Samenvatting (Dutch)

In deze dissertatie worden zowel het dynamische voertuig routing probleem met tijdskaders als een nieuwe versie met meerdere prioriteiten bestudeerd. Daarnaast wordt ook de uitgebreide versie gecombineerd met inventaris management, het zogeheten inventaris routing probleem, onderzocht.

Voor het dynamische voertuig routing probleem met tijdskaders zijn sommige orders niet vooraf bekend, deze worden pas bekend tijdens het afleggen van de route. Wanneer de orders bekend worden moeten deze ingevoegd worden tijdens de uitvoer van de vooraf gegenereerde route. Naarmate de tijd verstrijkt kunnen al uitgevoerde opdrachten en opdrachten die in uitvoer zijn niet meer aangepast worden. Als gevolg hiervan is het lastig een oplossing van hoge kwaliteit te vinden voor een hele werkdag. Een werkdag is verdeeld in een aantal tijdvakken. Aan het begin van elk tijdvak controleert het algoritme of er nieuwe opdrachten zijn die de *controller* af moet handelen. Wanneer er nieuwe opdrachten zijn moet het algoritme deze toevoegen en de routes herordenen. De grootste uitdaging is de efficiëntie van het algoritme. Allereerst moet het algoritme snel een valide oplossing vinden om deze oplossing vervolgens te verbeteren door het gebruik van een kortere route en minder voertuigen. Het bewaren van feromoon sporen is een mechanisme van het mierenkolonie optimalisatie algoritme dat het eenvoudiger maakt om extra kennis te gebruiken om valide oplossingen van hoge kwaliteit te genereren voor routing problemen.

Om dit probleem op te lossen wordt er een meervoudig mierenkolonie systeem toegepast. In dit algoritme wordt er een *controller* gebruikt om oplossingen voor mierenkolonies te initialiseren. Het nieuw voorgestelde *Ranking Time Windows Based Nearest Neighbor* algoritme is geïmplementeerd en vind goede initiële oplossingen voor het mierenkolonie algoritme en in het bijzonder voor het probleem met tijdskaders. De *controller* wordt ook gebruikt voor het bijhouden van de feromoon sporen van de twee kolonies, namelijk de ACS-VEI kolonie en de ACS-TIME kolonie. De ACS-VEI kolonie richt zich op het vinden van valide oplossingen die het aantal benodigde voertuigen minimaliseren voor het routing probleem. De ACS-TIME kolonie richt zich daarentegen op de minimalisatie van de totale afstand van het voertuig routing probleem, gegeven een specifiek aantal voertuigen. ACS-VEI gebruikt een IN array om kolonies aan te zetten

nog niet doorzochte gebieden te onderzoeken. Hiermee heeft de ACS-VEI kolonie een grotere kans om valide oplossingen te vinden, wat belangrijk is voor het dynamische voertuig routing probleem. Nadat de ACS-VEI kolonie een valide oplossing met minder voertuigen geproduceerd heeft worden zowel de oplossing als de gebruikte feromoon sporen teruggestuurd aan de *controller*. Vervolgens stopt de *controller* de kolonies en krijgen ze nieuwe parameter waardes. De kolonies starten weer en zoeken naar een oplossing voor het probleem met nieuw inkomende orders. Voor elk tijdvak wordt deze loop herhaald, totdat de hele werkdag ingevuld is. Dit algoritme is in staat oplossingen van hoge kwaliteit te produceren voor de dynamische problemen. Een reeks benchmark problemen die in verschillende mate dynamisch zijn worden gegenereerd aan de hand van Solomon's benchmark voor voertuig routing problemen. De resultaten laten zien dat het algoritme goed presteert voor al deze benchmarks.

Voor het dynamische voertuig routing probleem met tijdskaders en meerdere prioriteiten is het grootste verschil het attribuut "prioriteit". In veel praktijkproblemen zijn er klanten met verschillende prioriteit. Klanten met een hoge prioriteit moeten eerst bediend worden, bovendien moet voor diezelfde klanten de vertragingstijd geminimaliseerd worden. Het meervoudige mierenkolonie systeem is geïmplementeerd om het dynamische voertuig routing probleem op te lossen. Om met de prioriteit van klanten om te gaan worden er twee strategieën voorgesteld. De ene strategie bedient eerst de klanten met hoge prioriteit en dan de klanten met lage prioriteit. De andere strategie geeft strafpunten op basis van de vertragingstijd om elke klant te bedienen. De hoeveelheid strafpunten is afhankelijke van de klant, hoe hoger de prioriteit hoe meer strafpunten. Het attribuut prioriteit is toegevoegd aan de eerdere benchmark voor het dynamische voertuig routing probleem met tijdskaders. De resultaten laten zien dat de tweede strategie beter presteert dan de eerste, in tegenstelling tot onze intuïtie.

Het praktijkprobleem dat in deze dissertatie bestudeerd wordt kan gemodelleerd worden als een dynamisch voertuig routing probleem met tijdskaders en meerdere prioriteiten. Het probleem komt voort uit een Nederlands bedrijf dat klanten in heel Nederland bedient. Om het probleem te reduceren wordt er een testscenario in de regio Rotterdam gegenereerd. De verschillen tussen het theoretische benchmark probleem en het praktijkprobleem worden geanalyseerd. De op mieren gebaseerde *solver* wordt aangepast om het handmatige controle systeem dat in het bedrijf gebruikt wordt te vervangen. De resultaten laten zien dat de op mieren gebaseerde *solver* betere oplossingen kan produceren dan het handmatige controle systeem. Intussen worden

er ook lessen getrokken uit de implementatie van het algoritme. De drie belangrijkste punten zijn: iteraties werken; het sleutelwoord is communicatie; mensen zijn belangrijk. De vragenlijst over de ervaringen van de chauffeurs levert ook ideeën op over het verbeteren van het model van het logistiek probleem in de praktijk.

In de praktijk willen besluitvoerders niet alleen een oplossing voor het voertuig routing probleem voor een dag, maar ook een betere win-winstrategie voor het inventaris management en routing probleem. Als toevoeging aan het voertuig routing probleem integreert het inventaris routing probleem het inventaris management. In deze dissertatie wordt het inventaris routing probleem behandeld als een drie-doelstellingen optimalisatie probleem, in tegenstelling tot het klassieke twee-doelstellingen probleem. Het klassieke inventaris probleem heeft als doel om gelijktijdig het inventaris niveau en de routing kosten in de gegeven periode te minimaliseren. Hier worden de kosten bij een lege voorraad als derde doelstelling meegenomen in het probleem. Aangezien de toekomstige vraag van klanten niet altijd bekend is valt een lege voorraad in de praktijk niet altijd te voorkomen.

Om het inventaris routing probleem op te lossen wordt het *Multi-objective Optimization Cooperative Particle Swarms* (MOCOPS) algoritme in deze dissertatie voorgesteld. SMS-EMOA is een *state-of-the-art multi-objective optimization* algoritme dat gebruikt wordt om te vergelijken met het nieuw voorgestelde MOCOPS algoritme. De *hypervolume* bijdrage indicator wordt gebruikt als selectiecriterium voor het algoritme, dat probeert het *hypervolume* van het *Pareto front* te maximaliseren voor het inventaris routing probleem. De voorgestelde algoritmes worden ook vergeleken met NSGA-II en HGS14 op de benchmark. Het experiment wordt vijf keer uitgevoerd met een evaluatiebudget van 5000 voor de doel-functies. Het twee-doelstellingen inventaris routing probleem benchmark wordt gebruikt om de efficiëntie van het voorgestelde algoritme te testen. Aan de hand van de resultaten is te zien dat MOCOPS goede oplossingen kan produceren die vergelijkbaar zijn met die van SMS-EMOA, en beter dan die van NSGA-II en HSG14. Ook voor het drie-doelstellingen inventaris routing probleem geldt dat MOCOPS en SMS-EMOA vergelijkbare *attainment surfaces* produceren. Het is echter lastig de verschillen tussen deze *attainment surfaces* te analyseren. Om deze algoritmes te vergelijken worden de *hypervolume* indicatoren berekend. De analyse van de resultaten laat zien dat HGS14 beter presteert op kleinschalige inventaris routing problemen. Op grootschalige problemen laat MOCOPS echter de beste prestaties zien.