MAD
1-coalitie	754.05	41.36
geen HoldIA	538.60	162.84

Tabel 5.10: De verhouding tussen het aantal deadlocks en de HoldIA's. De titel "Aantal deadlocks" stelt de hoeveelheid deadlocks voor die zich in een experiment voordoen. "Totale deadlocktijdspanne" stelt de som voor van hoe lang het oplossen van alle deadlocks duurde, in seconden. Beide gegevens zijn de gemiddelden van alle experimenten met in de rechtse kolom de MAD van de gemiddelde oplossingstijd van de deadlocks over alle experimenten.

	Aantal deadlocks	Totale deadlockstijdpanne	MAD
1-coalitie	0.10	2.00	3.80
geen HoldIA	7.35	2228.25	3419.6



Figuur 5.8: Links: de verhouding tussen de leveringstraagheid en de totale rijtijd. Rechts: de verhouding tussen de leveringstraagheid en de totale oplossingstijd van de deadlocks. Allebei afkomstig uit het experiment rond de invloed van de HoldIA's. Op de horizontale as wordt, in beide grafieken, de leveringstraagheid afgebeeld, in seconden. Op de verticale as wordt op de linkse grafiek de totale rijtijd per experiment afgebeeld in seconden. Op de rechtse grafiek staat de oplossingstijd van de deadlocks in seconden. In beide figuren worden de verschillende methodes afgebeeld door de kleuren van de datapunten.

5.7 Conclusie

Deze sectie spitst zich toe op de resultaten van de experimenten, voorgesteld in de voorgaande secties, en hun implicaties op de oplossing die we bieden. Uit sectie 5.3 kunnen we afleiden dat het verplaatsen van de auto's als individuele taken per auto en als langzaam opbouwende coalitie, die alle te verplaatsen auto's als één taak ziet, kan beschouwd worden. Beide methodes behalen gelijkaardige resultaten maar de 1- en 2-coalitiemethodes blijken net iets beter te presteren. Die vaststelling wordt opnieuw bevestigd in sectie 5.4 als we de laadtijd vergroten. In dezelfde sectie blijkt echter dat als we de rijtijd vergroten, en de AGV's dus langer onderweg zijn naar de taken, de grotere coalities beter beginnen te presteren. Er is dus een verhouding tussen de rij- en laadtijd die beïnvloedt welke methode beter presteert. Dit bevestigt onze hypothese dat er een verhouding aanwezig is tussen de rij- en laadtijd die een impact heeft op de oplossingen. Dit houdt ook in dat er zoals verwacht geen enkele coalitiegrootte kan worden aanbevolen die in alle gevallen de beste resultaten zal opleveren.

Zoals te zien in de secties 5.3, 5.4 en 5.5 hebben we in de voorgestelde oplossing af en toe last van te nerveuze agenten, met soms rampzalige gevolgen voor de oplossing. Die te nerveuze agenten dienen opgelost te worden. Ze komen vooral voor in de experimenten met grotere coalities en bij experimenten rond de heuristische waardes. De heuristische waardes zullen dus moeten worden bijgesteld. Die heuristische waardes hebben een grote impact op het gedrag van agenten, zoals zichtbaar in sectie 5.5. Het wegnemen van de groepsprioriteiten, de hoeveelheid AGV's met al opgepikte taken en de hoeveelheid AGV's op weg naar een coalitie, verminderen het totale aantal uitgevoerde taken. In het geval waar er een lagere tijdsprioriteit wordt toegewezen, blijkt echter dat de leveringstraagheid sterk toeneemt. Het toekennen van een hogere tijdsprioriteit zorgt er echter voor dat de AGV's te veel focussen op taken die binnenkort verlopen, waar het totale aantal uitgevoerde taken onder lijdt. Verder onderzoek naar die heuristische waardes is noodzakelijk. De factoren zullen hoogstwaarschijnlijk net zoals de coalitiegrootte verschillen per scenario. Het is mogelijk dat het verschil tussen het aantal uitgevoerde taken per methode in sectie 5.3 te overbruggen is door gebruik te maken van betere heuristische waardes en gewogen sommen.

Bovendien veroorzaken grotere coalities over alle experimenten heen ook grotere en meer deadlocks rond de ingangen van de parkeerlanes. Een ingewikkelder en meer performante wachtsysteem voor de AGV's kan dit probleem eventueel oplossen. Hoewel de HoldIA-oplossing, voorgesteld in subsectie 4.6.4, blijkt te werken voor de kleine coalities, zoals aangetoond in sectie 5.6, is die mogelijks te kleinschalig voor de grotere coalities. Die vaststelling bevestigt dus de hypothese rond de deadlocks aan de ingangen van de parkeerplaatsen. De restricties rond het batterijniveau blijken ook te gelden voor de paden en de taken. In geen enkel experiment viel er een AGV zonder batterij. De vraag is of deze werkwijze ook zo performant is op grotere sites, of als er mogelijke aanpassingen nodig zijn die tot meer performante keuzes van de AGV's leiden.

Hoofdstuk 6

Besluit

In dit hoofdstuk sluiten we deze masterproef af. Eerst bespreken we het doel, de literatuurstudie, implementatie en resultaten van deze masterproef. In de volgende paragrafen worden de onderzoeksvragen ook herhaald en beantwoord. Daarna worden er enkele punten aangehaald die nog voor verbetering vatbaar zijn. Tot slot sommen we enkele vervolgonderzoeken op die uit deze masterproef kunnen voortvloeien.

Het hoofddoel van dit werk was het onderzoeken en implementeren van onderling sequentiële afhankelijke taken in een DMAS-omgeving. Om die taak te vergemakkelijken, werd een casestudy geïntroduceerd gebaseerd op een havenbedrijf waarin auto's verplaatst worden van parkeerlanes naar leveringslocaties. Dit werk is ook deel van een haalbaarheidsonderzoek voor het automatiseren van een kade voor hetzelfde havenbedrijf. Dit probleem kan als een typisch PDP omschreven worden. De aanwezige infrastructuur van het havenbedrijf, namelijk de YardSolver, laat ons toe het probleem verder te specificeren naar een DARPWD. PDP, en dus ook DARPWD, hebben enkele opvallende kenmerken die de interacties tussen de agenten moeilijk maken. Toch wordt geargumenteerd dat MAS door zijn gedecentraliseerde natuur geschikt is om PDP op te lossen. Door het tekort aan specifieke informatie rond de AGV's, die de auto's zullen verplaatsen, zijn er abstracties gemaakt rond die AGV's. We gaan er ook van uit dat de vloot van AGV's homogeen is en dat elke AGV elke taak zal kunnen uitvoeren.

Deze paragraaf beantwoordt de centrale onderzoeksvraag van deze masterproef, namelijke "Kunnen onderling sequentiële afhankelijke taken efficiënt in een DMAS-omgeving geïmplementeerd worden?". In het kort is het antwoord op die vraag duidelijk ja. Het verplaatsen van de auto's valt onder een multirobot-taakallocatieprobleem. De onderling sequentiële afhankelijke taken kunnen worden omschreven door het werk van Korsah et al. [24] als cross-schedule dependencies SR-taken of als MR-taken. De twee verschillende benaderingen van het probleem leiden ook tot twee verschillende oplossingen. In de SR-manier wordt bij het verplaatsen van een rij auto's elke auto als een individuele taak beschouwd en wordt er voor elke auto een aparte taak aangeboden die door een AGV wordt opgelost. In de MR-manier wordt het volledig verplaatsen van de rij auto's als één enkele taak opgevat, die dan wordt opgelost door een langzaam groeiende coalitie van AGV's. Het is niet interessant om een afhankelijke taak op te pikken als de andere AGV's zich toeleggen op andere afhankelijke taken. De AGV's bevatten zelf de beslissingslogica, in de vorm van een BDI-implementatie, die de acties van de AGV's en de

interesses in het aansluiten bij een coalitie bepaalt. Het beslissingsproces wordt aangestuurd via taakprioriteiten die worden afgestemd op zowel de taak als zijn afhankelijke taken. Uit de resultaten van de experimenten, voorgesteld in hoofdstuk 5, blijkt dat er geen optimale oplossing mogelijk is voor alle scenario's. In bepaalde scenario's, zoals het scenario beschreven in sectie 5.3, werken beide oplossingen op vergelijkbare niveaus. Uit sectie 5.4 blijkt echter dat de langzaam groeiende coalitiemethode, die ontstaat door alle verplaatsingen als één taak te beschouwen, betere resultaten kan behalen.

Voor de deelvraag, "Wat is het effect van de oplossingsmethode op de AGV's en hun coördinatie?", is het belangrijk om een onderscheid te maken tussen de verschillende methodes. De coördinatie tussen de AGV's op padbepalingsniveau wordt niet aangepast tussen de verschillende methodes en wordt bereikt door een DMAS-routeringsmethode die uitgebreid werd om ook informatie te verzamelen op batterij-impact van de paden en een deadlockpreventiemethode op die paden. Voor het toewijzen en uitvoeren van taken worden er extra mieren toegevoegd die dit proces helpen. Die extra mieren ontvangen elk nog uitbreidingen die het beslissingsproces van de AGV's moeten helpen om optimale beslissingen te nemen. In het geval van de enkele-consumentmethode is er weinig onderlinge coördinatie nodig tussen de AGV's doordat elke taak als een individuele taak wordt beschouwd. Bij de langzaam groeiende coalitiemethode is er meer coördinatie noodzakelijk tussen de AGV's zowel op beslissingsvlak als op padbepalingsvlak. De langzaam groeiende coalitiemethode voegt immers deadlocks toe rond de parkeerlanes door het benaderen van meerdere AGV's. Om die deadlocks te vermijden, werden er op verschillende niveaus extra deadlockpreventiemethodes toegevoegd. Die preventiemethodes halen goede resultaten bij een kleine hoeveelheid AGV's en rond de ingangen van de parkeerlanes blijken ze een verplicht onderdeel te zijn van een performante oplossing. Daarnaast blijkt er ook een verhouding te zijn tussen laad- en rijtijd die bepaalt welke oplossing betere resultaten behaalt. Beide hypothesen, voorgesteld in sectie 1.2, zijn dus correct. De voorgestelde oplossing blijkt echter te nerveuze agenten aan te moedigen, die door zo vaak van taak en pad te veranderen slechte resultaten behalen. Dit is gedeeltelijk toe te schrijven aan de heuristische waardes die de AGV's gebruiken voor hun beslissingsproces en de mogelijkheid om onbeperkt van beslissing te mogen veranderen. Als enkele AGV's te vaak van taak veranderen, kan dit probleem zich door het volledige netwerk verspreiden. Die te nerveuze agenten dienen te worden weggewerkt in verbeteringen van de oplossing en zijn het grootste probleem op niveau van coördinatie tussen de AGV's. Dit probleem wordt echter nog verergerd wanneer er heel veel taken beschikbaar zijn, zoals in het geval van grote coalities. De heuristische waardes, en dus ook het beslissingsproces van de AGV's, dienen verder te worden geoptimaliseerd voor de verschillende scenario's en kunnen mogelijks de doorslag geven of één oplossing significante betere resultaten oplevert dan de andere oplossing.

Vervolgens kaarten we enkele suggesties aan van zaken die in deze masterproef nog verder kunnen worden verbeterd. Het oplossen van de deadlocks kan optimaler verlopen door ze nauwer te bekijken voordat de AGV's hun opdrachten krijgen die de deadlocks dienen op te lossen. Wachtrijen worden in sommige gevallen ook als deadlocks beschouwd. Om de resultaten doorslaggevender te maken, kan er in de toekomst ook een standaard dataset worden gebruikt die een werkelijke situatie voorstelt. Die dataset kan zodanig worden opgebouwd zodat in theorie elke taak met een leveringstraagheid van 0 uitvoerbaar is.

Ook grote hoeveelheden AGV's in nauwe locaties leiden tot problemen doordat de reservaties voor de padexploratiemieren niet op een optimale manier verlopen. De verdere optimalisatie en de vertragingspropagatie zouden de AGV's moeten toelaten om in grotere aantallen aanwezig te zijn op nauwe locaties. Ook de heuristische waardes kunnen nog verder worden geoptimaliseerd. Het is ook mogelijk om de infrastructuuragenten uit te breiden met een padopslagsysteem dat toelaat om bepaalde mieren op te slaan. Op die manier kunnen afstanden tussen een knooppunt en een oplaadstation worden bijgehouden en kan de hoeveelheid mieren daardoor verminderen.

Een interessant vervolgonderzoek zou eruit bestaan om de scoringsmethode van algoritme 3, in subsectie 4.7.1, uit te breiden met een haalbaarheidsmierenstructuur waar de opslagagent een mier naar het wegennet stuurt en meet hoe druk het wegennet is. Die lokale informatie kan dan gebruikt worden om de scores te verbeteren en een optimalere taaktoewijzing te bereiken. Als alle auto's moeten worden verplaatst, dan wordt aan elke kant de helft van de taken toegewezen. Het toevoegen van een haalbaarheidsmierstructuur zou die situatie mogelijks kunnen optimaliseren, afhankelijk van de verkeersdrukte. Het invoeren van die haalbaarheidsmierstructuur zou het gebruik van dynamische toewijzingen bij de dualopslagagenten toelaten en de toewijzing kunnen aanpassen afhankelijk van accurate up-to-date lokale informatie. Daarnaast zou het invoeren van een andere graaf om de invloed ervan op de oplossing te testen, ook zeer interessant zijn. Die nieuwe graaf kan ook drastisch vergroot worden om de impact van de voorgestelde oplossing op grotere schaal te testen. Bovendien zouden dynamische knooppunten die het gebruik van de wegen voorstellen door mensen en niet-AGV vervoer op de site ook een interessante uitbreiding zijn, die een dynamische real-time omgeving realistischer zou voorstellen. Het implementeren van een state-of-the-art-techniek in zo'n dynamische realistische omgeving waar paden onderbroken worden en de vergelijking van die techniek met onze voorgestelde oplossing zou doorslaggevend zijn om te weten of de voorgestelde oplossingen verbeteringen kan bieden op het vlak van taakallocatie in sterk dynamische grootschalige omgevingen. Onderzoeken of de voorgestelde oplossingen in een ander scenario onderling nog steeds dezelfde eigenschappen vertonen, zoals het totale aantal taken en de leveringstraagheid, kan ook verder afbakenen in welke situatie welke methodes beter presteren. Het laatste vervolgonderzoek bestaat eruit te bekijken of het langzaam groeiende deel van de coalitie dynamisch kan worden gemaakt. Via een haalbaarheidsmiereninfrastructuur zou een opslagagent de drukte om zich heen kunnen meten. Dat kan worden gecombineerd met extra beslissingscapaciteiten in de infrastructuuragenten, via bijvoorbeeld een veilingmechanisme om dynamisch de beste grootte van het langzaam groeiende deel van de coalitie te bepalen. Zo kunnen er eventueel betere resultaten ontstaan.

De langzaam groeiende coalitiemethode op basis van DMAS toont veelbelovende resultaten en kan makkelijk worden aangepast op basis van de scenario's waarin ze zich afspeelt. Er zijn nog verdere optimalisaties mogelijk die het in dynamische DARP-situaties interessant maken om de DMAS-oplossing te vergelijken met state-of-the-art-methodes. Het is dus zeker mogelijk om DMAS en de langzaam groeiende coalitiemethode in de verdere haalbaarheidsstudie van het havenbedrijf verder te bestuderen en te implementeren als de vergelijking met de state-of-the-art-methodes positieve resultaten oplevert.

Bibliografie

[1] *Automated Guided Vehicle Systems: A Primer with Practical Applications*, second revised and expanded edition. ed. Springer Berlin Heidelberg : Imprint: Springer, Berlin, Heidelberg, 2015.

[2] Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., and Laporte, G. Static pickup and delivery problems: A classification scheme and survey. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research 15* (02 2007), 1–31.

[3] Botelho, S. C., and Alami, R. M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)* (1999), vol. 2, pp. 1234–1239 vol.2.

[4] Chopra, S., Notarstefano, G., Rice, M., and Egerstedt, M. A distributed version of the hungarian method for multirobot assignment. *IEEE Transactions on Robotics 33*, 4 (2017), 932–947.

[5] Cordeau, J.-F., and Laporte, G. The dial-a-ride problem (darp): Variants, modeling issues and algorithms. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies 1* (2003), 89–101.

[6] Desaulniers, G., Desrosiers, J., Erdmann, A., Solomon, M., and Soumis, F. *VRP with Pickup and Delivery*. 01 2002, pp. 225–242.

[7] Dinh, H. T., van Lon, R., and Holvoet, T. Multi-agent route planning using delegate mas. Komenda, Antonín, London, UK, pp. 24–32.

[8] Dorri, A., Kanhere, S., and Jurdak, R. Multi-agent systems: A survey. *IEEE Access* (04 2018), 1–1.

[9] Fischer, K., and Müller, J. A decision-theoretic model for cooperative transportation scheduling. vol. 1038, Springer Verlag, pp. 177–189.

[10] Fischer, K., Müller, J., and Pischel, M. Cooperative transportation scheduling: an application domain for dai. *Applied Artificial Intelligence 10* (02 1996), 1–33.

[11] Gerkey, B., and Mataric, M. A formal analysis and taxonomy of task allocation in multi-robot systems. *I. J. Robotic Res. 23* (09 2004), 939–954.

[12] Gonzalez, C. Changing the future of warehouses with amazon robots. *Machine Design* (2017).

[13] Guerrero, J., and Oliver, G. Multi-robot task allocation strategies using auction-like mechanisms.

[14] Hanif, S., and Holvoet, T. Applying delegate MAS patterns in designing solutions for dynamic pickup and delivery problems. In *Advances in Artificial Transportation Systems and Simulation*. Elsevier, 2015, pp. 79–102.

[15] Hanif, S., and Holvoet, T. Applying delegate mas patterns in designing solutions for dynamic pickup and delivery problems. In *Advances in artificial transportation systems and simulation*. Elsevier, 2015, pp. 79–102.

[16] Hanif, S., Lon, R., Gui, N., and Holvoet, T. Delegate mas for large scale and dynamic pdp: A case study. vol. 382, pp. 23–33.

[17] Ho, S. C., Szeto, W., Kuo, Y.-H., Leung, J. M., Petering, M., and Tou, T. W. A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological 111*, C (2018), 395–421.

[18] Holvoet, T., and Valckenaers, P. Beliefs, desires and intentions through the environment. pp. 1052–1054.

[19] Holvoet, T., Weyns, D., and Valckenaers, P. Delegate mas patterns for large-scale distributed coordination and control applications. *EuroPLoP 2010* (07 2010).

[20] Jones, E., Dias, M., and Stentz, A. Time-extended multi-robot coordination for domains with intra-path constraints. *Autonomous Robots 30* (01 2009), 41–56.

[21] Kadar, A., Muhammad Masum, A. K., Faruque, M., Shahjalal, M., Iqbal, M., and Sarker, I. Solving the vehicle routing problem using genetic algorithm. *International Journal of Advanced Computer Science and Applications 2* (08 2011).

[22] Khamis, A., Hussein, A., and Elmogy, A. *Multi-robot Task Allocation: A Review of the State-of-the-Art*, vol. 604. 05 2015, pp. 31–51.

[23] Koes, M., Nourbakhsh, I., and Sycara, K. Constraint optimization coordination architecture for search and rescue robotics. pp. 3977 – 3982.

[24] Korsah, G. A., Stentz, A., and Dias, M. B. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research 32*, 12 (2013), 1495–1512.

[25] Laukens, J. A multi-agent system for avoiding deadlocks in an automatic guided vehicle system, 2005.

[26] Lemaire, T., Alami, R., and Lacroix, S. A distributed tasks allocation scheme in multi-uav context. vol. 4, pp. 3622 – 3627 Vol.4.

[27] Liu, D., Barba-Guamán, L., Díaz, P. V., and Riofrío, G. Intelligent tutoring module for a 3dgame-based science e-learning platform. *Inteligencia Artif. 20* (2017), 1–19.

[28] Parker, L., and Tang, F. Building multirobot coalitions through automated task solution synthesis. *Proceedings of the IEEE 94* (08 2006), 1289 – 1305.

[29] Parragh, S. N., Doerner, K. F., and Hartl, R. F. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft 58* (2008), 81–117.

[30] Rao, A., and Georgeff, M. Modeling rational agents within a bdi-architecture.

[31] Rizk, Y., Awad, M., and Tunstel, E. Cooperative heterogeneous multi-robot systems: A survey.

[32] Savelsbergh, M., and Sol, M. The general pickup and delivery problem. *Transportation Science 29* (02 1995), 17–29.

[33] Shehory, O., and Kraus, S. Methods for task allocation via agent coalition formation. *Artificial Intelligence 101*, 1 (1998), 165 – 200.

[34] Shiroma, P., and Campos, M. Comutar: A framework for multi-robot coordination and task allocation. pp. 4817 – 4824.

[35] van Lon, R. Multi-agent systems for dynamic logistics - systematic evaluation and bio-inspired optimization, 2017.

[36] van Lon, R. R., Ferrante, E., Turgut, A. E., Wenseleers, T., Berghe, G. V., and Holvoet, T. Measures of dynamism and urgency in logistics. *European Journal of Operational Research 253*, 3 (2016), 614 – 624.

[37] van Lon, R. R. S., and Holvoet, T. RinSim: A simulator for collective adaptive systems in transportation and logistics. In *Proceedings of the 6th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2012)* (Lyon, France, 2012), pp. 231–232. 10.1109/SASO.2012.41.

[38] Weyns, D., Boucké, N., Holvoet, T., and Demarsin, B. Dyncnet: A protocol for dynamic task assignment in multiagent systems. pp. 281–284.

[39] Weyns, D., Boucké, N., and Holvoet, T. Gradient field-based task assignment in an agv transportation system. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems* (New York, NY, USA, 2006), AAMAS '06, ACM, pp. 842–849.

[40] Wooldridge, M. An introduction to multiagent systems, second edition.

[41] Yuan, Q., Guan, Y., Hong, B., and Meng, X. Multi-robot task allocation using cnp combines with neural network. *Neural Computing and Applications 23* (12 2013).

[42] Zhao, K. Multiple-agent task allocation algorithm utilizing ant colony optimization. *Journal of Networks 8* (10 2013), 2599–2606.

[43] Zlot, R. M. *An Auction-Based Approach to Complex Task Allocation for Multirobot Teams*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, December 2006.

AFDEL
Straat nr bus (
3000 LEUVEN, BEL
tel. + 32 16 00 0
fax + 32 16 00 0
www.kuleuve