Real-time adaptive algorithms for dial-a-ride problems

Pandi, Ramesh Ramasamy

2020

Pandi, R. R. (2020). Real-time adaptive algorithms for dial-a-ride problems. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/144558>

<https://doi.org/10.32657/10356/144558>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 22:45:23 SGT

Real-time Adaptive Algorithms for Dial-A-Ride Problems

Ramesh Ramasamy Pandi

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2020

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

15/06/2020

.....

Date

.....

Ramesh Ramasamy Pandi

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

15/06/2020

.....

Date

.....

Associate Professor Justin Dauwels

Authorship Attribution Statement

Please select one of the following; *delete as appropriate:

*(A) This thesis does not contain any materials from papers published in peer-reviewed journals or from papers accepted at conferences in which I am listed as an author.

*(B) This thesis contains material from [x number] paper(s) published in the following peer-reviewed journal(s) / from papers accepted at conferences in which I am listed as an author.

Please amend the typical statements below to suit your circumstances if (B) is selected.

Chapter 3 is published as

(1) **Ramesh Ramasamy Pandi**, Song Guang Ho, Sarat Chandra Nagavarapu, and Justin Dauwels. *A Generic GPU-Accelerated Framework for the Dial-A-Ride Problem*. IEEE Transactions on Intelligent Transportation Systems, DOI: 10.1109/TITS.2020.2992560.

(2) **Ramesh Ramasamy Pandi**, Song Guang Ho, Sarat Chandra Nagavarapu, Twinkle Tripathy, and Justin Dauwels. *A GPU-Accelerated Tabu Search Algorithm for Dial-A-Ride Problem*. Proceedings of the 21st IEEE Intelligent Transportation Systems Conference (ITSC), Hawaii, USA, 2018.

The contributions of the co-authors in published material are as follows:

- Prof. Justin Dauwels (1,2) provided the initial project direction, guidance and comments on manuscript drafts.
- I (1,2) designed the study, coded, and performed all the experimental works.
- I (1,2) prepared the manuscript drafts, tables and figures.
- Song Guang Ho (1,2) assisted in plotting the graphs in the publication using matlab.
- I revised the manuscript drafts together with Song Guang Ho (1,2), Sarat Chandra Nagavarapu (1,2), Twinkle Tripathy (2), and Prof. Justin Dauwels (1,2).

Chapter 4 is published as

(1) **Ramesh Ramasamy Pandi**, Song Guang Ho, Sarat Chandra Nagavarapu, Twinkle Tripathy, and Justin Dauwels. *Disruption Management for Dial-A-Ride Systems*, In IEEE Intelligent Transportation Systems Magazine, DOI: 10.1109/

MITS.2020.3014429.

(2) **Ramesh Ramasamy Pandi**, Song Guang Ho, Sarat Chandra Nagavarapu, and Justin Dauwels. *Adaptive Algorithm for Dial-A-Ride Problem with Vehicle Breakdown*. Proceedings of the 18th European Control Conference (ECC), Naples, Italy, 2019.

The contributions of the co-authors in published material are as follows:

- Prof. Justin Dauwels (1,2) provided guidance and comments on manuscript drafts.
- I (1,2) designed the study, coded, and performed all the experimental works.
- I (1,2) prepared the manuscript drafts, tables and figures.
- Song Guang Ho (1,2) assisted in plotting the graphs in the publication using matlab.
- I (1,2) revised the manuscript drafts together with Song Guang Ho (1,2), Sarat Chandra Nagavarapu (1,2), Twinkle Tripathy (1), and Prof. Justin Dauwels (1,2).

Chapter 5 is published as

(1) **Ramesh Ramasamy Pandi** and Justin Dauwels. *The Electric Autonomous Dial-A-Ride Problem with Mobile Energy Disseminators*. Under review. IEEE Transactions on Intelligent Transportation Systems.

(2) **Ramesh Ramasamy Pandi**, Song Guang Ho, Sarat Chandra Nagavarapu, and Justin Dauwels. *Deterministic Annealing for Depot Optimization: Applications to the Dial-A-Ride Problem*. Proceedings of the 9th IEEE Symposium Series on Computational Intelligence (SSCI), Bangalore, India, 2018.

The contributions of the co-authors in published material are as follows:

- Prof. Justin Dauwels (1,2) provided guidance and comments on manuscript drafts.
- I (1,2) designed the study, coded, and performed all the experimental works.
- I (1,2) prepared and revised the manuscript drafts including all the tables and figures.
- Song Guang Ho (2) assisted in plotting the graphs in the publication using matlab.
- I (1,2) revised the manuscript drafts together with Song Guang Ho (2), Sarat Chandra Nagavarapu (2), and Prof. Justin Dauwels (1,2).

15/06/2020

.....

Date

.....

Ramesh Ramasamy Pandi

Acknowledgements

First, I would like to express my sincere gratitude to my Ph.D. advisor, Professor *Justin Dauwels* for his support and guidance throughout my journey at Nanyang Technological University (NTU). I am extremely grateful to Prof Dauwels for providing me the opportunity to work as a research student, and helping me to discover my research potentials. The research training provided by Prof Dauwels has not only improved my scientific, communication, and research skills over the last four years of my life but also helped me to gain great confidence for the career progression as an independent scientist.

Special thanks to *Sarat Chandra Nagavarapu* for his continuous support, encouragement, and fruitful discussions. Special thanks to *Song Guang Ho* for the brainstorming sessions and the sleepless manuscript revision nights spent together with Sarat. I thank *Twinkle Tripathy* for inspiring me to do high-quality research. I thank *Prakash Khunti* for the political chit-chats during lunch breaks and for introducing me to timeless books such as the autobiography of a yogi and steve jobs, which transformed my perspective of life. I thank the Singapore government and NTU management for offering a world-class infrastructure to conduct research. I thank my *TEDxNTU* team for all the dynamic and wild adventures in Singapore and Bali.

I thank my friend, *Vivek Chakaravarthi*, who encouraged and believed in my decision to pursue higher studies when the rest of the world was against me. Thank you for standing beside me especially during the toughest times of my life for more than a decade now. I thank my friend, *Bollu Sathish Kumar*, for introducing me to plenty of exciting opportunities and constantly supporting me since we met at the bachelor's degree. They gave me the confidence to discover the aspiring scientist in me. I thank *Janani Gopiravi* for teaching the lesson of my life: "kill them with your success and bury them with your smile", without which I wouldn't have come this far in life. I thank my friends in Singapore who made this country feel home:

my FIFA trainer, *Abishek Sethupandi*, my fitness consultant, *Thanipatti Vignesh*, and my cooking coaches, *Thanipatti Maruthu Machan* and *Prasanna*. The championship games on Sunday nights were a great refreshment to energetically restart my research on Monday mornings. I thank *Ganesh Sai* for nobly sending his GATE notes from Andhra Pradesh to Madurai without expecting anything in return. I thank *Chandrakant Mishra* for his genius mentorship in FB gate groups.

I would like to thank my mother, *V Nagarani*, for the unconditional love, sacrifices, and support that allowed me to pursue my dream of advancing frontiers in science. I thank my father, *R Pandi*, my twin-sisters, *Rama*, and *Latha*, for their never-ending love. I am grateful to my uncle, *Rajkumar*, and my grandparents for all their love and unconditional support. Last, but not least, I thank my Sayali, *Preethy Sudhan*, who came into my life like a bolt of lightning, and for whom I constantly try to make myself deserving of her fierce love.

Note: This PhD thesis was submitted for examination on 15th June 2020, revised on 4th September 2020, and the Oral Examination was completed on 26th October 2020 at the School of Electrical and Electronic Engineering (EEE), Nanyang Technological University (NTU), Singapore.

Abstract

Autonomous Mobility-on-Demand, a key step towards sustainable future urban mobility, refers to a transformative and rapidly developing mode of transportation wherein self-driving vehicles transport passengers in a given environment. Dial-a-ride problem (DARP) is an underlying optimization problem in the operational planning of mobility-on-demand systems. The primary objective of DARP is to design routes and schedules to perform passenger transportation services with a high-level of user comfort. DARP often arises in dynamic real-world scenarios, where rapid route planning is crucial.

Despite the advancements in technologies, autonomous mobility-on-demand systems are still being considered to be in the trial phase, as the commercial deployment is impeded by the following research gaps.

First, the traditional CPU-based algorithms are generally too slow to be useful in practice. There is a growing interest in fast methods since the users expect the service providers to provide an answer quickly, particularly in case of dynamic and disruptive scenarios. Second, regardless of the technological advancements, vehicle breakdowns lead to major disruption of transportation services across the globe. In the context of dial-a-ride systems, this results in an inflated operational cost and damaged reputation for the service providers (operators of taxis, private buses, etc.). However, modeling such dynamic events for dial-a-ride systems has not been addressed in the literature. Third, the advent of autonomous electric vehicles (EV) has the potential to revolutionize intelligent transportation systems and especially useful in mobility-on-demand services to combat issues such as global warming, fuel consumption, air pollution, and road safety. Despite the importance of EVs, the existing distribution of static charging stations (SCS) is not abundant and reachable, which causes range anxiety. The latest charging technology, mobile energy disseminator (MED), is energy-efficient and able to recharge EV on the move without causing any detours. Nevertheless, to promote the usage of autonomous

EVs and achieve green transportation, it is necessary to integrate them into the daily transit systems.

Therefore, this thesis introduces real-time adaptive algorithms for the dial-a-ride system. The following are the major contributions. First, a generic GPU framework has been proposed to accelerate time-critical neighborhood exploration of local search operations under the guidance of meta-heuristics to efficiently solve DARP. Specifically, device-oriented optimization strategies are introduced to enhance the utilization of a current-generation GPU architecture (Tesla P100). Second, models and algorithms for the dial-a-ride problem with the vehicle breakdown. Specifically, we propose a disruption management framework with the combination of online optimization and fleet size minimization. Third, models and algorithms for the electric autonomous dial-a-ride problem with mobile energy disseminators. We investigate the effectiveness of the proposed recharging strategy for electric autonomous DARP in comparison with the conventional SCS.

Simulations are conducted on DARP benchmark instances and real-world Uber taxi trips in San Francisco City, USA. The proposed GPU-based solution methodologies perform 10x to 100x faster than their CPU counterpart and achieve better solution quality in a short time when compared to the existing approaches. Furthermore, when compared to the conventional disruption management techniques, the proposed disruption management scheme achieves a significant reduction in fleet idle time under normal operations and a reduction in operational cost under disruption in dial-a-ride operations. In terms of the EV charging methodologies, the latest mobile energy disseminators that act as a moving charging station seems to improve the operational service quality of electric autonomous dial-a-ride systems when compared to conventional recharging methodologies.

In summary, we introduce novel formulations and solution frameworks to advance the state-of-the-art in DARP and conduct simulations on both benchmark and real-world test instances to verify the suitability and effectiveness of the proposed frameworks. We report the effect of GPU on accelerating solution algorithms for solving DARP under dynamic real-life settings, the effect of minimizing fleet size on the service provider and customer experience both before and after disruptive events such as vehicle breakdown, the effect of optimizing depot locations on transportation costs, and the effect of the latest recharging infrastructures on reducing

the EV range anxiety in dial-a-ride operations. At the end of the thesis, we provide concluding remarks and several interesting future research directions.

Contents

Acknowledgements	xi
Abstract	xv
List of Figures	xxv
List of Tables	xxix
Acronyms	xxxiii
1 Introduction	1
1.1 Motivation	2
1.2 Research gaps	4
1.2.1 GPU-based optimization algorithms to efficiently solve DARP	4
1.2.2 Disruption management schemes for DARP with vehicle break-down	5
1.2.3 DARP with electric autonomous vehicles and modern recharging infrastructures	6
1.3 Aim and Objective of the Thesis	7
1.4 Overview of the Contributions and Thesis Structure	9
2 The Dial-A-Ride Problem: A Review of Formulations and Algorithms	15
2.1 Introduction	16
2.2 Problem classifications	19
2.2.1 Definition of DARP	19
2.2.2 Taxonomy	21
2.2.3 Advanced Service Design	26
2.2.3.1 Heterogeneity	26
2.2.3.2 Routing properties	32
2.2.3.3 Service quality	33
2.2.3.4 Green transport	33
2.2.4 Objective functions	34
2.2.4.1 Single objective	34

2.2.4.2	Multiple objectives	34
2.2.5	Stochastic/Dynamic factors	35
2.2.5.1	Transportation times	35
2.2.5.2	Customer requests	36
2.2.5.3	User characteristics	37
2.3	Solution Methods	37
2.3.1	Taxonomy	37
2.3.2	Exact Methods	39
2.3.2.1	Branch-and-cut	39
2.3.2.2	Column generation	41
2.3.2.3	Branch-(cut)-and-price	41
2.3.2.4	Dynamic programming	41
2.3.3	Approximation Methods	41
2.3.3.1	Insertion Heuristics	42
2.3.3.2	Local-search based Metaheuristics	42
2.3.3.3	Population-based Metaheuristics	45
2.3.3.4	Hybrid solution approaches	46
2.3.4	Parallel Computing Approaches	46
2.4	Discussion	47
2.5	Knowledge Gaps and Research Challenges	48
3	A Generic GPU-Accelerated Framework for Dial-A-Ride Problem	53
3.1	Introduction	54
3.2	Relevant Literature	56
3.2.1	Relevant studies on DARP	56
3.2.2	Relevant studies on GPU-based methods	58
3.2.3	Research gap	60
3.2.4	Our Contributions	61
3.3	Formulation of Dial-A-Ride Problem	62
3.4	Proposed generic GPU framework	65
3.4.1	Class structure	66
3.4.2	Local search operation	69
3.4.3	CUDA kernels	70
3.5	Proposed Device Optimization Strategies	72
3.5.1	Device Hierarchical Memory Optimization	73
3.5.2	Adaptive Kernel Grid Configuration	75
3.5.3	Optimizing Data Transfer	77
3.5.4	Reducing Control Divergence	78
3.5.5	Enhancing Kernel Launch Throughput	79
3.6	Integrating the proposed GPU framework with Metaheuristics	81
3.6.1	Overview of the GPU-based metaheuristics	81
3.6.2	Neighborhood move	81
3.6.3	Acceptance criteria to iteratively select the best solution	82

3.6.4	Allowance of infeasible solutions	82
3.6.5	Implementation details	83
3.6.5.1	GPU-based Tabu Search Heuristic	83
3.6.5.2	GPU-based Variable Neighborhood Search	84
3.6.5.3	GPU-based Adaptive Large Neighborhood Search	86
3.7	Simulation Results	89
3.7.1	Simulation settings	89
3.7.2	Effect of different components on computational speedup for the proposed GPU methodology	91
3.7.3	Comparison of solution quality of our approach with the existing results	93
3.8	Discussion	107
4	Disruption Management Framework for Dial-A-Ride Problem	111
4.1	Introduction	112
4.1.1	Our Contributions	114
4.2	Formulation of Dial-A-Ride Problem with Vehicle breakdown	115
4.3	Proposed Disruption Management Framework	119
4.3.1	Overall Framework	119
4.3.2	G-ALNS Optimization Framework	123
4.3.2.1	Fleet size minimization	125
4.3.2.2	Implementation details	125
4.4	Disruption management techniques	126
4.4.1	Benchmark 1: Solution before breakdown / Benchmark solution	126
4.4.2	Method A: Solution after breakdown/ No action taken	126
4.4.3	Method B: Conventional method (send a rented vehicle in predefined route)	127
4.4.4	Method C: Reoptimized solution using G-ALNS with a rented vehicle	127
4.4.5	Benchmark 2: Solution before breakdown / Benchmark solution with fleet size minimization	127
4.4.6	Method D: Solution after breakdown/ No action taken	127
4.4.7	Method E: Conventional method (send an available/rented vehicle in predefined route)	128
4.4.8	Method F: Reoptimized solution using G-ALNS-FSM with a rented vehicle	128
4.5	Simulation Results	128
4.5.1	Simulation settings	129
4.5.2	Operational cost function	130
4.5.3	Comparison of solution quality with the conventional method	136
4.5.4	Effectiveness of the proposed approach on passenger experiences	141
4.5.5	Policy implications on the key performance indicators	143

4.6	Discussion	144
5	The Dial-A-Ride Problem with Electric Autonomous Vehicles and Modern Recharging Infrastructures	147
5.1	Introduction	148
5.1.1	Our Contributions	154
5.2	Formulation of Electric Autonomous Dial-A-Ride Problem	155
5.3	Solution Methodology and Scheduling heuristic	159
5.3.1	Proposed Solution Framework	160
5.3.2	Proposed Scheduling Algorithm	164
5.3.2.1	Layer 1	164
5.3.2.2	Layer 2	164
5.3.2.3	Layer 3	165
5.3.3	Recharging Infrastructures	166
5.3.3.1	Static Charging Station	166
5.3.3.2	Mobile Energy Disseminators	166
5.4	Simulation Results	166
5.4.1	Simulation settings	167
5.4.2	Comparison of solution quality with the existing results	169
5.4.3	Effectiveness of the proposed e-ADARP with MED	173
5.4.4	Effectiveness of various recharging infrastructures on passenger experiences for e-ADARP	179
5.5	Discussion	181
6	Conclusion and Future Directions	183
6.1	Conclusion and Summary of the Contributions	183
6.2	Future Directions	187
6.2.1	Peer-to-peer EV recharging scheme for electric vehicle dial-a-ride systems	187
6.2.2	Modeling frameworks and accelerated algorithms for real-world emergency dial-a-ride operations:	189
6.2.3	Routing a mix of vehicular technologies and recharging stations with new energy consumption models:	190
6.2.4	Accelerating the exact solution methods in GPU and solving multi-objective DARPs:	192
6.2.5	Improving the methodological aspects of the proposed GPU framework:	194
A	Detailed Analysis of Computational Speed-up for Various GPU-based Metaheuristics	197
B	Detailed Analysis of Fleet Size Minimization on Passenger Experiences	201

C Detailed Solutions for Electric Autonomous Dial-A-Ride Problems	217
D Detailed Analysis of Depot Optimization in Dial-A-Ride Problems	223
List of Author's Publications	233
Bibliography	235

List of Figures

1.1	Thesis flow.	10
2.1	Classification: Mass Transport Systems [1].	17
2.2	Taxonomy: Percentage of studies that consider IC and EV fleet. . .	23
2.3	Taxonomy: Percentage of studies that consider single and multiple depots.	23
2.4	Taxonomy: Percentage of studies that consider single and multiple trips to depot.	23
2.5	Taxonomy: Percentage of studies that consider single and multiple vehicles.	23
2.6	Taxonomy: Percentage of studies that consider homogeneous and heterogeneous fleet.	23
2.7	Taxonomy: Percentage of studies that consider seating capacity con- straint.	23
2.8	Taxonomy: Percentage of studies that consider user time window constraint.	24
2.9	Taxonomy: Percentage of studies that consider passenger ride time constraint.	24
2.10	Taxonomy: Percentage of studies that consider vehicle route dura- tion constraint.	24
2.11	Taxonomy: Percentage of studies that consider selective visit to cus- tomers.	24
2.12	Taxonomy: Percentage of studies that consider single objective func- tion.	24
2.13	Taxonomy: Percentage of studies that consider static and dynamic factors.	24
2.14	Taxonomy: Percentage of studies that consider deterministic and stochastic factors.	25
2.15	Taxonomy: Percentage of studies that consider optimizing depot visits.	25
2.16	Taxonomy: Percentage of DARP studies in various time-period con- sidered in our review.	25
2.17	Taxonomy: Percentage of studies that utilize exact solution methods.	38
2.18	Taxonomy: Percentage of studies that utilize heuristic-based solu- tion methods.	38

2.19	Taxonomy: Percentage of studies that utilize GPU-based solution methods.	39
3.1	Memory mapping on the GPU.	74
3.2	Optimized data transfer between CPU and GPU.	78
3.3	Concurrent kernel execution with slicing.	80
3.4	(A) Comparison of solution quality (<i>gap</i>) with objective function f' between Proposed and Existing methods on DARP instances after 30 sec of CPU time.	101
3.5	(B) Comparison of solution quality (<i>gap</i>) with objective function f' between Proposed and Existing methods on DARP instances after 30 sec of CPU time.	101
3.6	(C) Comparison of solution quality (<i>gap</i>) with objective function f'' between Proposed and Existing methods on DARP instances after 30 sec of CPU time.	102
3.7	(D) Comparison of solution quality (<i>gap</i>) with objective function f'' between Proposed and Existing methods on DARP instances after 30 sec of CPU time.	102
3.8	(A) Comparison of solution quality (<i>gap</i>) with objective function f' between Proposed and Existing methods on DARP instances after 60 sec of CPU time.	103
3.9	(B) Comparison of solution quality (<i>gap</i>) with objective function f' between Proposed and Existing methods on DARP instances after 60 sec of CPU time.	103
3.10	(C) Comparison of solution quality (<i>gap</i>) with objective function f'' between Proposed and Existing methods on DARP instances after 60 sec of CPU time.	104
3.11	(D) Comparison of solution quality (<i>gap</i>) with objective function f'' between Proposed and Existing methods on DARP instances after 60 sec of CPU time.	104
3.12	(A) Comparison of solution quality (<i>gap</i>) with objective function f' between Proposed and Existing methods on DARP instances after 180 sec of CPU time.	105
3.13	(B) Comparison of solution quality (<i>gap</i>) with objective function f' between Proposed and Existing methods on DARP instances after 180 sec of CPU time.	105
3.14	(C) Comparison of solution quality (<i>gap</i>) with objective function f'' between Proposed and Existing methods on DARP instances after 180 sec of CPU time.	106
3.15	(D) Comparison of solution quality (<i>gap</i>) with objective function f'' between Proposed and Existing methods on DARP instances after 180 sec of CPU time.	106
4.1	Real-time adaptive algorithm for the dial-a-ride system.	120

4.2	Case 1: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R1a Instance for DDARP-VB.	131
4.3	Case 2: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R2a Instance for DDARP-VB.	132
4.4	Case 3: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R3a Instance for DDARP-VB.	132
4.5	Case 4: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R4a Instance for DDARP-VB.	133
4.6	Case 5: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R5a Instance for DDARP-VB.	133
4.7	Case 6: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R6a Instance for DDARP-VB.	134
4.8	Case 7: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R7a Instance for DDARP-VB.	134
4.9	Case 8: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R8a Instance for DDARP-VB.	135
4.10	Case 9: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R9a Instance for DDARP-VB.	135
4.11	Case 10: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R10a Instance for DDARP-VB.	136
5.1	EV-recharging mechanism: Station charging station (SCS)	149
5.2	EV-recharging mechanism: Battery swapping station (BSS)	149
5.3	EV-recharging mechanism: Mobile charging (MC)	150
5.4	EV-recharging mechanism: Mobile energy disseminator (MED)	150
5.5	Best route plan for a2-16-0.7 e-ADARP instance.	170
5.6	Best route plan with SCS visits for a2-16-0.7 e-ADARP instance.	171
5.7	Best route plan with MED visits for a2-16-0.7 e-ADARP instance.	172
5.8	Legends.	172
6.1	Peer-to-Peer EV-recharging mechanism [2].	188
6.2	A general framework for real-world emergency routing operations.	189
6.3	Calculation of required energy for the mixed fleet.	191
6.4	GPU-accelerated branch-and-bound framework [3].	193
6.5	Cuda Dynamic Parallelism [4].	194

B.1	Fleet size for various instances.	205
B.2	Travel Cost for various instances.	206
B.3	DARP instance Set PR: Relationship between travel cost and fleet size.	208
B.4	DARP instance Set A: Relationship between travel cost and fleet size.	209
B.5	DARP instance Set B: Relationship between travel cost and fleet size.	210
B.6	Vehicle Route Duration: Quantitative analysis of solutions for various instances.	211
B.7	Vehicle Idle Time: Quantitative analysis of solutions for various instances.	212
B.8	Passenger Ride Time: Quantitative analysis of solutions for various instances.	213
B.9	Passenger Idle Time: Quantitative analysis of solutions for various instances.	214
B.10	Vehicle Early Arrival Time: Quantitative analysis of solutions for various instances.	215
B.11	Vehicle Late Arrival Time: Quantitative analysis of solutions for various instances.	216
D.1	Flowchart: Bi-level optimization framework for solving DARP with depot optimization.	229
D.2	Small-scale instance: BKS for b2-16 instance.	230
D.3	Small-scale instance: DA for b2-16 instance.	230
D.4	Medium-scale instance: BKS for pr01 instance.	231
D.5	Medium-scale instance: DA for pr01 instance.	231
D.6	Large-scale instance: BKS for pr12 instance.	232
D.7	Large-scale instance: DA for pr12 instance.	232

List of Tables

2.1	Taxonomy: Problem classifications for DARP (part 1)	27
2.2	Taxonomy: Problem classifications for DARP (part 2)	28
2.3	Taxonomy: Problem classifications for DARP (part 3)	29
2.4	Taxonomy: Problem classifications for DARP (part 4)	30
2.5	Taxonomy: Problem classifications for DARP (part 5)	31
2.6	Taxonomy: Exact methods for solving DARP	40
2.7	Taxonomy: Approximation methods for solving DARP	43
3.1	Terminology.	63
3.2	Device hierarchical memory optimization	76
3.3	Representation of the methodologies.	91
3.4	Effect of different components on computational speed up for the G-TS methodology.	94
3.5	Effect of different components on computational speed up for the G-VNS methodology.	95
3.6	Effect of different components on computational speed up for the G-ALNS methodology.	96
3.7	Percentage deviation from the best speed up for the G-TS methodology.	97
3.8	Percentage deviation from the best speed up for the G-VNS methodology.	98
3.9	Percentage deviation from the best speed up for the G-ALNS methodology.	99
3.10	Comparison of solution quality (<i>gap</i>) of our approach (<i>P</i>) when compared to the existing methods (<i>E</i>) after 30 s of CPU time.	100
3.11	Comparison of solution quality (<i>gap</i>) of our approach (<i>P</i>) when compared to the existing methods (<i>E</i>) after 60 s of CPU time.	107
3.12	Comparison of solution quality (<i>gap</i>) of our approach (<i>P</i>) when compared to the existing methods (<i>E</i>) after 180 s of CPU time.	108
4.1	Disruption Management: Representation of the methodologies. . . .	137
4.2	R1a Instance: Results attained by the G-ALNS for DDARP-VB. Best result out of five random runs are reported.	138
4.3	R1a Instance: Results attained by the G-ALNS with fleet size minimization (G-ALNS-FSM) for DDARP-VB. Best result out of five random runs are reported.	139

4.4	Results for various proposed recourse methods (G-ALNS, G-ALNS-FSM) and conventional method (CM) on solving DDARP-VB for five random runs with stopping criterion as 30 sec. Reduction in operational cost is achieved by the G-ALNS-FSM on both before and after vehicle breakdown.	140
4.5	Influence of vehicle breakdown and effectiveness of the proposed algorithm on passenger experience. Comparison of user ride times for G-ALNS vs G-ALNS-FSM for DARP. Reduction in user ride time is achieved by our G-ALNS-FSM.	141
4.6	Details of the solution attained by G-ALNS-FSM on tested instances. G-ALNS-FSM attains significant fleet savings at a reasonable increase in detour distance when compared to the benchmark solutions [5] without fleet size minimization.	142
4.7	Policy implications: Average savings with fleet size minimization by G-ALNS and G-TS [6] when compared to the benchmark solutions [5] without fleet size minimization for all the tested DARP instances.	143
5.1	Survey of EV charging methodologies in the context of vehicle routing problem (VRP).	151
5.2	Terminology.	155
5.3	Computational speedup attained by ALNS on GPU vs CPU for e-ADARP. Stopping criterion is 10^3 iterations. The value of r refers to the final minimum battery level ratio.	171
5.4	Benchmark instances: Comparison of solution quality attained by the proposed method for e-ADARP with SCS and MED vs. existing method for e-ADARP.	173
5.5	Uber instances: Comparison of solution quality attained by the proposed approach vs. existing method for e-ADARP.	174
5.6	Benchmark instances: Policy implications: Effect of MED fleet size on solution quality for e-ADARP with MED.	175
5.7	Uber instances: Policy implications: Effect of MED fleet size on solution quality for e-ADARP-MED with $r=0.1$	176
5.8	Uber instances: Policy implications: Effect of MED fleet size on solution quality for e-ADARP-MED with $r=0.4$	177
5.9	Uber instances: Policy implications: Effect of MED fleet size on solution quality for e-ADARP-MED with $r=0.7$	178
5.10	Benchmark instances: Effectiveness of the proposed e-ADARP-MED system on passenger experiences, fleet idle time, etc.	179
5.11	Uber instances: Effectiveness of the proposed e-ADARP-MED system on passenger experiences, fleet idle time, etc.	180
A.1	Magnitude difference from the best speed up for the G-TS methodology.	198
A.2	Magnitude difference from the best speed up for the G-VNS methodology.	199

A.3	Magnitude difference from the best speed up for the G-ALNS methodology.	200
B.1	Results on imposing fleet size minimization on PR DARP instances of [7].	202
B.2	Results on imposing fleet size minimization on Set A DARP instances of [8].	203
B.3	Results on imposing fleet size minimization on Set B DARP instances of [8].	204
B.4	Improvement attained by the fleet size minimization scheme when compared to the best-solutions without fleet size minimization [9][10][11][12] on single-objective DARP.	207
C.1	Benchmark instances: Details of the best solution attained by ALNS[GPU] for e-ADARP with SCS.	218
C.2	UBER instances: Details of the best solution attained by ALNS[GPU] for e-ADARP with SCS.	219
C.3	Benchmark instances: Details of the best solution attained by ALNS[GPU] for e-ADARP with MED.	220
C.4	UBER instances: Details of the best solution attained by ALNS[GPU] for e-ADARP with MED.	221
D.1	Results attained by the proposed DA on small-scale DARP instances [8].	225
D.2	Results attained by the proposed DA on medium-scale DARP instances [7][8].	226
D.3	Results attained by the proposed DA on large-scale DARP instances [7].	227
D.4	Comparison of results attained by the proposed DA vs BKS [9][8][10] on DARP instances [7][8].	228

Acronyms

AV	Autonomous Vehicle
ALNS	Adaptive Large Neighborhood Search
B&B	Branch-and-Bound
B&C	Branch-and-Cut
BSS	Battery Swapping Station
CPU	Central Processing Unit
CDP	Cuda Dynamic Parallelism
CUDA	Compute Unified Device Architecture
CACC	Cooperative Adaptive Cruise Control
DP	Dynamic Programming
DAR	Dial-A-Ride
DWC	Dynamic Wireless Charging
DRT	Demand Responsive Transit
DART	Dial-A-Ride Transit
DARP	Dial-A-Ride Problem
DDARP-VB	Disruptive Dial-A-Ride Problem with Vehicle Breakdown
EV	Electric Vehicle
ELS	Evolutionary Local Search
e-AV	Autonomous Electric Vehicle
e-ADARP	Electric Autonomous Dial-A-Ride Problem
FMS	Fleet Management Strategy
FSM	Fleet Size Minimization
GA	Genetic Algorithm
GPU	Graphics Processing Unit
G-TS	GPU-Accelerated Tabu Search
G-VNS	GPU-Accelerated Variable Neighborhood Search
G-ALNS	GPU-Accelerated Adaptive Large Neighborhood Search

GRASP	Greedy Randomized Adaptive Search Procedure
IH	Insertion Heuristic
IC	Internal Combustion Engine
IPT	Inductive Power Transfer
INLP	Integer Non-linear Programming
LS	Local Search
LNS	Large Neighborhood Search
MC	Mobile Charging
MED	Mobile Energy Disseminator
MTC	Montreal Transit Commission
MOD	Mobility-On-Demand
MILP	Mixed Integer Linear Programming
NP	Non-deterministic Polynomial Time
P2P	Peer-to-Peer Charging
PDP	Pickup and Delivery Problem
PRT	Passenger Ride Time
PDPTW	Pickup and Delivery Problem with Time Windows
SV	Selective Customer Visits
SCS	Static Charging Station
SPRP	Shorest Path Routing Problem
TW	Time Window
TS	Tabu Search
VNS	Variable Neighborhood Search
VRD	Vehicle Route Duration
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows

Chapter 1

Introduction

This thesis presents modeling and solution frameworks for routing and scheduling operations of the dial-a-ride transportation. The approach enables a group of vehicles to complete the pickup and delivery of passengers while satisfying the constraints of the fleet operator and passenger convenience. Besides, the approach is effectively able to address disruptive events such as vehicle breakdown and ensures optimized scheduling of visits to modern recharging infrastructures during the employment of electric autonomous vehicles. Applications of DARP include public and commercial transit, door-to-door transportation for elderly or disabled, medical emergency transportation, ground, and seaport operations.

As a motivating example, consider the following scenario: a dial-a-ride system is designed to respond to the passenger's transportation requests in real-time. A fleet of vehicles is available for serving the customer demand and provide flexible transportation service. The schedules and the sequence in which the vehicles visit customer pickup and drop-off locations need to be designed while respecting the customer convenience and service provider's resources. In today's ride-sharing transportation scenarios, the customers expect a cheaper travel cost in a short request response time while the fleet operator prefers efficient or profitable solutions, which may require a large computation time. Due to the NP-hard computational complexity of the DARP, the existing sequential optimization algorithms may not be applicable in practice. Besides, in many real-life scenarios, fleet operations are often hindered by the occurrence of many unanticipated stochastic events, such as

vehicle breakdown, accidents, drastic weather conditions, etc., which in turn diminish the solution quality. Efficient disruption management schemes and fast on-line optimization algorithms are necessary to dynamically insert new trips into routes that are already on-going. Furthermore, when using modern vehicular technologies such as electric and autonomous vehicles, efficiently scheduling the vehicles to visit the recharging stations and analyzing the benefits of the modern recharging technologies are necessary to feasibly complete the transportation service. Therefore, the problem is to design modeling frameworks and solution algorithms to enable real-time decision-making capability in vehicle routing for dial-a-ride operations.

Accordingly, this thesis presents efficient models and algorithms that typically blend ideas from the fields of computational intelligence, combinatorial optimization, and high-performance computing. This thesis presents the first results of GPU-accelerated solution methods for DARP. The key novelty in our approach is the simultaneous introduction of GPU-based optimization methods, modeling frameworks, disruption schemes, recharging mechanisms, and scheduling algorithms for dial-a-ride operations.

In this chapter, we first describe the context of this thesis and its relevance to real-world problems (Section 1.1). Second, we describe the potential research gaps (Section 1.2) associated with the core problem in this thesis. Third, we present the objectives of the thesis and the research questions addressed in the thesis (1.3). Fourth, we discuss the thesis structure and outline the contributions in each chapter (Section 1.4).

1.1 Motivation

Smart urban mobility is one of the prime areas of focus for most of the countries nowadays to fulfill people's daily transportation needs, enhance travel comfort, and provide superior user convenience. Rapid advancements in the areas of autonomous vehicle technology, parallel computing, optimization techniques for vehicle routing, work as the driving forces to achieve this objective. Many day-to-day services such as emergency vehicles, door-to-door people transport, goods delivery, etc. need efficient fleet management solutions developed using these emerging technologies to provide satisfactory service to the customers. Mobility-on-demand (MOD) is a

transformative and rapidly developing mode of transportation wherein the vehicles transport passengers in a given environment. As autonomous vehicular (AV) technologies continue to emerge, it is more likely that the future MOD systems will rely on AVs to fulfill daily transportation needs [13]. Moreover, MOD has been advocated as a key step toward sustainable urban mobility in the 21st-century [14]. These services often involve transportation of human passengers who place transportation requests with an origin and a destination location. The underlying optimization problem is usually modeled in terms of a dial-a-ride problem (DARP), which consists of designing vehicle routes and schedules to perform pick up and delivery of passengers while respecting the passenger convenience.

DARP is an advanced user-oriented form of mobility service that is becoming increasingly relevant in today's public and commercial transit services, emergency transportation, ground, and seaport operations. Dial-a-ride problem (DARP), a special class of vehicle routing problem (VRP), belongs to the category of people transportation, offers pick-up and drop-off services to a group of users between their preferred origin and destination locations, while minimizing the total travel cost subject to a set of constraints such as vehicle load, route duration, passenger ride time and time windows associated with pick-up and drop-off. The advent of electric vehicles (EV) has the potential to revolutionize intelligent transportation systems and especially useful in mobility-on-demand services to combat issues such as global warming, fuel consumption, air pollution, and road safety. As indicated by the reference [15], "94% of the road accidents in the United States of America (USA) are caused by human error, so self-driving vehicles could drastically reduce the fatalities that occur on the roads today". Despite the importance of EVs, the existing distribution of static charging stations (SCS) is not abundant and reachable, which causes range anxiety. The modern charging technology, mobile energy disseminator (MED), is energy-efficient and able to recharge EV on the move without causing any detours. Nevertheless, to promote the usage of autonomous EVs and achieve green transportation, it is necessary to integrate them into the daily transit systems that can be formulated as dial-a-ride problems (DARP).

The primary objective of the dial-a-ride problem with electric autonomous vehicles is to design vehicle routes and schedules to perform passenger transportation service with a high-level of user comfort while satisfying a range of constraints related to the fleet operator, user service quality, and vehicle battery levels. Due to the

high computational complexity of this problem, the traditional CPU-based algorithms may not be applicable in practice. Besides, disruptive events such as vehicle breakdown may result in an inflated operational cost and damaged reputation for the service providers (operators of taxis, private buses, etc.), which requires efficient models and fast algorithms to facilitate real-time optimization. To verify the effectiveness of the algorithms, it is important to conduct simulations on both the DARP benchmark instances and real-world problem instances.

1.2 Research gaps

Despite the advancements in technologies, autonomous mobility-on-demand systems are still being considered to be in the trial phase, as the commercial deployment is impeded by the following research gaps. In Section 1.2.1, we elaborate on the need for accelerating optimization algorithms using the latest computing technologies such as GPU to efficiently solve DARP under dynamic real-life settings. In Section 1.2.2, we explain the need for designing effective models and algorithms for dial-a-ride operations under disruption scenarios such as vehicle breakdown. In Section 1.2.3, we describe the necessity to design effective models and algorithms for dial-a-ride operations with electric autonomous vehicles and the modern recharging infrastructures. We detail the motivation and context of each of these research gaps in the following subsections.

1.2.1 GPU-based optimization algorithms to efficiently solve DARP

There is a growing interest in fast methods since users expect the service providers to provide an answer/solution quickly. This is quite evident from the recent dial-a-ride survey from Sin C Ho et al. [5], which suggests that the users tend to prefer lesser response time in the current world, and the ride-hailing companies have started to react accordingly. The advancements in big data and telemetrics have transformed the dial-a-ride (DAR) systems. State-of-the-art technologies yield large amounts of data about transportation networks. Despite the traditional DAR systems requiring advanced trip booking, often the planned schedule may need to be adjusted on the fly due to unexpected events. In today's world, the sophisticated

machine learning and deep learning approaches can dynamically estimate the travel times much more accurately, faster algorithms and more powerful hardware will allow real-time planning and dispatching. This trend may shift research on fast solution methodologies for the DAR system to facilitate real-time optimization.

The technological advancements are constantly creating new trends to operate DAR services. To achieve the computational speeds necessary, especially in dynamic problems, parallel computation (perhaps using meta-heuristics) is a promising research direction that is currently lacking in the literature for DAR [5]. Particularly, GPU technology has evolved significantly in the past decade with the introduction of the Compute Unified Device Architecture (CUDA) programming model. Much progress has been made on computing technologies over the years, with the invention of high-performance parallel-computing hardware and its usage in non-graphical computationally intensive general-purpose applications. Most of the GPU-based works on routing problems do not show a thorough investigation of fully utilizing GPU (i.e., kernel concurrency, hierarchical device memory optimization, optimizing CPU-GPU communication, reducing thread divergence, adaptive kernel grid configuration). From the scientific perspective, it is critical to design methods that yield better and more robust solutions than existing methods for problems by fully utilizing state-of-the-art computing technology.

Therefore, a generic GPU-accelerated framework needs to be designed for integrating metaheuristics to efficiently solve the dial-a-ride problem.

1.2.2 Disruption management schemes for DARP with vehicle breakdown

Besides designing efficient vehicle routes, another important problem in the fleet management system is the minimum fleet problem. The objective is to determine the minimum number of vehicles required to provide an uninterrupted and high-quality transportation service to the users [16]. According to a recent study on fleet size minimization for taxi services, in New York City alone, taxis travel around the city without serving any passengers for about 40% of the time [17]. With the increased need for transportation, several other issues such as traffic congestion, carbon emissions per vehicle, passenger ride times, etc. also increase significantly.

Furthermore, the fleet depreciation loss, maintenance cost, supporting infrastructure cost, etc. pose an additional burden on the service providers. Thus, there is a great need for fleet size minimization in DAR systems.

Furthermore, regardless of how technology advances, vehicle breakdowns lead to major disruption of transportation services across the globe. In the US alone, a massive 32 million car breakdowns are recorded in 2015 and the numbers are rising further every year [18]. In the context of dial-a-ride systems, breakdowns result in an inflated operational cost and damaged reputation for the service providers (operators of taxis, private buses, etc.). In literature, the dial-a-ride problem with traffic [19] or demand uncertainty [20] has been addressed. The common methodology is to forecast the demand or traffic uncertainty to perform anticipative routing. However, in the case of vehicle breakdown, it is not possible to rely on prediction outputs. The conventional method to handle breakdown is to rent a vehicle and dispatch in a predefined route, which may be a sub-optimal solution. Therefore, there is a great need to develop the following: i) a formulation for the dial-a-ride problem with vehicle breakdown with the realistic constraints and operational cost function that considers fuel, maintenance, parking, rental, cleaning, and wear and tear, ii) a fast solution method to perform recourse action as soon as possible in an event of vehicle breakdown, ii) an efficient fleet management strategy to address vehicle breakdown and provide uninterrupted high-quality transportation service to the passengers.

Therefore, models and algorithms need to be designed for the dial-a-ride problem with a vehicle breakdown.

1.2.3 DARP with electric autonomous vehicles and modern recharging infrastructures

Climate change is a major concern, and it is essential to reduce greenhouse gas emissions. In recent years, one of the primary focuses of the automobile industry has been to come up with alternative vehicular technologies such as electric vehicles to prevent carbon emissions and the use of fossil fuels [21]. According to a survey in 2019 [22], the estimated number of EVs will increase drastically in the future with the estimated decrease in manufacturing cost [23]. Due to the emergence of autonomous vehicles (AV), the future AMoD systems will depend on both AV and

EV to fulfill user transportation. The utilization of autonomous EVs for everyday activities would rise up, only if the charging infrastructure available is made abundant. However, the existing infrastructure to charge these vehicles is either slender or disorganized, and any changes would be very costly. Due to this problem of EV range anxiety, adopting them to on-demand transit systems is a challenging problem, especially in DARP, where user convenience is of supreme importance. Nevertheless, to promote the usage of autonomous EVs and achieve green transportation, it is necessary to integrate them into the daily transit systems.

Existing study on electric autonomous dial-a-ride problem [24] has considered the conventional static charging station (SCS), which has a slender infrastructure that may require costly changes and may cause large detours to recharge EVs. Meanwhile, the modern EV charging technology such as mobile energy disseminator (MED) is superior in terms of efficiency and suitability for DARP, when compared to SCS. Recently, [25] has studied MED in the context of the shortest path routing problem (SPRP) using the Dijkstra method. However, the electric autonomous dial-a-ride problem is very challenging when compared to SPRP due to the range of constraints associated with it. Furthermore, it is essential to develop novel scheduling algorithms to accommodate the modern recharging infrastructures in the solutions for dial-a-ride problems with electric autonomous vehicles. Besides, state-of-the-art metaheuristics should be adapted alongside parallel computing techniques to efficiently solve the electric autonomous dial-a-ride problem

Therefore, models and algorithms need to be designed for the dial-a-ride problem with electric autonomous vehicles and modern recharging infrastructures.

1.3 Aim and Objective of the Thesis

The objective of the thesis is to develop real-time adaptive algorithms for routing and scheduling of autonomous and electric vehicles for dial-a-ride transportation in static and dynamic/disruptive environments. DARP is an advanced user-oriented form of mobility service that is becoming increasingly relevant in today's public and commercial transit services, emergency transportation, ground, and seaport operations. By leveraging on the state-of-the-art GPU computing architectures, and integrating ideas from combinatorial optimization, computational intelligence,

and high-performance computing, we develop a generic framework with models and algorithms for complex real-world dial-a-ride problems. The models of DARP considered in this thesis include the features such as pickup/delivery locations, time window, maximum seating capacity, maximum vehicle route duration, maximum customer ride time, user service time, depot locations, uncertain vehicle breakdown, battery management, and recharging times.

In today's transportation scenarios, the passengers are willing to opt for a cheaper ride and expect a shorter request response time from the service provider. In contrast, large optimization time might be required to achieve a quality solution (with cheaper costs). Since DARP is NP-hard, existing CPU-based algorithms are not useful in practice, particularly when the problem includes richer constraints. It is important to emphasize that the translational challenges in practical DARP applications are to rapidly produce high-quality solutions by utilizing modern computing architectures. Despite the evolution of parallel computing architecture over the past years and the criticality of optimization times in real-time dial-a-ride operations, the effect of GPU-based algorithms for DARP has not been studied in the literature. Along the same lines, efficient models and quicker optimization methods are essential to address more complex and computationally expensive DARP with advanced features (electric and autonomous vehicles) or service disruptions (vehicle breakdown, weather, request cancellation, etc.). Disruption management has been a long-standing problem; disruptive events such as vehicle breakdown cannot be predicted in advance since it is highly uncertain, and it is important to develop efficient models/algorithms to address this problem while ensuring high operational efficiency. Furthermore, next-generation urban mobility is approaching towards autonomous vehicles fuelled by electric power, and therefore analyzing the benefits of modern recharging infrastructures is crucial towards fully achieving a green transit system.

Therefore, the objective of the thesis is to introduce novel models and algorithms that merge ideas from multiple disciplines (combinatorial optimization, computational intelligence, and high-performance computing) for an under-explored yet highly important combinatorial problem of DARP. Specifically, the thesis attempts to provide answers to the following research questions.

- How to fully utilize the state-of-the-art GPU architecture in accelerating the optimization methods for DARP? What is the effect of GPU on accelerating

state-of-the-art optimization methods in terms of computational time and solution quality when compared to existing sequential approaches? (*Refer Chapter 3*)

- How to model and solve the problem of disruption management for dial-a-ride operations? Is it possible to improve the existing fleet management strategies and what are the advantages and drawbacks of such new improvements? (*Refer Chapter 4*)
- What is the effect of electric autonomous vehicles and modern recharging infrastructures such as mobile energy disseminators in dial-a-ride operations? How to model such problems and devise solution methods to enable efficient schedules for autonomous EVs to visit such recharging infrastructures? (*Refer Chapter 5*)

The contributions and the thesis structure are detailed in the next section.

1.4 Overview of the Contributions and Thesis Structure

In this section, we provide an overview of the contributions and structure of the thesis. Figure 1.1 shows the flow of the thesis. This thesis is structured in such a way that the readers should start with Chapters 1–3 sequentially, thereafter Chapters 4 and 5 can be optionally navigated in any order before reaching Chapter 6. The main contribution of the thesis is the development of real-time adaptive algorithms for dial-a-ride problems. In Chapter 2, we provide a detailed review of the formulations and algorithms for the dial-a-ride problems. In Chapter 3, we introduce a generic GPU framework to accelerate state-of-the-art meta-heuristic approaches for efficiently solving DARP. In Chapter 4, we introduce models and algorithms for the dial-a-ride problem with the vehicle breakdown. In Chapter 5, we introduce models and algorithms for the dial-a-ride problem with electric autonomous vehicles and the modern recharging infrastructures. In Chapter 6, we provide concluding remarks and future research directions. We elaborate on the contributions of each chapter in the following.

FIGURE 1.1: Thesis flow.

In Chapter 2, we provide a detailed review of formulations and solution algorithms for the dial-a-ride problem (DARP), which is becoming increasingly famous in today's ride-sharing transportation. DARP consists in designing vehicle routes and schedules with a minimum value of the objective function to perform door-to-door passenger transportation while respecting several constraints related to the service operator and customer convenience. This problem falls under the category of vehicle routing problems (VRP), which is computationally NP-hard, while the major challenge of DARP consists of balancing the user convenience with the fleet operator's cost. The DARP formulation varies depending on the objective function and constraints. The DARP solution algorithms can be categorized into exact methods and approximation methods. We attempt to provide an in-depth summary and discussion of different formulation schemes, solution frameworks, point out the algorithmic and computational issues, and discuss the real-world applications of the DARP.

In Chapter 3, we introduce a generic GPU-based solution methodology to efficiently solve DARP. The idea of accelerating local search operation in GPU has been introduced before for other optimization problems but never applied for DARP. Besides applying this conventional GPU approach for DARP, five device-oriented optimization strategies are proposed to enhance the utilization of a state-of-the-art GPU architecture (Tesla P100). Specifically, the following device optimization strategies are proposed. i) a device hierarchical memory optimization strategy to effectively map the DARP data structure to various GPU memories, ii) a method to adaptively select the kernel grid configuration for DARP kernels that maximizes GPU utilization, iii) optimizing data transfer: allocation of data in page-locked memory, segregation of the developed DARP kernels, and overlapping data transfer, iv) a scheme to reduce branch divergence during the process of generating neighborhood solutions for DARP, and v) concurrent kernel execution to enable overlapping data transfers between the developed GPU kernels and CPU functions. In the proposed methodology, iterative selection of the best admissible solution is performed in CPU and local search is performed in GPU. For the GPU framework, the choice of requests will be given as input and the evaluated neighborhood solutions will be the output. Therefore, any type of removal and re-insertion operators could be utilized to dislodge the requests involved during the local search. Moreover, any meta-heuristic that utilizes local search operation could be integrated with the proposed approach, which makes the GPU framework to be generic. The proposed GPU framework has been integrated with two meta-heuristics: tabu search and variable neighborhood search. The choice of meta-heuristics can be justified by their simplicity, efficiency, and applicability for a wide range of problems. Results are based on standard dial-a-ride test instances from the literature. Firstly, the proposed GPU-based methods perform significantly faster than their conventional CPU counterparts. Secondly, the effect of inclusion and exclusion of each device optimization strategy on speedup is shown, and the combination of all strategies leads to the highest speedup. Thirdly, the proposed GPU-based methods generate better solutions than the existing best-performing approaches.

Contributions:

- **Ramesh Ramasamy Pandi**, Song Guang Ho, Sarat Chandra Nagavarapu, and Justin Dauwels, “A Generic GPU-Accelerated Framework for the Dial-A-Ride Problem”, In IEEE Transactions on Intelligent Transportation Systems, doi: 10.1109/TITS.2020.2992560.

- **Ramesh Ramasamy Pandi**, Song Guang Ho, Sarat Chandra Nagavarapu, Twinkle Tripathy, and Justin Dauwels, “*GPU-Accelerated Tabu Search Algorithm for the Dial-A-Ride Problem*”, Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems (ITSC), Hawaii, USA, 2018.

In Chapter 4, we introduce models and solution algorithms for disruptive dial-a-ride problem with vehicle breakdown (DDARP-VB) with new additional realistic assumptions such as immediate fleet re-routing after a vehicle breakdown, the decision to serve all the customers to avoid damage to service providers’ reputation, design of a realistic operational cost function, and dynamic updates in travel times between the newly generated nodes, i.e., location of vehicles during vehicle breakdown and all the other nodes in the network. Adaptive Large Neighborhood Search (ALNS) method is one of the outstanding algorithms for vehicle routing problems. The proposed solution methodology is an extension of the basic ALNS approach with GPU acceleration (G-ALNS) to address the real-time dynamic rescheduling for disruption in dial-a-ride context. The proposed method is a novel approach since there are no such G-ALNS in the literature for any extension of VRP-TW, including DARP. A fleet management strategy (FMS) has been proposed, which is based on minimizing the fleet size for normal routing operations and addressing the disruption using a backup fleet in combination with online optimization. This is a novel technique since such a strategy was never reported to address disruption for any VRP variants. Based on realistic estimates of operational cost, the results suggest that the proposed FMS attains reduced operational cost and fleet idle time on both normal and disruptive scenarios. In addition to that, an investigation is carried out to discover the influence of breakdown and effectiveness of the proposed approach on passenger experiences, and the results suggested that the passenger experience in terms of user ride improves both before and after the breakdown, when compared to the conventional method. Furthermore, we observe that the higher user comfort levels (lesser ride times) are achieved using our disruption management method when compared to other methods.

Contributions:

- **Ramesh Ramasamy Pandi**, Song Guang Ho, Sarat Chandra Nagavarapu, Twinkle Tripathy, and Justin Dauwels, “*Disruption Management for Dial-A-Ride Systems*”, In IEEE Intelligent Transportation Systems Magazine, DOI:

10.1109/ MITS.2020.3014429.

- **Ramesh Ramasamy Pandi**, Song Guang Ho, Sarat Chandra Nagavarapu, and Justin Dauwels, “*Adaptive Algorithm for Dial-A-Ride Problem with Vehicle Breakdown*”, Proceedings of the 18th European Control Conference (ECC), Naples, Italy, pp. 2682-2688, 2019.

In Chapter 5, we introduce models and solution algorithms for the electric autonomous dial-a-ride problems with mobile energy disseminators. The advent of “electric autonomous vehicles” has revolutionized intelligent transportation systems, and especially useful in mobility-on-demand services to combat the issues such as global warming, fuel consumption, air pollution, and road safety. In this work, we propose mobile energy dissemination for recharging autonomous electric vehicles (EV) in the dial-a-ride system. The electric autonomous dial-a-ride problem (e-ADARP) consists of designing routes and schedules for user-oriented transit service between specific origins and destinations while satisfying several constraints including time windows, ride-time, battery management, and recharge times. Despite the importance of EVs, the existing distribution of static charging stations (SCS) is either slender or disorganized, which causes range anxiety. The modern charging technology, mobile energy disseminator (MED), is energy-efficient and able to recharge EV on the move without causing any detours. Therefore, we formulate the electric autonomous dial-a-ride problem with MEDs and propose a GPU-based solution methodology. Simulations are conducted on benchmark instances and real-world UBER taxi trip datasets, and the results are compared with existing approaches. Finally, the effectiveness of the proposed MED for the electric autonomous dial-a-ride system in comparison with the conventional SCS. In summary, this chapter is focused on investigating the modern electric autonomous vehicular technology for dial-a-ride systems. The following are the major contributions. 1) Formulation of the electric autonomous dial-a-ride problem with MEDs. 2) Development of a GPU-based solution methodology to efficiently solve e-ADARP. 3) Analyze the effectiveness of MED in the proposed electric autonomous dial-a-ride system when compared to the conventional static recharging stations.

Contributions:

- **Ramesh Ramasamy Pandi** and Justin Dauwels, “*The Electric Autonomous Dial-A-Ride Problem with Mobile Energy Disseminators*”, Under review, IEEE Transactions on Intelligent Transportation Systems.

- **Ramesh Ramasamy Pandi**, Song Guang Ho, Sarat Chandra Nagavarapu, and Justin Dauwels, “*Deterministic Annealing for Depot Optimization: Applications to the Dial-A-Ride Problem*”, Proceedings of the 9th IEEE Symposium Series on Computational Intelligence (SSCI), Bangalore, India, 2018.

In Chapter 6, we provide concluding remarks and discuss interesting future research directions to encourage further advancing of the state-of-the-art developments in DARP.

Chapter 2

The Dial-A-Ride Problem: A Review of Formulations and Algorithms

Summary - Dial-a-ride problem (DARP) has been studied since 1971, which arises in the context of on-demand mobility, and it is becoming increasingly famous in today's ride-sharing transportation.

DARP consists of designing vehicle routes and schedules with a minimum value of the objective function to perform door-to-door passenger transportation while respecting several constraints related to the service operator and customer convenience. This problem falls under the category of vehicle routing problems (VRP), which is computationally NP-hard, while the major challenge of DARP consists of balancing the user convenience with the fleet operator's cost.

The DARP formulation varies depending upon the application context, objective function, and modeling constraints. The DARP solution algorithm can be broadly categorized into exact methods and approximation methods that run on various computational hardware.

This section provides a comprehensive review of various modeling and solution techniques for the DARPs. We attempt to provide an in-depth summary and discussion of different formulation schemes, algorithmic and computational issues.

In this chapter, we review the literature on formulations and algorithms for dial-a-ride problems, which is essential to understand the rest of this thesis. First, we introduce the context of dial-a-ride problem (Section 2.1). Second, we provide a taxonomy and discuss various problem classifications of DARP in the literature (Section 2.2). Third, we provide a taxonomy and discuss the relevant DARP solution methods in the literature (Section 2.3). Finally, we discuss the critical aspects of the state-of-the-art developments in DARP (Section 2.4).

2.1 Introduction

The efficient transportation system is a key factor for the economic development of a country. The various types of classifications in mass transit systems are shown in Figure 2.1. Public trains and buses are cheap and suitable for efficiently traveling long distances but are highly restrictive on service locations and less convenient at times. Personal mobility devices are quite handy for first and last-mile transit, but unfit for a long journey. On the other hand, Demand Responsive Transit (DRT), also known as Dial-A-Ride Transit (DART), refers to a form of shared flexible private or quasi-public transport for traveling in which the vehicle routes and schedules are designed according to the customer demand rather than fixing the timetable or routes. On-demand mobility service provides ride-sharing transportation with high user comfort and flexibility in service locations while incurring a slightly higher fare, and often modeled in terms of dial-a-ride problems (DARP).

Mass Transport Classes

Transport Categories Relationship and Apply Examples

FIGURE 2.1: Classification: Mass Transport Systems [1].

To operator dial-a-ride services, the fleet operator must design vehicle routes and schedules to perform pickup and delivery of passengers while respecting various constraints related to the service provider and customer convenience. The first successful trial of on-demand DAR in the USA and UK was conducted in 1970 and 1972, respectively. Over the years, the planning of DAR services was computerized due to the size and complexity of realistic instances. Nowadays, DARP is often formulated as a combinatorial optimization problem, and computerized solution methods are used to generate optimal or high-quality vehicle routes and schedules for user transportation. One of the pioneering solution approaches for DARP was designed by Wilson (1971) [26]. Later, Psaraftis (1980) [27] has proposed dynamic programming-based exact method for both static and dynamic versions of DARP. A well-studied problem variant, which is often referred to as a "Standard DARP", was formulated by Cordeau and Laporte (2003) [7]. Since 2003, most publications related to the DARP are primarily based on the above-mentioned formulation of [7].

DARPs are often motivated by real-life applications. Each variant of the DARP addresses a different objective function or constraints to derive insights on a particular application. Traditionally, DARP was applied for para-transit service, i.e., door-to-door transportation service for elderly and disabled people, with several operational constraints such as user ride time, user waiting time, pickup, and delivery time windows, vehicle capacity, vehicle type, vehicle layout, customer type. The objective function may consist of single or multiple factors depending on the application of interest. Besides, the DARP was also applied primarily in health-care sectors such as transporting patients between hospitals and homes, isolation of patients and staff, and preferred order of requests based on emergency. This type of application is particularly relevant in pandemic scenarios such as COVID-19 (2019-present) that has been killing millions of people worldwide. During the last decade, a significant application of DARP has been devoted to public transportation. Whenever fixed transit systems (train, bus, etc.) are not available, the DRT becomes a necessary transport choice for people, in which the service provider focuses on maximizing the profit while providing a high-quality mobility service. Several real-life applications of DARP has been reported in the literature for the cities of Bologna, Porto, Milan, Netherlands, New York City, etc.

The DARP solution methods can be broadly classified into exact methods and approximation methods. Exact methods are used to attain optimal solutions while incurring long execution time, and this type of method is particularly suitable for small-scale instances and for the problems in which the designed routes may never be adjusted again. However, there are many factors in the real-world that may necessitate adjusting routes again due to changes in the travel plans, disruptions, vehicle breakdowns, accidents, weather changes, etc. Therefore, approximation methods are often preferred as a solution method for DARP, and it is widely studied in the last decade due to its capability of attaining good solution quality in a short time for instances of realistic scale. Recently, the hybridization of exact methods and heuristics is becoming a new trend for efficiently solving complex variants of DARP.

In this survey, we provide a comprehensive overview of various formulations and solution methodologies related to the dial-a-ride problem.

2.2 Problem classifications

In this section, we describe the mathematical formulation of standard DARP (Section 2.2.1), provide a taxonomy (in Section 2.2.2) and classify DARP models in the literature based on the service design characteristics (Section 2.2.3), objective function (Section 2.2.4), and nature of dynamic/stochastic information available to the service provider (Section 2.2.5).

2.2.1 Definition of DARP

$$\text{Min}f(x) \tag{2.1}$$

subject to,

$$\sum_{j \in N} x_{0j}^k = 1 \quad \forall k \in K, \tag{2.2}$$

$$\sum_{i \in D} x_{i,2n+1}^k = 1 \quad \forall k \in K, \tag{2.3}$$

$$\sum_{j \in N} x_{ji}^k - \sum_{j \in N} x_{ij}^k = 0 \quad \forall i \in P \cup D, k \in K, \tag{2.4}$$

$$\sum_{k \in K} \sum_{j \in N} x_{ij}^k = 1 \quad \forall i \in P \cup D, \tag{2.5}$$

$$\sum_{j \in N} x_{ij}^k - \sum_{j \in N} x_{n+i,j}^k = 0 \quad \forall i \in P, k \in K, \tag{2.6}$$

$$Q_j^k \geq (Q_i^k + q_j)x_{ij}^k \quad \forall i \in N, \forall j \in N, \forall k \in K, \tag{2.7}$$

$$\max(0, q_i) \leq Q_i^k \leq \min(C_k, C_k + q_i) \quad \forall i \in N, \forall k \in K \tag{2.8}$$

$$B_j^k \geq (B_i^k + d_i + t_{ij})x_{ij}^k \quad \forall i \in N, \forall j \in N, \forall k \in K, \tag{2.9}$$

$$e_i \leq B_i^k \leq l_i \quad \forall i \in N, \forall k \in K, \tag{2.10}$$

$$L_i^k = (B_{n+i}^k - (B_i^k + d_i))x_{ij}^k \quad \forall i \in P, \forall k \in K, \tag{2.11}$$

$$t_{i,n+i} \leq L_i^k \leq L \quad \forall i \in P, \forall k \in K, \tag{2.12}$$

$$B_{2n+1}^k - B_0^k \leq T_k \quad \forall k \in K, \tag{2.13}$$

Cordeau (2006) [28] described a Mixed Integer Linear Programming Formulation for DARP. In this problem, a fleet of m vehicles represented by set K have to serve n customer requests. The customer requests lead to n pick up and n drop off nodes. The nodes form a directed graph (N, A) graph where $N = P \cup D \cup \{0, 2n + 1\}$, $P = \{1, \dots, n\}$ and $D = \{n + 1, \dots, 2n\}$. Nodes 0 and $2n + 1$ represent the origin and destination depots. Any arc (i, j) denotes a path from the i^{th} to the j^{th} node. The time to travel the arc (i, j) is t_{ij} and the cost of routing is c_{ij} . The standard formulation of dial-a-ride problem can be described as follows.

Here, $f(x)$ represents the objective function of solution x . The objective function $f(x)$ in Eq. (5.1) may consist of single or multiple factors related to fleet travel time, vehicle waiting time, user ride time, vehicle arrival time, and the number of served customers. The constraints specified in Eqs. (5.2)–(5.3) ensure that all the vehicle routes start and end at the depot. Eqs. (5.4)–(5.5) makes it mandatory for every node to be visited by only one vehicle such that any vehicle reaching a particular node must exit it. Eq. (5.6) ensures that the customers are picked up and dropped off by the same vehicle. Additionally, every vehicle must not exceed its maximum capacity, and the maximum ride time constraints must hold for the vehicles and the customers. Eq. (5.7) updates the load of the k^{th} vehicle once it reaches node j from node i . The constraint in Eq. (5.8) ensures that the capacity of vehicle k is not exceeded at any of the nodes. The time consistency is maintained by Eq. (5.9), which ensures that node j can be serviced only after the arrival of the vehicle. Eq. (5.10) specifies the time window of the service at each location and makes sure that service begins only within the windows. The constraints in Eqs. (5.11)–(5.12) guarantee that the maximum ride time of a customer is not exceeded. Finally, the constraint on the route duration of any vehicle k is ensured by Eq. (5.13).

Meanwhile, Ropke et al. (2007) [8] present a DARP formulation with a two-index scheme, eliminating the index k since a single arc cannot be traversed by more than one vehicle. This formulation was identified to be suitable for the branch-and-cut exact method, while the set-partitioning or set-covering formulations of Parragh and Schmid (2013) [10] was beneficial for column generation approaches. However, the general aim of any type of formulations is to construct a feasible set of routes subject to all constraints and full/partial demand satisfaction.

2.2.2 Taxonomy

In this taxonomy, we classify DARP based on the following factors.

- **Single/multiple depots:** Depot refers to the starting/ending location of a vehicle involved in vehicle routing operations. Single/multiple depots refer to the usage of single or multiple depots associated with vehicles.
- **Single/multiple vehicles:** Single/multiple vehicles refer to the usage of single or multiple vehicles for the routing operations.
- **Homogeneous/heterogeneous fleet:** A homogeneous fleet refers to the collection of vehicles of the same type. A heterogeneous fleet refers to the collection of vehicles of different types (e.g. different seating capacity, battery capacity, etc.).
- **Seating capacity constraint:** It refers to the maximum number of passengers that can be accommodated into a vehicle at any given time.
- **Maximum passenger ride time constraint:** It refers to the maximum allowed time that a passenger can sit inside the vehicle between the pickup and drop off service.
- **Maximum vehicle route duration constraint:** It refers to the time horizon during which a vehicle is employed for service.
- **Selective visits to customers:** Selective visits to the customers denote whether serving all the customers is an essential criterion.
- **Single/multiple objective:** Single/multiple objectives refers to the consideration of single/multiple factors in the objective function.
- **Static/dynamic customer demand:** The customer demand is static if all transportation requests are known in advance. Meanwhile, the customer demand is dynamic if the request arrives in real-time and the planned routes need to be adjusted according to newly-arrived requests.
- **Deterministic/stochastic factors:** If the information about factors of the transportation system is known with certainty, then the system is deterministic. If the information is known with a certain probability, then the system is referred to be stochastic.

- **Internal combustion (IC) fleet/Electric vehicle (EV) fleet:** Internal Combustion (IC) fleet refers to a fleet of vehicles operated by fuel-powered engine. Electric vehicle (EV) fleet refers to a fleet of vehicles operated by battery-powered engines.
- **Arrival/return depot decisions:** The arrival/return depots may need to be considered during the planning phase if a vehicle is not necessarily required to return to a preplanned depot. This is particularly true in cases of autonomous vehicles, where the vehicles can go to the area of higher demand instead of returning back to a preplanned depot.
- **Exact/approximation solution method:** It refers to the type of solution method used to solve the DARP.
- **Sequential/parallel algorithms:** A Sequential/parallel algorithm refers to the involvement of sequential or parallel computation in DARP solution algorithms.
- **CPU/GPU utilization:** CPU/GPU utilization refers to the employment of the type of hardware used in the realization of the DAR system or solution algorithm to solve DARP.

We provide a detailed taxonomy of DARP problem classifications on publications (2007–2020) in the Tables [2.1–2.5](#). We provide additional insights on the number of studies conducted on different areas of the DARP (2007–2020) in Figures [2.2–2.16](#).

FIGURE 2.2: Taxonomy: Percentage of studies that consider IC and EV fleet.

FIGURE 2.3: Taxonomy: Percentage of studies that consider single and multiple depots.

FIGURE 2.4: Taxonomy: Percentage of studies that consider single and multiple trips to depot.

FIGURE 2.5: Taxonomy: Percentage of studies that consider single and multiple vehicles.

FIGURE 2.6: Taxonomy: Percentage of studies that consider homogeneous and heterogeneous fleet.

FIGURE 2.7: Taxonomy: Percentage of studies that consider seating capacity constraint.

FIGURE 2.8: Taxonomy: Percentage of studies that consider user time window constraint.

FIGURE 2.9: Taxonomy: Percentage of studies that consider passenger ride time constraint.

FIGURE 2.10: Taxonomy: Percentage of studies that consider vehicle route duration constraint.

FIGURE 2.11: Taxonomy: Percentage of studies that consider selective visit to customers.

FIGURE 2.12: Taxonomy: Percentage of studies that consider single objective function.

FIGURE 2.13: Taxonomy: Percentage of studies that consider static and dynamic factors.

FIGURE 2.14: Taxonomy: Percentage of studies that consider deterministic and stochastic factors.

FIGURE 2.15: Taxonomy: Percentage of studies that consider optimizing depot visits.

FIGURE 2.16: Taxonomy: Percentage of DARP studies in various time-period considered in our review.

2.2.3 Advanced Service Design

In this section, we classify DARP based on the characteristics related to the design of the service. Specifically, we discuss the DARP variants that consider heterogeneous users, vehicles or drivers (Section 2.2.3.1), more complex routing properties (Section 2.2.3.2), different service level specifications (Section 2.2.3.3), and the usage of electric and autonomous vehicles (Section 2.2.3.4).

2.2.3.1 Heterogeneity

In healthcare transportation service, the patients might require different seating arrangements such as **wheelchair or stretcher** in the vehicle (e.g., Parragh 2011 [53]; Parragh 2012 [62]; Liu et al. 2015 [91], Zhang et al. 2015 [96]), meanwhile the accompanying staff require a proper seating arrangement. Parragh 2011 [53] extended the formulation of Cordeau (2006) [28] in which an additional index is introduced to respect vehicle capacity for each resource type. Artificial origin and destination depots are introduced by Parragh 2011 [53], which is extended from the formulation of Ropke et al. 2007 [8].

Qu and Bard (2013) [70] extended the formulation of Parragh 2011 [53] and introduced **configurable vehicles** for DARP. The vehicle capacity or the internal vehicle layout can be adjusted according to the demand requirement. Reconfigurable vehicles are also studied by Liu et al. 2015 [91], Hanne et al. 2009 [41], Wong and Bell 2006 [133], and Beaudry et al. 2010 [43], however the mathematical formulation was formally introduced by Qu and Bard (2013) [70].

Besides, the passengers may be classified based on **medical conditions**, i.e., high-risk patients might need to be transferred to isolation areas or hospitals, which is particularly relevant in the time of pandemic such as COVID-19. Hanne et al. 2009 [41], and Beaudry et al. 2010 [43] have introduced priority related constraints to transfer patients between hospitals. After the transfer of passengers, the vehicle needs to be cleaned at any of the cleaning depots. Xiang et al. 2006 [134] consider heterogeneity in emergency medical vehicles, i.e., a driver's qualifications should match the vehicle's medical supply levels. Detti et al. 2017 [101] and Molenbruch et al. 2017 [106] imposed compatibility restrictions on types of patients and medical

Sequential/Parallel	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
GPU-usage	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Depot decisions	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
IC/EV Fleet	IC	IC	IC	IC	IC	IC	IC	IC	IC	IC	IC	IC	IC	IC	IC	IC	IC	IC	IC
Deterministic/Stochastic	D	D	D	D	D	D	D	S	S	D	D	D	D	D	D	D	S	D	D
Static/Dynamic	S	S	S	S	S	S	D	D	D	S	S	S	S	S	S	D	S	D	S
S/M Objective	M	S	M	S	S	M	S	S	S	M	S	S	S	M	M	M	S	M	S
SV	N	N	N	N	Y	Y	N	N	Y	N	N	N	N	N	N	N	Y	N	N
VRD	Y	N	N	Y	N	Y	Y	N	Y	N	N	Y	Y	Y	Y	Y	N	Y	N
PRT	Y	Y	N	Y	N	Y	Y	N	Y	N	Y	Y	Y	Y	Y	Y	N	Y	N
TW	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y
Seating capacity	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Ho/He Fleet	HO	HO	HE	HO	HO	HE	HO	HE	HO	HO	HE	HE	HE	HO	HE	HO	HE	HE	HO
S/M Vehicles	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	S	M	M
S/M Trips	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
S/M Depots	S	S	M	S	S	M	M	M	S	S	S	M	M	S	M	S	M	M	M
Year	2007	2007	2007	2007	2007	2008	2008	2008	2009	2009	2009	2009	2009	2009	2009	2010	2010	2010	2010
Publication	Jorgensen et al. (2007) [29]	Luo and Schonfeld (2007) [30]	Melachrinoudis et al. (2007) [31]	Ropke et al.(2007) [8]	Wolfer Calvo and Colomi (2007) [32]	Quadrifoglio et al. (2008) [33]	Waisanen et al. (2008) [34]	Xiang et al. (2008) [35]	Cubillos et al. (2009) [36]	Hall et al.(2009) [37]	Karabuk (2009) [38]	Mauri et al.(2009) [39]	Parragh et al. (2009) [40]	Hanne et al. (2009) [41]	Hyytia et al. (2010) [42]	Beaudry et al. (2010) [43]	Cortes et al. (2010) [44]		

TABLE 2.1: Taxonomy: Problem classifications for DARP (part 1)

Garaix et al. (2010) [45]	2010	M	S	S	M	HE	Y	Y	Y	N	N	N	N	N	N	IC	D	S	M	S	S	N	S
Gupta et al. (2010) [46]	2010	S	S	S	S	HO	Y	N	N	N	N	N	N	N	N	IC	D	S	S	S	S	N	S
Haugland and Ho (2010) [47]	2010	S	S	S	S	HO	Y	Y	Y	N	N	N	N	N	N	IC	D	S	S	S	S	N	S
Parragh et al. (2010) [9]	2010	S	S	S	S	HO	Y	Y	Y	Y	N	N	N	N	N	IC	D	S	S	S	S	N	S
Tang et al. (2010) [48]	2010	S	S	S	S	HO	Y	Y	Y	Y	N	N	N	N	N	IC	D	S	S	S	S	N	S
Berbeglia et al. (2011) [49]	2011	S	S	S	S	HO	Y	Y	Y	Y	N	N	N	N	N	IC	D	S	S	S	S	N	S
Firat and Woeginger (2011) [50]	2011	S	S	S	S	HO	Y	Y	Y	Y	N	N	N	N	N	IC	D	S	S	S	S	N	S
Garaix et al. (2011) [51]	2011	M	S	S	M	HE	Y	Y	Y	Y	N	N	N	N	N	IC	D	S	S	S	S	N	S
Hame (2011) [52]	2011	S	S	S	S	HO	Y	Y	Y	Y	N	N	N	N	N	IC	D	S	M	S	S	N	S
Parragh (2011) [53]	2011	S	S	S	M	HE	Y	Y	Y	Y	N	N	N	N	N	IC	D	S	S	S	S	N	S
Schilde et al. (2011) [54]	2011	S	S	S	M	HO	Y	Y	Y	Y	N	N	N	N	N	IC	S	D	M	D	S	N	S
Heilporn et al. (2011) [55]	2011	S	S	S	S	HO	Y	Y	Y	Y	N	N	N	N	N	IC	S	S	M	S	S	N	S
Ho and Haugland (2011) [56]	2011	S	S	S	M	HE	Y	Y	Y	Y	N	N	N	N	N	IC	S	S	S	S	S	N	S
Hyttia et al. (2012) [57]	2012	S	S	S	S	HO	N	N	N	N	N	N	N	N	N	IC	S	S	M	D	S	N	S
Berbeglia et al. (2012a) [58]	2012	S	S	S	M	HO	Y	Y	Y	Y	Y	Y	Y	Y	Y	IC	D	D	S	D	S	N	S
Berbeglia et al. (2012b) [59]	2012	M	S	M	M	HO	Y	N	Y	N	N	N	N	N	N	IC	D	S	S	S	S	N	S
Chevrier et al. (2012) [60]	2012	S	S	S	M	HE	Y	Y	Y	Y	Y	N	N	N	N	IC	D	S	M	S	S	N	S
D'Souza et al. (2012) [61]	2012	S	S	S	S	HO	N	N	N	Y	Y	N	N	N	N	IC	D	S	M	S	S	N	S
Parragh et al. (2012) [62]	2012	S	S	S	M	HE	Y	Y	Y	Y	Y	N	N	N	N	IC	D	S	S	S	S	N	S
Zidi et al. (2012) [63]	2012	S	S	S	M	HE	Y	Y	Y	N	N	N	N	N	N	IC	D	S	M	S	S	N	S
Carnes et al. (2013) [64]	2013	M	S	S	M	HE	Y	Y	Y	Y	N	N	N	N	N	IC	D	S	S	S	S	N	S
Guerriero et al. (2013) [65]	2013	S	S	S	M	HE	Y	Y	Y	Y	Y	N	N	N	N	IC	D	S	S	S	S	N	S
Hame and Hakula (2013) [66]	2013	S	S	S	M	HO	Y	Y	Y	Y	Y	N	N	N	N	IC	D	S	S	S	S	N	S
Kirchler and Wölfler Calvo (2013) [67]	2013	S	S	S	M	HO	Y	Y	Y	Y	Y	Y	Y	Y	Y	IC	D	S	M	S	S	N	S
Muelas et al. (2013) [68]	2013	S	S	S	M	HO	Y	Y	Y	Y	N	N	N	N	N	IC	D	S	S	S	S	N	S
Paquette et al. (2013) [69]	2013	S	S	S	M	HE	Y	Y	Y	Y	Y	N	N	N	N	IC	D	S	M	S	S	N	S
Parragh and Schmid (2013) [10]	2013	S	S	S	M	HO	Y	Y	Y	Y	Y	Y	Y	Y	Y	IC	D	S	M	S	S	N	S

TABLE 2.2: Taxonomy: Problem classifications for DARP (part 2)

Qu and Bard (2013) [70]	2013	S	S	M	HE	Y	Y	N	N	N	S	S	D	IC	N	N	S
Reinhardt et al. (2013) [71]	2013	M	M	M	HE	Y	Y	N	Y	Y	M	S	D	IC	N	N	S
Shen and Quadrifoglio (2013) [72]	2013	M	S	M	HE	Y	Y	Y	N	N	S	S	D	IC	N	N	S
Hall and Peterson (2013) [73]	2013	M	S	M	HE	Y	Y	Y	N	N	S	S	D	IC	N	N	S
Atahran et al. (2014) [74]	2014	S	S	M	HE	Y	Y	Y	N	N	M	S	D	IC	N	N	S
Brackers et al. (2014) [11]	2014	M	S	M	HE	Y	Y	Y	Y	N	S	S	D	IC	N	N	S
Feng et al.(2014) [75]	2014	S	M	M	HE	Y	Y	Y	Y	N	S	S	D	IC	N	N	S
Ilani et al. (2014) [76]	2014	M	M	M	HE	Y	Y	Y	N	N	S	S	D	IC	N	N	S
Masson et al. (2014) [77]	2014	M	S	M	HO	Y	Y	Y	Y	N	S	S	D	IC	N	N	S
Lehuede et al. (2014) [78]	2014	M	S	M	HO	Y	Y	Y	Y	N	M	S	D	IC	N	N	S
Wong et al. (2014) [79]	2014	S	S	M	HO	N	Y	Y	N	Y	M	D	D	IC	N	N	S
Maalouf et al. (2014) [80]	2014	S	S	M	HO	Y	Y	Y	N	Y	M	D	S	IC	N	N	S
Nunez et al. (2014) [81]	2014	M	S	M	HO	Y	N	N	N	N	M	D	S	IC	N	N	S
Schilde et al. (2014) [82]	2014	S	S	M	HO	Y	Y	Y	Y	N	M	D	S	IC	N	N	S
Hall et al. (2015) [83]	2015	M	S	M	HE	Y	Y	Y	N	N	M	D	D	IC	N	N	S
Hame and Hakula (2015) [84]	2015	S	S	M	HO	Y	Y	Y	Y	N	S	D	D	IC	N	N	S
Markovic et al. (2015) [85]	2015	S	S	M	HE	Y	Y	Y	Y	N	M	D	D	IC	N	N	S
Santos and Xavier (2015) [86]	2015	M	S	M	HE	Y	Y	Y	Y	Y	M	D	D	IC	N	N	S
Munoz-Carpintero et al. (2015) [87]	2015	M	S	M	HO	Y	N	N	N	N	S	D	S	IC	N	N	S
Sayarshad and Chow (2015) [88]	2015	S	S	M	HO	Y	N	N	N	N	S	D	S	IC	N	N	S
Gschwind and Irnich (2015) [89]	2015	S	S	M	HO	Y	Y	Y	Y	N	S	S	D	IC	N	N	S
Hu and Chang (2015) [90]	2015	S	S	M	HO	Y	Y	Y	Y	N	S	S	D	IC	N	N	S
Liu et al. (2015) [91]	2015	S	M	M	HE	Y	Y	Y	Y	N	S	S	D	IC	N	N	S
Muelas et al. (2015) [92]	2015	S	S	M	HO	Y	Y	Y	N	N	S	S	D	IC	N	N	S
Parragh et al.(2015) [93]	2015	S	M	M	HO	Y	Y	Y	Y	Y	S	S	D	IC	N	N	S
Qu and Bard (2015) [94]	2015	S	S	M	HE	Y	Y	Y	Y	N	M	S	D	IC	N	N	S
Urrea et al.(2015) [95]	2015	S	S	M	HO	Y	Y	Y	Y	Y	N	M	S	D	IC	N	S
Zhang et al. (2015) [96]	2015	S	M	M	HO	Y	Y	Y	Y	Y	M	S	D	IC	N	N	S

TABLE 2.3: Taxonomy: Problem classifications for DARP (part 3)

Braekers and Kovacs (2016) [97]	2016	S	S	M	HE	Y	Y	Y	Y	N	S	S	D	IC	N	N	S
Chassaing et al. (2016) [12]	2016	S	S	M	HO	Y	Y	Y	Y	N	S	S	D	IC	N	N	S
Gschwind and Drexler (2016) [98]	2016	S	S	M	HO	Y	Y	Y	Y	N	S	S	D	IC	N	N	S
Masmoudi et al. (2016) [99]	2016	M	M	M	HE	Y	Y	Y	Y	N	S	S	D	IC	N	N	S
Ritzinger et al. (2016) [100]	2016	S	S	M	HO	Y	Y	Y	Y	N	S	S	D	IC	N	N	S
Detti et al. (2017) [101]	2017	M	M	M	HE	Y	Y	Y	Y	N	M	S	D	IC	N	N	S
Lim et al. (2017) [102]	2017	S	M	M	HE	Y	Y	Y	Y	Y	M	S	D	IC	N	N	S
Masmoudi et al. (2017) [103]	2017	S	S	M	HE	Y	Y	Y	Y	N	S	S	D	IC	N	N	S
Molenbruch et al. (2017a) [104]	2017	M	M	M	HE	Y	Y	Y	Y	N	S	S	D	IC	N	N	S
Molenbruch et al. (2017b) [105]	2017	S	S	M	HO	Y	Y	Y	Y	N	S	S	D	IC	N	N	S
Molenbruch et al. (2017c) [106]	2017	S	S	M	HO	Y	Y	Y	Y	N	M	S	D	IC	N	N	S
Pimenta et al. (2017) [107]	2017	S	M	M	HO	Y	N	Y	Y	N	S	S	D	IC	N	N	S
Posada et al. (2017) [108]	2017	S	S	M	HE	Y	Y	Y	Y	N	S	S	D	IC	N	N	S
Schonberger (2017) [109]	2017	M	M	M	HE	Y	N	Y	Y	N	S	S	D	IC	N	N	S
Tripathy et al. (2017) [110]	2017	S	S	M	HO	Y	Y	Y	Y	N	M	S	D	IC	N	N	S
Masmoudi et al. (2018) [111]	2018	S	S	M	HE	Y	Y	Y	Y	N	S	S	D	EV	N	N	S
Oscar Tellez et al. (2018) [112]	2018	M	M	M	HE	Y	Y	Y	Y	N	M	S	D	IC	N	N	S
Reidler et al. (2018) [113]	2018	S	S	M	HO	Y	Y	Y	Y	Y	S	S	D	IC	N	N	S
Drakoulis et al. (2018) [114]	2018	M	M	M	HO	Y	Y	Y	Y	N	S	S	D	IC	N	N	S
Van Engelen et al. (2018) [19]	2018	S	M	M	HO	Y	Y	Y	Y	N	M	D	S	IC	N	N	S
Ramesh et al. (2018a) [6] (This thesis)	2018	S	S	M	HO	Y	Y	Y	Y	N	M	S	D	IC	N	YES	Parallel
Ramesh et al. (2018b) [115] (This thesis)	2018	M	M	M	HO	Y	Y	Y	Y	N	S	S	D	IC	YES	YES	Parallel
SG Ho et al. (2018a) [116]	2018	S	S	M	HO	Y	Y	Y	Y	N	M	S	D	IC	N	N	S
SG Ho et al. (2018b) [117]	2018	S	S	M	HO	Y	Y	Y	Y	N	M	S	D	IC	N	N	S

TABLE 2.4: Taxonomy: Problem classifications for DARP (part 4)

SG Ho et al. (2018c) [118]	2018	S	S	M	HO	Y	Y	Y	Y	N	M	S	S	IC	N	N	S
Gschwind et al. (2019) [119]	2019	S	S	M	HO	Y	Y	Y	Y	N	S	S	D	IC	N	N	S
Bongiovanni et al. (2019) [24]	2019	M	S	M	HO	Y	Y	Y	Y	N	M	S	D	EV	YES	N	S
M Abedi et al. (2019) [120]	2019	S	S	M	HE	Y	Y	Y	Y	N	M	S	D	IC	N	N	S
Z Luo et al. (2019) [121]	2019	S	M	M	HE	Y	Y	Y	Y	Y	M	S	D	IC	N	N	S
Riley et al. (2019) [122]	2019	M	S	M	HO	Y	Y	Y	Y	N	S	D	D	IC	N	N	S
Arslan Ali Syed et al. (2019) [123]	2019	M	S	M	HO	Y	Y	Y	Y	Y	S	D	D	IC	N	N	S
Ramesh et al. (2019) [124] (This thesis)	2019	S	S	M	HO	Y	Y	Y	Y	N	S	D	D	IC	N	YES	Parallel
J Christiaens et al. (2020) [125]	2020	M	S	M	HE	Y	Y	Y	Y	N	S	S	D	IC	N	N	S
Masmoudi et al. (2020) [126]	2020	S	S	M	HE	Y	Y	Y	Y	N	S	S	D	IC	N	N	S
Paquay et al. (2020) [127]	2020	M	S	M	HE	Y	Y	Y	Y	Y	M	D	D	IC	N	N	S
Oscar Tellez et al. (2020) [128]	2020	S	S	M	HO	Y	Y	Y	Y	N	M	S	D	IC	N	N	S
X Liang et al. (2020) [129]	2020	S	M	M	HO	Y	Y	Y	N	Y	M	D	D	IC	N	N	S
Ramesh et al. (2020a) [130] (This thesis)	2020	S	S	M	HO	Y	Y	Y	Y	N	M	S	D	IC	N	YES	Parallel
Ramesh et al. (2020b) [131] (This thesis)	2020	S	S	M	HO	Y	Y	Y	Y	N	S	D	D	IC	N	YES	Parallel
Ramesh and Dauwels (2020c) [132] (This thesis)	2020	S	S	M	HO	Y	Y	Y	Y	N	M	S	D	EV	YES	YES	Parallel

TABLE 2.5: Taxonomy: Problem classifications for DARP (part 5)

vehicles for transportation services. Oscar Tellez et al. 2018 [112] devised DARP with reconfigurable vehicle capacity for disabled passenger transportation service.

2.2.3.2 Routing properties

The **transfers of users between vehicles** are allowed in DARP by [77], in which Masson et al. 2014 have added the constraints related to user ride time and route duration to extend the formulation of standard DARP of Cordeau and Laporte (2003) [7] and PDPTW of Cortes et al. 2010 [44]. This is particularly interesting due to the emerging behavior of autonomous vehicles. Autonomous vehicles (AV) are emerging vehicular technologies, which are robotic controlled vehicles to transport passengers. Machine learning algorithms are used in AVs to capture the feature of a specific operating area so that the AV can be safely navigated. In such cases, the DARP with passenger transfers can be very attractive in the initial stages of implementing the AV in daily transit systems. Similarly, the **integration** of different systems balance the negative aspects of each other. The transfer of users between on-demand transit and public transit system was considered in Hickman and Blume 2001 [135], Hall et al 2009 [37], and Aldaihani and Dessouky 2003 [136].

Splitting **multiple requests on similar location** was considered in Parragh et al. 2015 [93]. Meanwhile, Qu and Bard 2015 [94] and Liu et al. 2015 [91] have grouped individual requests with similar customer locations and pickup time windows to reduce the problem complexity.

In some DARP applications, the staff roster needs to be transported in **multiple trips** to the working sites (e.g., Parragh et al. 2012 [62]), while considering scheduled breaks such as lunchtime (e.g., Zhang et al. 2015 [96], Liu et al. 2015 [91], Masmoudi et al. 2016 [99], Beaudry et al. 2010 [43], Hanne et al. 2009 [41], Karabuk 2009 [38]). Recently, Braekers and Kovacs 2016 [97] consider a multi-trip multiple day's DAR application while ensuring driver consistency. Riedler et al. 2018 [113] present mixed-integer linear programming for DARP in which there is no guarantee that all customers are served and devise a Bender's decomposition as a solution method. Christiaens et al. 2019 [125] study a prisoner transportation problem in dial-a-ride context. Masmoudi et al. 2020 [126] present a hybrid ALNS for the green heterogeneous DARP while allowing the vehicles to be refueled at the

fueling stations. Oscar et al. 2020 [128] consider service time-consistency for the DARP.

2.2.3.3 Service quality

The service quality of DARP is expressed in terms of **time window** and *maximum user ride time limit*. Initially, Jaw et al. 1986 [137] expressed only preferred departure or arrival time. Later, Cordeau and Laporte (2003) formalized time windows for DARP since the customer may arrive earlier than the requested time. For most of the DARP variants in the literature, the time window is only defined for either the inbound (tighter pickup time window) or outbound (tighter drop off time window) request along with a maximum user ride time (e.g., Diana and Dessouky 2004 [138]). Paquette et al. 2009 [139] review various methods to express customer ride time limit. Maximum user ride time can be implicitly set by defining both pickup and drop off time windows for each user (e.g., Wolfer Calvo et al. 2011 [140], Parragh et al. 2015 [93], Atahran et al. 2014 [74]). Meanwhile, the maximum user ride time can be considered as **soft constraints** by adding penalty cost in the objective function (e.g., Jorgensen et al. 2007 [29], Melachrinoudis et al. 2007 [31], Kirchler et al. 2013 [67]). Similarly, the load violations are minimized in the objective function instead of formulating as hard constraints (e.g. Mauri and Lorena 2006 [141]). Waiting times are avoided except for pick up and drop off service in Diana and Dessouky 2004 [138], Luo and Schonfield 2007 [30], and Schilde et al. 2014 [82]. Differently from other approaches, Drakoulis et al. 2018 [114] investigated a real-life on-demand public transportation service based on Greek city. The authors modeled the system as DARP with time windows and proposed an insertion heuristic algorithm with a gamification layer as a solution approach.

2.2.3.4 Green transport

Autonomous **electric vehicles** are being increasingly relevant in the ride-sharing transportation systems. Pimenta et al. 2017 [107] studied models and algorithms for reliability oriented DARP with autonomous electric vehicles. The battery-operated vehicles are considered by Masmoudi et al. 2018 [111], in which battery swapping stations are employed for vehicle recharging mechanisms. The electric autonomous vehicle fleet in DARP was considered in Bongiovanni et al. 2019 [24].

These authors consider static charging stations for the vehicle recharging process during the planning time horizon. Abedi et al. 2019 [120] present a tabu search algorithm for the green heterogeneous dial-a-ride problem. Ramesh and Dauwels 2020c [132] (Chapter 5 of this thesis) present the models and algorithms for DARP with electric autonomous vehicles and modern recharging infrastructures.

2.2.4 Objective functions

In this section, we classify the DARP based on the number of objectives considered in the optimization process. The general problem definition includes a single operational objective (Section 2.2.4.1), while several studies have considered both operational and service-related objectives (Section 2.2.4.2) whose practical relevance has been described in the literature.

2.2.4.1 Single objective

A single objective related to **operational cost** has been considered in many DARP applications. Cordeau and Laporte 2003 [7] consider reducing the total fleet travel distance and Ritzinger et al. 2016 [100] consider reducing the total fleet travel time. Minimizing fleet size was considered by Diana et al. 2006 [142]. Maximizing vehicle occupancy was considered by Gariax et al. 2011 [51]. Profit maximization was considered by Parragh et al. 2015 [93]. Since all customers may not be served in dynamic scenarios, maximizing the number of served requests was considered by Attanasio et al. 2004 [143], Yi and Tian 2005 [144], and Berbeglia et al. 2012 [58]. Minimizing the time needed to serve all requests were studied by Feuerstein and Stougie 2001 [145]. Maximum flow time was minimized by Krumke et al. 2005 [146].

2.2.4.2 Multiple objectives

To **optimize service quality**, multiple objectives need to be included in the objective function (Lehuede et al. 2014 [78]). Minimizing the total user wait time and user ride time are the common objectives. Meanwhile, additional factor such as vehicle idle time, driver wage, fleet route duration, additional taxis to cover fleet

shortage was considered by Diana and Dessouky 2004 [138], Xiang et al. 2006 [134], Jorgensen et al. 2007 [29], Wong and Bell 2006 [133], respectively. Minimizing fleet size (Xiang et al. 2006 [134], Guerriero et al. 2013 [65]) and **penalty terms** for soft constraints (Mauri and Lorena 2006 [141], Jorgensen et al. 2007 [29], Melachrinoudis et al. 2007 [31]) were also considered in the literature. Most of the existing literature considers **weighted-sum objective** function for the DARP [29] [67] [9] [78]. Schilde et al. 2011, 2014 [54] [82] consider hierarchical objectives, while Pareto dominance was studied in Parragh et al. 2009 [40], Paquette et al. 2013 [69], and Molenbruch et al. 2017 [106].

2.2.5 Stochastic/Dynamic factors

In this section, we classify DARP based on the nature of the information available to the service provider. In static and deterministic DARP, all the factors are deterministic and known before vehicle routes and schedules are designed. In stochastic DARP, one or more factors related to the system are considered to follow a random probability distribution. In dynamic DARP, one or more factors related to the system are considered to change even after designing the vehicle routes and schedules. This section discusses the stochastic and dynamic DARP variants that involves stochastic or dynamic information on transportation times (Section 2.2.5.1), requests (Section 2.2.5.2), and customer behavior (Section 2.2.5.3).

2.2.5.1 Transportation times

Time-dependent travel times can be considered to account for congestion, thereby increasing the system reliability (Fu 2002 [147]). Xiang et al. 2008 [35] consider stochastic service and travel time in dynamic vehicle routing problems. Stochastic factors of vehicle accidents are considered in Schilde et al. 2014 [82]. Liang et al. 2020 [129] present an integer non-linear programming (INLP) for the automated taxi's dial-a-ride problem in which dynamic requests are considered using the rolling horizon approach. These authors present a customized Lagrangian relaxation algorithm as a solution approach and evaluate the DARP with dynamic travel times on real-life instances based on Delft, Netherlands.

2.2.5.2 Customer requests

In dynamic DARP, the new dynamic requests are considered during or after the route planning phase (Attanasio et al. 2004 [143]). Hyytia et al. 2010 [42] present the risk of congestive collapse, and Berbeglia et al. 2012 [58] present the formulation to study the problem of infeasibility. Hanne et al. 2009 [41], Beaudry et al. 2010 [43] study emergency transportation problem. Coslovich et al. 2006 [148] study dynamic DARP with time windows. Unexpected events such as request cancellation, breakdowns, and traffic gridlocks are also considered (Donoso et al. 2009 [149], Hame 2011 [52], Xiang et al. 2008 [35]). For a detailed review of dynamic DARP, the reader is referred to Berbeglia et al. 2010 [150].

The dynamic DARP is a **time-critical** problem, which can be reflected by **competitive ratio** and the lower bound for this ratio is computed by Feuerstein and Stougie 2001 [145] and Ascheuer et al. 2000 [151]. Lipmann et al. 2004 [152] model the system in such a way that the information about user destination is only known after visiting the customer's origin location. Krumke et al. 2005 [146] identify that a solution algorithm cannot be competitive for the unit-capacity single-vehicle problem. Yi and Tian 2005 [144] provide lower bounds in the competitive ratio for the unit or infinite capacity single-vehicle problem. Probability distribution may be known about dynamic requests, therefore DARP can be also modeled as **stochastic** or probabilistic (Schilde et al. 2011 [54], Ho and Haugland 2011 [56]).

Van Engelen et al. 2018 [19] present a dynamic insertion algorithm for the DARP with demand forecasts. The performance of the algorithms is tested on a real-life case study based in the Netherlands. It has been discovered that incorporating the information about demand forecast in route designing algorithms can be beneficial in reducing the overall transportation costs. Riley et al. 2019 [122] present a column generation approach for dynamic DARP for the instances based on taxi trip information of the New York City (NYC) and guarantee the transportation of all customer requests. For the NYC dataset, Arslan et al. 2019 [123] present an adaptive large neighborhood search algorithm for the dynamic DARP. Pacquay et al. 2020 [127] present recovery management techniques for the DARP with real-time disruptions in which a customer request might be canceled, modified, or delayed due to unforeseen circumstances. Ramesh et al. 2020b [131] (Chapter 4 of

this thesis) present the models and algorithms for the dynamic dial-a-ride problem with a vehicle breakdown.

2.2.5.3 User characteristics

Stochastic user behavior was considered by Heilporn et al. 2011 [55]. Deflorio et al. 2002 [153] study the stochastic factors due to the lateness of users and drivers. The driver's decision to wait for the customer requests might affect the problem feasibility.

2.3 Solution Methods

In this section, we provide a taxonomy (Section 2.3.1) and classify the solution methods for DARP into two major subsections: exact methods (Section 2.3.2) and approximation methods (Section 2.3.3). Exact methods guarantee optimality while consuming large execution time and they are suitable for small-scale instances. Approximation methods provide good solutions (not necessarily optimal) in a short time and they are suitable for large-scale instances. Finally, we discuss the DARP solution approaches based on parallel computing techniques (Section 2.3.4).

2.3.1 Taxonomy

In this section, we provide a taxonomy and insights on the percentage of various solution methods considered for solving DARP between 2007 and 2020 in Figures 2.18–2.19.

FIGURE 2.17: Taxonomy: Percentage of studies that utilize exact solution methods.

FIGURE 2.18: Taxonomy: Percentage of studies that utilize heuristic-based solution methods.

FIGURE 2.19: Taxonomy: Percentage of studies that utilize GPU-based solution methods.

2.3.2 Exact Methods

Exact methods guarantee optimal solutions while consuming large execution times. Therefore, these type of methods are suitable for small-scale instances. In this section, we provide a taxonomy (Table 2.6), and classify the exact methods for DARP in the literature into several subcategories: branch-and-cut (Section 2.3.2.1), column generation (Section 2.3.2.2), branch-(cut)-and-price (Section 2.3.2.3), and dynamic programming (Section 2.3.2.4).

2.3.2.1 Branch-and-cut

The Branch-and-cut algorithm (B&C) is based on the procedure of branch-and-bound (B&B) with cutting planes in tree problem to tighten the linear programming relaxations. This provides stronger bounds to verify optimal solutions. Cordeau (2006) [28] presents a branch-and-cut method for mixed-integer linear programming formulation of DARP. Ropke et al. 2007 [8] added valid inequalities for reachability and fork structures. Parragh (2011) [53] and Braekers et al. (2014) [11] adapt the formulation of Cordeau (2006) [28] for heterogeneous DARP variant. Liu et al.

Exact methods
Braekers et al. (2014), Cordeau (2006), Hall et al. (2009), Heilporn et al. (2011), Liu et al. (2015), Parragh et al. (2009), Parragh (2011), Ropke et al. (2007)
Garaix et al. (2011), Parragh et al. (2012)
Gschwind and Irnich (2015), Parragh et al. (2015), Qu and Bard (2015)
Desrosiers et al. (1986), Hame (2011), Hame and Hakula (2013), Hame and Hakula (2015), Lois et al. (2007), Psaraftis (1980), Psaraftis (1983), Ziliaskopoulos and Kozanidis (2006)
Pimenta et al. (2017), Posada et al. (2017), Oscar Tellez et al (2018), M Reidler et al (2018), Bongiovanni et al (2019), Z Luo et al (2019), Riley et al (2019), J Christiaens (2020), Oscar Tellez et al (2020)

TABLE 2.6: Taxonomy: Exact methods for solving DARP

2015 [91] model configurable vehicles for DARP variant. Heilporn et al. 2011 [55] apply branch-and-cut method in the stochastic context of the DARP.

2.3.2.2 Column generation

The DARP can be split into a master and a subproblem. The master problem can be tackled by set-partitioning/covering formulation while the subproblem can be approached with a heuristic or metaheuristic (Parragh et al. 2012 [62], Garaix et al. 2011 [51], Coppi et al. 2013 [154], Parragh and Schmid 2013 [10]).

2.3.2.3 Branch-(cut)-and-price

Gschwind and Irnich (2015) [89] reported efficient exact solutions for standard DARP instances [7]. This is due to the combination of column generation, branch-and-cut algorithm with pre-processing methods and cutting planes (Cordeau 2006 [28], Ropke et al. 2007 [8]). Qu and Bard 2015 [94] reported branch-and-cut-and-price method for DARP with configurable vehicles. Parragh et al. 2015 [93] solved DARP with split requests and profits using **branch-and-price** method. Z Luo et al. 2019 [121] present a branch-cut-and-price approach for the DARP with patient transportation.

2.3.2.4 Dynamic programming

Standalone dynamic programming approaches were explored for single-vehicle routing problem (e.g., Psaraftis 1980 [27], Psaraftis 1983 [155], Desrosiers et al. 1986 [156]) and multi-vehicle routing problems (e.g., Ziliaskopoulos and Kozanidis 2006 [157], Lois et al. 2007 [158], Hame and Hakula 2013 [66]). Hame and Hakula 2015 [84] present DP method to solve constraint programming formulation (Berbeglia et al. 2011 [49]).

2.3.3 Approximation Methods

Approximation methods do not guarantee optimal solutions but provide near-optimal or very good solutions in a relatively short time. Therefore, these types of

methods are suitable for instances of a realistic scale. Due to this reason, during the last decade, the focus of researchers was swiftly on developing approximation methods for the DARPs. In this section, we provide a taxonomy (Table 2.7), and classify the approximation methods for DARP in the literature into several sub-categories: insertion heuristics (Section 2.3.3.1), local-search based metaheuristics (Section 2.3.3.2), population-based metaheuristics (Section 2.3.3.3), and hybrid solution approaches (Section 2.3.3.4).

2.3.3.1 Insertion Heuristics

Jaw et al. 1986 [137] introduced a sequential **insertion heuristic** in which the customer requests are sorted according to their earliest arrival time, and iteratively inserted into the best feasible positions of the vehicle routes. Diana and Dessouky 2004 [138] introduced a parallel insertion strategy with a regret criterion to improve the final solution quality. The insertion order of the customer requests can be altered based on the problem variant (e.g., Wong and Bell 2006 [133], Reinhardt et al. 2013 [71]). This technique was also applied in a dynamic problem context (e.g., Coslovich et al. 2006 [148]). Luo and Schonfeld 2011 [159] discovered that postponing the insertion of requests to be served in the future might reduce the fleet size requirement.

Ioachim et al. (1995) [160] present a column generation approach, in which the master problem is solved as a set-partitioning problem, and the subproblem is solved as constrained PDP shortest path problem. This type of approach is referred to as *cluster-first route second* heuristic. Other studies of this heuristic for DARP can be found in Borndorfer et al. 1999 [161] and Karabuk 2009 [38].

2.3.3.2 Local-search based Metaheuristics

The **tabu search** (TS) heuristic was first proposed by Glover 1989 [162], and first applied to solve DARP by Cordeau and Laporte (2003) [7]. In each iteration, a single solution attribute is changed. Such previously visited solution attribute is termed as a tabu move, which is avoided for a predefined number of iterations, thereby avoiding cycling, or getting stuck in locally optimal solutions. However,

Approximate methods

- Borndorfer et al. (1999), Coslovich et al. (2006), Diana and Dessouky (2004), Ioachim et al. (1995),
 Jaw et al. (1986), Karabuk (2009), Luo and Schonfeld (2011), Teodorovic and Radivojevic (2000),
 Toth and Vigo (1997) Attanasio et al. (2004), Baugh et al. (1998), Beaudry et al. (2010),
 Braekers et al. (2014), Braekers and Kovacs (2016), Cordeau and Laporte (2003),
 Chassaing et al. (2016), Detti et al. (2017), Guerriero et al. (2013), Hall and Peterson (2013),
 Healy and Moll (1995), Ho and Haugland (2011), Kirchler and Wolfer Calvo (2013),
 Lehoude et al. (2013), Li et al. (2016), Masmoudi et al. (2016), Masson et al. (2014),
 Mauri and Lorena (2006), Melachrinoudis et al. (2007), Molenbruch et al. (2017), Muelas et al. (2013),
 Muelas et al. (2015), Neven et al. (2014), Paquette et al. (2013), Parragh et al. (2010),
 Parragh et al. (2015), Qu and Bard (2013), Ritzinger et al. (2016), Schilde et al. (2011),
 Schilde et al. (2014), Toth and Vigo (1997), Wolfer Calvo and Touati-Moungla (2011), Xiang et al. (2006),
 Xiang et al. (2008) Atahran et al. (2014), Chevrier et al. (2012), Cubillos et al. (2007),
 Cubillos et al. (2009), Hanne et al. (2009), Jorgensen et al. (2007), Masmoudi et al. (2017),
 Parragh et al. (2009), Rekiek et al. (2006), Zhang et al. (2015) Berbeglia et al. (2012),
 Crawford et al. (2007a), Crawford et al. (2007b), Espinoza et al. (2008a), Jain and Hentenryck (2011),
 Oberscheider and Hirsch (2016), Parragh and Schmid (2013), Ritzinger et al. (2016)
 Deflorio et al. (2002), Fu (2002b), Hall et al. (2012), Hyttia et al. (2012),
 Quadrifoglio et al. (2008), Schilde et al. (2011), Schilde et al. (2014), Shen and Quadrifoglio (2012),
 Shen and Quadrifoglio (2013) Detti et al. (2017), Lim et al. (2017), Masmoudi et al. (2017),
 Molenbruch et al. (2017a), Molenbruch et al. (2017b), Molenbruch et al. (2017c) Pimenta et al. (2017),
 Schonberger (2017), Masmoudi et al. (2018) Oscar Tellez et al. (2018), Drakoulis et al. (2018),
 Van Engelen et al. (2018), Gschwind et al. (2019), M Abedi et al (2019), Ramesh et al. (2019),
 Arslan Ali Syed et al. (2019), J Christiaens (2020), Masmoudi et al. (2020) Pacquay et al. (2020),
 Oscar Tellez et al. (2020), X Liang et al. (2020), Ramesh et al. (2020a), Ramesh et al. (2020b)
-

TABLE 2.7: Taxonomy: Approximation methods for solving DARP

an aspiration criterion can be used to override tabu moves under specific conditions, for example, when the newly found solution is better than the best solution found during the search. Intra-route local search operation can be additionally performed for intensification. Following this approach of Cordeau and Laporte 2003 [7], many authors (Melachrinoudis et al. 2007 [31]; Beaudry et al. 2010 [43]; Ho and Haugland 2011 [56]; Paquette et al. 2013 [69]; Detti et al. 2017 [101]) adopted TS approach to solve different variants of the DARP. Attanasio et al. 2004 [143] introduced a parallel tabu search using a multi-core CPU for dynamic DARP. Guerriero et al. 2013 [65] applied TS inside a hybrid GRASP algorithm for DARP. Wolfler Calvo and Touati-Moungla 2011 [140], Kirchler and Wolfler Calvo 2013 [67] applied granular tabu search for a DARP that increases service quality. In this algorithm, a granular neighborhood is defined to solve an assignment subproblem by concentrating only on the potentially good candidate solutions. This algorithm reported very good solutions in less than 60 sec, which is highly desirable for the DARP under the dynamic real-life settings.

The **variable neighborhood search** (VNS) framework was first proposed by Mladenovic and Hansen 1997 [163], and first applied to solve DARP by Parragh et al. 2010 [9]. The basic idea of VNS is to switch between multiple neighborhood search operators in order to swiftly escape the local optima. Parragh et al. 2010 [9] introduced three operators: exchange, chain, and zero-split. These authors also allow infeasible solutions to be accepted during the search process by using the relaxation mechanism of Cordeau and Laporte 2003 [7] while using the intra-route local search operation for intensification. Following that, Muelas et al. 2013 [68] devised relocation and exchange operations to solve large-scale problems. Detti et al. 2017 [101] mildly disturb the infeasible solutions in the hope to attain feasibility inside the VNS framework. Furthermore, the VNS approach is applied in different problem context such as Stochastic DARP (Schilde et al. 2011 [54], Schilde et al. 2014 [82]) and DARP with split-requests (Parragh et al. 2015 [93]). Recently, Molenbruch et al. 2017 [106] integrated a VNS approach within a multi-directional local search algorithm.

The **large neighborhood search** algorithm removes a large chunk of requests and reinserts better positions in the solution using destroy and repair operators, respectively. This type of approach was extended by Ropke and Pisinger 2006 [164] to solve VRPTW with an adaptive mechanism that alters the probability of

selecting the operator based on their past performance. The resultant approach is often referred to as adaptive large neighborhood search (ALNS), in which random removal, worst removal, sequential removal, route and related removal, random insertion, greedy insertion, k-regret insertion, space-time related insertion operators are often integrated. Masson et al. 2014 [77] apply the ALNS approach to solve DARP with transfers. Several variants of ALNS were introduced to solve different variants of DARP (Lehuede et al. 2014 [78], Braekers and Kovacs 2016 [97], Ritzinger et al. 2016 [100], Qu and Bard 2013 [70], Masmoudi et al. 2016 [99], Gschwind et al. 2019 [119]).

The **simulated annealing** (SA) algorithm accepts deteriorating solutions with metropolis criterion. This framework is termed as deterministic annealing (DA) when thresholding criterion is employed. Braekers et al. 2014 [11] first applied DA to solve DARP with several operators such as relocate, exchange, 2-opt*, r-4-opt, and route elimination in random order. They obtain best-known solutions for pure local search-based algorithms for general DARP instances. SA was employed by Baugh et al. 1998 [165] and Mauri and Lorena 2006 [141] with other metaheuristic approaches. Masmoudi et al. 2016 [99] integrated hybrid bee algorithm with DA and SA to solve multi-depot DARP.

The **evolutionary local search** framework was first applied to solve DARP by Chassaing et al. 2016 [12]. This approach is a hybrid of iterative local search with each iteration starting with a population of solutions. This approach performed better than Braekers et al. 2014 [11] for solving the general DARP.

2.3.3.3 Population-based Metaheuristics

The **genetic algorithm** (GA) framework is based on generating improved solutions by combining attractive attributes of two-parent solutions by encoding and crossover schemes. A GA in which the users are clustered into vehicles, and then inserted into a specific order was applied by Rekiek et al. 2006 [166], Jorgensen et al. 2007 [29], Hanne et al. 2009 [41]. Different encoding schemes (e.g. Cubillos et al. 2009 [36], Chevrier et al. 2012 [60], Atahran et al. 2014 [74], Zhang et al. 2015 [96]) were introduced for different variants of DARP.

Hybrid GA (otherwise, referred to as memetic algorithms) is an extension of evolutionary local search. Masmoudi et al. 2017 [103] developed this approach for

DARP with several operators, and the results are better than Chassaing et al. 2016 [12] for general DARP instances.

Parragh et al. 2009 [40] and Molenbruch et al. 2017 [106] employ path-relinking for bi-objective DARP.

2.3.3.4 Hybrid solution approaches

The *matheuristic* framework hybridizes the exact methods and meta-heuristic algorithms, which is an emerging trend in vehicle routing as defined by Archetti and Speranza 2014 [167]. Oberscheider and Hirsch 2016 [168] initially optimized user allocation into vehicles using an exact algorithm before invoking a tabu search strategy. Jain and Hentenryck (2011) [169] developed a hybrid LNS in which a constraint programming is used to complete a partial solution. Ritzinger et al. 2016 [100] performed insertions using DP inside the hybrid LNS framework. Parragh and Schmid 2013 [10] applied a hybrid LNS for solving DARP in which the column generation approach is employed. In a dynamic problem context, Berbeglia et al. 2012 [58] combine TS of Cordeau and Laporte (2003) [7] and constraint programming technique of Berbeglia et al. 2011 [49] to decide on the insertion possibility of a dynamic request. Gschwind and Drexler 2016 [98] present a hybrid LNS with DP to efficiently solve DARP.

2.3.4 Parallel Computing Approaches

The **parallel computing** techniques are promising to efficiently solve the DARP and they are essential for the time-critical applications (e.g. emergency transit, fire, or rescue operations). Attanasio et al. 2004 [143] applied tabu search using a multi-core CPU for the dynamic DARP. However, none of the journal publications are found (except the publications resulted out of this thesis) for the GPU-based solution techniques for the DARP. Ramesh et al. 2020a [130] (Chapter 3 of this thesis) present the GPU-accelerated framework to accelerate metaheuristic approaches for solving the dial-a-ride problem.

2.4 Discussion

Most of the contributions in the literature for the DARP are based on practical problems due to which the results were reported on the case-specific real-life instances. Due to this, it was very difficult to compare different algorithms proposed to solve DARP. A set of medium to large-scale benchmark instances provided by Cordeau and Laporte (2003) [7] was widely used to compare different metaheuristic approaches for solving DARP. Meanwhile, a set of small-scale instances provided by Ropke et al. 2007 was used to compare different exact methods to solve DARP. These instances are designed based on the real-life transportation dataset provided by the Montreal Transit Commission (MTC) of Canada. Later, these instances are adapted to account for different factors such as heterogeneous customers, vehicles, and multiple depots (Parragh 2011 [53], Braekers et al. 2014 [11], Masmoudi et al. 2016 [99]). For the single-objective general DARP, the best-known solutions are provided by Gschwind et al. 2019 [119], while Kirchler et al. 2013 [67] provide best-known results within 60 sec for the DARP that maximizes service quality. More recently, a real-life DARP dataset was introduced by Bongiovanni et al. 2019 [24], which is based on the UBER taxi trips in San Francisco City at United States of America (USA). As of 2019, the best-known solutions for these UBER instances are reported by Bongiovanni et al. 2019 [24]. The emerging trend in mathematical modeling is in terms of adapting the general DARP with more realistic features such as autonomous electric vehicles, dynamic factors, congestion, stochastic factors, and simulation-based modeling. Meanwhile, the emerging trend in solution method is towards developing hybrids of metaheuristics due to its promising capability to produce high-quality solutions, and development of parallel solution techniques to generate high-quality solutions for the dynamic problems in which the timely response can be critical (e.g. emergency applications, disruptions, today's ride-hailing transportation). We provide a detailed discussion on the future trends and emerging applications of DARP in Chapter 6.

2.5 Knowledge Gaps and Research Challenges

Despite the advancements in technologies, autonomous mobility-on-demand systems, for which the underlying optimization problem is DARP, are still being considered to be in the trial phase, as the commercial deployment is impeded by the following knowledge gaps. First, accelerating optimization algorithms using the latest computing technologies such as GPU to efficiently solve DARP under dynamic real-life settings. Second, effective models and algorithms for dial-a-ride operations under disruptive scenarios such as vehicle breakdown. Third, effective models and algorithms for dial-a-ride operations with electric autonomous vehicles and modern recharging infrastructures.

In the following, we summarize the knowledge gaps addressed in this thesis and key research challenges associated with each of these knowledge gaps.

First, technological advancements are constantly creating new trends to operate DAR services. To achieve the computational speeds necessary, especially in dynamic problems, parallel computation (perhaps using meta-heuristics) is a promising research direction that is currently lacking in the literature for DARP [5]. Specifically, GPU technology has evolved significantly in the past decade with the introduction of the Compute Unified Device Architecture (CUDA) programming model. Much progress has been made on computing technologies over the years, with the invention of high-performance parallel-computing hardware and its usage in non-graphical computationally intensive general-purpose applications. Most of the GPU-based works on routing problems do not show a thorough investigation of fully utilizing GPU (i.e., kernel concurrency, hierarchical device memory optimization, optimizing CPU-GPU communication, reducing thread divergence, adaptive kernel grid configuration). Without sufficiently addressing these challenges, the GPU-based algorithms might be computationally slower than the sequential CPU approaches, due to the huge overhead associated with CPU-GPU communication latency and read/write memory access latency. From the scientific perspective, it is critical to design methods that yield better and more robust solutions than existing methods for problems by fully utilizing state-of-the-art computing technology. Besides applying the conventional GPU approach for DARP, device-oriented optimization strategies are required to enhance the utilization of a state-of-the-art

GPU architecture (Tesla P100). Specifically, i) a device hierarchical memory optimization strategy to effectively map the DARP data structure to various GPU memories, ii) a method to adaptively select the kernel grid configuration for DARP kernels that maximizes GPU utilization, iii) optimizing data transfer: allocation of data in page-locked memory, segregation of the developed DARP kernels, and overlapping data transfer, iv) a scheme to reduce branch divergence during the process of generating neighborhood solutions for DARP, and v) concurrent kernel execution to enable overlapping data transfers between the developed GPU kernels and CPU functions. Furthermore, a framework should be designed in a generic way to easily integrate state-of-the-art optimization approaches. The choice of meta-heuristic optimizers should be selected based on their design simplicity, efficiency, and applicability for a wide range of problems. We attempt to address the above knowledge gaps and research challenges in Chapter 3.

Second, vehicle breakdown leads to major disruption of transportation services across the globe despite the technological advancements. This is substantiated by the fact that 32 million car breakdowns are recorded in the USA in 2015, which keeps increasing every year [18]. When considering an advanced user-oriented form of mobility such as DARP, vehicle breakdowns result in an inflated operational cost and damaged reputation for the service providers (operators of taxis, private buses, etc.). In literature, DARP with traffic [19] or demand uncertainty [20] can be addressed using predictive analytics. Meanwhile, in the case of vehicle breakdown, it is not possible to rely on prediction due to the highly uncertain nature of the event. The conventional method to handle breakdown is to rent a vehicle and dispatch in a predefined route, which may lead to a sub-optimal solution. Along the same lines, another important problem is to determine the minimum number fleet size required for an uninterrupted and high-quality transportation service, which is substantiated by the fact that New York City taxis travel without customers for about 40% of their time [17]. This also directly affects the fleet depreciation loss, maintenance cost, supporting infrastructure cost, etc. pose an additional burden on the service providers. The key research challenge in addressing the disruption management for fleet management problems such as DARP is to deploy efficient models and algorithms. In terms of the problem model, we need to consider realistic constraints such as immediate fleet re-routing after a vehicle breakdown, imposing decisions to serve all the customers to avoid damage on service providers' reputation, design of a realistic operational cost function, and dynamic updates in

travel times between the newly generated nodes, i.e., location of vehicles during vehicle breakdown and all the other nodes in the network. In terms of solution algorithm, there are two important challenges: a fast recourse method is necessary to quickly re-optimize the solution within a short time, and an efficient fleet management strategy is required that both reduce the fleet size as well as the operational cost on both the normal and disruptive scenarios. Furthermore, an assessment is required to determine the positive and negative effects of fleet size minimization during the normal and disruptive scenarios. Finally, to assess the real-world implications of the above techniques, it is necessary to model an operational cost function that considers costs of fuel, maintenance, parking, rental, cleaning, and wear and tear based on realistic estimates. We attempt to address the above knowledge gaps and research challenges in Chapter 4.

Third, the invention of autonomous electric vehicles has revolutionized intelligent transportation systems, and they are increasingly being used for mobility services to combat the issues of global warming, fuel consumption, air pollution, and road safety. The increase in the number of DAR service providers around the world substantiates the fact that the general public increasingly prefers on-demand mobility for their daily transportation needs. The key challenge of integrating the autonomous EVs into the daily transit system is to resolve the problem of "range anxiety", which means the anxiety of EV driver to locate the charging station before the battery runs out. The recharging infrastructure should be abundant and accessible to sustain and progress towards the idea of green transit systems. An existing study on the electric autonomous dial-a-ride problem [24] has considered the conventional static charging station (SCS), which has a slender infrastructure that may require costly changes and may cause large detours to recharge EVs. In the case of people-oriented DARP, customer satisfaction may be reduced when detouring to static/fixed recharging stations. Modern recharging infrastructures such as mobile energy disseminators (MED) are capable of recharging EV while the vehicle is on the move. In terms of modeling, the key challenge is to properly capture the intrinsic constraints related to battery management, recharge times, detours, and choices of charging stations besides the classical DARP features. In terms of algorithms, the challenge is to devise methods to efficiently schedule visits to new recharging infrastructures such as MED. Another challenge is that the computational complexity of the problem will be increased due to the advanced problem features, and the existing approaches such as exact methods in [24] may

not be suitable for realistic problems. Therefore, there is a great need to adapt state-of-the-art meta-heuristics for such a new complex, and a richer variant of DARP. We attempt to address the above knowledge gaps and research challenges in Chapter 5.

Chapter 3

A Generic GPU-Accelerated Framework for Dial-A-Ride Problem

Summary - Accelerating the performance of optimization algorithms is crucial for many day-to-day applications. Mobility-on-demand is one such application that is transforming urban mobility by offering reliable and convenient on-demand door-to-door transportation at any time. Dial-a-ride problem (DARP) is an underlying optimization problem in the operational planning of mobility-on-demand systems.

The primary objective of DARP is to design routes and schedules to serve passenger transportation requests with high-level user comfort. DARP often arises in dynamic real-world scenarios, where rapid route planning is essential. The traditional CPU-based algorithms are generally too slow to be useful in practice. Since customers expect a quick response to their mobility requests, there has been a growing interest in fast methods.

Therefore, in this chapter, we introduce a GPU-based solution methodology for the dial-a-ride problem to produce good solutions in a short time. Specifically, we develop a GPU framework to accelerate time-critical neighborhood exploration of local search operations under the guidance of metaheuristics such as tabu search, variable neighborhood search, and adaptive large neighborhood search. Besides, we propose device optimization strategies to enhance the utilization of a current-generation GPU architecture (Tesla P100).

We report speedup achieved by our GPU approach when compared to its classical CPU counterpart, and the effect of each device optimization strategy on computational speedup.

Results are based on standard test instances from the literature. Ultimately, the proposed GPU methodology generates better solutions in a short time when compared to the existing sequential approaches.

The preliminary results of this chapter are published as a conference paper in Ramesh et al. 2018a [6], and the journal version is published as Ramesh et al. 2020a [130] in the IEEE Transactions on Intelligent Transportation Systems.

3.1 Introduction

Mobility-on-demand (MoD) is a transformative and rapidly developing mode of transportation wherein the vehicles transport passengers in a given environment [13]. As autonomous vehicular (AV) technologies continue to emerge, it is more likely that the future MoD systems will rely on AVs to fulfill daily transportation needs. Moreover, MoD has been advocated as a key step toward sustainable urban mobility in the 21st-century [14]. These services often involve transportation of human passengers who place transportation requests with an origin and a destination location. The underlying optimization problem is usually modeled as the dial-a-ride problem, which is an advanced, user-oriented form of mobility service characterized by flexible routing and scheduling. DARP is becoming increasingly relevant in today's public and commercial transit, emergency transportation, ground, and seaport operations.

The vehicle routing problem (VRP) was first introduced in 1959 by Dantzig and Ramser [170]. Dial-a-ride problem (DARP), which is a special class of VRP, arises in the context of demand-responsive transportation. DARP deals with transportation of passengers between pickup and dropoff locations while satisfying constraints such as time-windows, user ride time, and route duration limits. DARP is formulated as a combinatorial optimization problem, and the objective is to design vehicle routes and schedules to serve all the users with minimal cost. DARP solution methodologies can be broadly classified into two categories: exact methods and heuristics. Exact methods guarantee optimal solutions while requiring a long execution time, which is suitable for small-scale problems. On the other hand, heuristics can generate solutions quickly without assuring optimality, which is suitable for large-scale problems.

Most of the methods from the literature focus on attaining optimal or near-optimal solutions while little attention has been paid towards developing methodologies to rapidly attain high-quality solutions. In recent times, the advancements in big data and telemetrics have transformed the dial-a-ride (DAR) systems. State-of-the-art technologies yield large amounts of data about transportation networks. The conventional DAR systems require advanced trip booking; however, the planned schedule may need to be adjusted on the fly due to unexpected events. With the advent of advanced machine learning and deep learning approaches, we can estimate the travel times on-the-fly with higher accuracy, and it enables fast algorithms on powerful hardware to achieve “real-time” trip planning. This trend may shift research on fast solution methodologies for DAR systems to facilitate real-time optimization. To achieve the computational speeds necessary, parallel computation (using metaheuristics) is a promising research direction that is currently lacking in the literature [5][171].

Graphics processing unit (GPU) technologies have rapidly evolved in the last decade through the advancements in very large-scale integration [172]. The application areas of GPUs are as diverse as image processing, machine learning, statistics, 3D reconstruction, game engines, and optimization. GPU technology has the scope of adding even more programmability and parallelism to the ever-evolving hardware architectures. Furthermore, the invention of compute unified device architecture (CUDA) has allowed the users to focus on high-performance computing concepts over the underlying hardware. CUDA programming model that includes C/C++ abstraction has greatly simplified the programmer’s burden in building GPU-enabled applications. Although the CUDA abstraction layer hides hardware details from the developer, a comprehensive set of hardware and software level constraints must be satisfied to fully exploit the GPU-acceleration. A CUDA kernel is a set of procedures executed in the GPU, which runs a grid of threads to perform parallel processing in GPU cores. A thread block contains a set of threads segregated into multiple thread warps, where each thread warp contains 32 threads that operate together to execute a single instruction on multiple data at a time. The programmer has control over setting the number of threads and blocks. However, the underlying hardware handles the order of thread execution. The warp scheduler of GPU hardware reduces the data transfer latency by performing arithmetic computation of certain warps while waiting for the read/write

process of other warps to complete. Because of this, the arithmetic intensity is kept high at all times to enhance the overall computational efficiency.

Although GPU computing is recognized as an effective way to improve computational efficiency, the development of GPU-based metaheuristics is not straightforward. Various challenges pertaining to hierarchical memory management have to be addressed. Hence, designing GPU-accelerated algorithms based on neighborhood structures for solving real-world optimization problems can be very challenging [173]. To the best of our knowledge, there is no substantive existing work on GPU-based methods for DARP in the literature. This has motivated us to propose a generic GPU-accelerated framework and integrate it with three well-known metaheuristics: tabu search, variable neighborhood search, and adaptive large neighborhood search for solving DARP. Simulations are conducted on standard benchmark instances from the literature and the results are compared against other relevant methods.

The remainder of the chapter is organized as follows. In Section 3.2, we review the relevant literature. In Section 3.3, we describe the DARP formulation. In Section 3.4, we describe the proposed generic GPU framework. In Section 3.5, we present the proposed device optimization strategies. In Section 3.6, we detail the GPU-based metaheuristics for DARP. We present the simulation results in Section 3.7 and offer concluding remarks in Section 3.8.

3.2 Relevant Literature

In this section, we discuss the relevant DARP studies (Section 3.2.1), GPU-based solution approaches (Section 3.2.2), and the contributions of this chapter (Section 3.2.4) to fill the research gap presented in Section 3.2.3.

3.2.1 Relevant studies on DARP

The Dial-a-ride (DAR) system was first introduced by Wilson [26] in 1972. Applications have been reported from several cities such as Bologna [174], Copenhagen

[175], Milan [32], etc. Several exact methods have been proposed for the single-vehicle [155][156][176], and multi-vehicle [177][28] cases. The largest problem optimally solved by an exact method has 9 vehicles and 96 requests, that reportedly consumed 899 sec, which in terms of both computational time and problem scale, is far higher to be suited for real-world usability.

To reduce the computation time, heuristics for different versions of DARP were introduced [137][175]. A heuristic for the transportation of disabled people with a homogeneous fleet of vehicles has been proposed in [160]. A DARP with the homogeneous fleet was addressed by a tabu search heuristic in [7] and a variable neighborhood search in [9]. A DARP with transfer points was solved by adaptive large neighborhood search [77]. A DARP with heterogeneous users was addressed by methods such as deterministic annealing [11], evolutionary local search [12], and hybrid genetic algorithm [103]. A DARP with a fixed number of vehicles and whose objective function maximizes service quality for passengers is discussed in [32][9][29][138][178][10]. We study this version of DARP due to its resemblance towards the current DAR services around the world: 1) The constraints of this DARP seems to be in line with the practice of several North American transporters [28]. 2) The information regarding time window widths, vehicle capacity, route duration, and maximum ride time was provided by the Montreal Transit Commission (MTC), Canada. The instances are generated according to realistic assumptions [29]. 3) The assumptions of homogeneity of vehicles and their capacities are reasonable in the employment of autonomous electric mini-buses, as nowadays such vehicles mount the same commercial batteries and do not significantly differ in terms of dimensions, capacity, and weights [24]. 4) The assumptions regarding vehicle capacities reflect the situation of transporters using mini-buses for the transportation of individuals or groups of individuals [28]. 5) This version of DARP treats multiple objectives that aim to improve overall service quality which is of primary importance in DARP [29][139][67]. 6) Heuristic solutions have been reported in real-time for medium to large-scale problems, which is applicable in the current world since the users appreciate shorter response time [5]. Moreover, the ride-hailing companies have recently started to recognize the importance of response time to customer requests [179][180][181].

In the context of attaining quick solutions (i.e., within 60 sec) for DARP, the Granular Tabu Search proposed by Kirchler and Wolfler [67] performs the best when

compared to the other existing algorithms, which are (i) Variable Neighborhood Search [9], (ii) Genetic Algorithm [29], and (iii) Tabu Search [7]. We could not find any other algorithms for direct comparison since many [11][77][12] focus on attaining optimal solutions over a long time and do not report solutions in a short time. Therefore, we regard this Granular Tabu Search [67] as the best-performing approach in the context of fast methodologies for DARP. For a broader overview of solution methods and variants of DARP, we refer to [5], a recent survey in the literature.

3.2.2 Relevant studies on GPU-based methods

The classical routing and scheduling problems are often solved by local search based heuristics. Such methods usually start from an initial solution which is iteratively improved by exploring the problem-specific neighborhood in the search space. Recently, GPUs are being used for time-efficient neighborhood exploration for different optimization problems.

Several authors have considered the tabu search method as a starting point to integrate a GPU-based local search method to solve the problems not limited to VRP, TSP, and Q3AP. Janiak et al. (2008) [182] have solved a traveling salesman problem (TSP) and a flow shop scheduling problem (FSP). TSP is a classical routing problem that aims to design routes for a salesman who must travel between several cities. FSP is a classical scheduling problem that aims to design an optimal schedule for a set of jobs to be processed on a set of machines. These authors apply the tabu search method while considering the insertion and swap type of moves in GPU for generating neighborhood solutions. Luong et al. (2010) [183] have solved a quadratic 3-dimensional assignment problem (Q3AP), which consists of determining permutation matrices to minimize a quadratic function and has found applications in wireless communication systems [184]. These authors apply swap type of moves during permutations to generate neighborhood solutions in GPU. Michal and Barnes (2011) [185] have solved an FSP with a swap-type neighborhood move. When compared to Janiak et al. (2008), these authors have evaluated various configurations of the number of threads per block to find the best configuration. Szymon and Dominik (2013) [186] have applied a GPU-based local search with a tabu list to optimize routes for a general vehicle routing problem. Coelho

et al. (2016) [187] have applied a variable neighborhood search for single-vehicle VRP with pickup and delivery, in which neighborhood solutions are evaluated using GPU.

In comparison with the literature, we solve a different problem known as DARP under the category of VRP. DARP is a difficult problem with time windows and maximum user ride-time constraints, and it generalizes the VRP with pickup and delivery. Attaining a feasible solution for DARP is itself NP-hard since it also generalizes the traveling salesmen problem (TSP) with time windows [28]. The solution space and the evaluation function are problem-dependent. The DARP consists of designing optimal routes and schedules for multiple vehicles to perform door-to-door passenger transit service. We perform insertion-type neighborhood moves similar to Janiak et. al. [182], however, in our case, we additionally consider pair-precedence constraints, i.e., a drop off node should always be placed after its corresponding pickup node of the request. The kernel configuration is an important factor to maximize GPU hardware utilization. Since the exhaustive search of the best configuration of threads and blocks for various instances can be time-consuming (as demonstrated by Szymona and Dominik (2013) [186]), we present a method that optimizes the values of the block and thread for the GPU kernels. We develop an evaluation function (see Algorithm 2) to find the feasibility of the DARP solution in GPU. Furthermore, to maximize the GPU utilization, we perform a problem-dependent device-hierarchical memory optimization of data structure, which is regarded as challenging depending upon the considered problem by Luong et al. (2010) [183]. We have also adapted the branch refactoring method of Imen Chakroun et al. (2013) [188] in order to reduce the well-known thread divergence problem of GPU. We perform overlapping data transfers and design multiple kernels deliberately to perform concurrent kernel execution for increased GPU usage. Finally, we also demonstrate the computational speedup achieved by the proposed GPU approach that consists of device optimization strategies in comparison with the conventional approach.

In the literature, ParadisEO [189] is a software framework dedicated to the reusable design of parallel hybrid metaheuristics, in which “the user does not need to build his or her own CUDA code for the specific problem to be solved”. On the other hand, our GPU framework has DARP-specific CUDA codes built into it. Our work

differs from ParadisEO in the following ways: 1) CUDA kernels that employ concurrent kernel launching scheme, i.e., the kernels are launched simultaneously in an attempt to maximize the computational resource usage. 2) DARP-specific scheme for memory management, i.e., we perform selective data transfer and contiguous memory copy after the kernel executions in order to minimize the CPU-GPU communication latency. 3) Device-specific optimization strategies for the current generation GPU architecture to effectively accelerate DARP solvers, e.g., adaptive kernel grid configuration to dynamically optimize the thread and block size based on the problem size. In summary, the proposed GPU framework is tailored for DARP and has the potential to be integrated with the existing ParadisEO framework. However, we could not make a direct comparison since no results of using this framework have been found in the literature for DARP or any extensions of VRP.

3.2.3 Research gap

There is a growing interest in fast methods since users expect the service providers to provide a solution quickly [5]. This is also evident from the recent dial-a-ride surveys, which suggest that the users tend to prioritize short response time, and the ride-hailing companies have started to act accordingly [179][180][181].

In the context of attaining high-quality solutions in a short time, especially for problems of a realistic scale, the traditional CPU based approaches may be too slow to be applicable. To the best of our knowledge, GPU-accelerated methodologies for DARP are not reported in the literature as stated in Section 3.2.1. Besides, we observe that none of the works in Section 3.2.2 show a thorough investigation of fully utilizing GPU (i.e., kernel concurrency, hierarchical device memory optimization, optimizing CPU-GPU communication, reducing thread divergence, adaptive kernel grid configuration). From the scientific perspective, it is critical to design methods that yield better and more robust solutions for important real-world problems such as DARP by fully utilizing state-of-the-art computing technology.

3.2.4 Our Contributions

The main contribution of this chapter is the development of a GPU-based solution methodology for DARP with the objective of producing good solutions in a short time. The idea of accelerating local search operation in GPU has been introduced earlier for other optimization problems [182][183][185][186], but never applied for DARP. In our work, besides applying this conventional GPU approach for DARP, we develop five device-oriented optimization strategies to enhance the utilization of state-of-the-art GPU architecture.

Specifically, we propose the following device-oriented optimization strategies: i) a device hierarchical memory optimization strategy to effectively map the DARP data structure to various GPU memories, ii) a method to adaptively select the kernel grid configuration for DARP kernels that maximizes GPU utilization, iii) an optimization scheme to reduce data transfer latency, i.e., allocation of data in page-locked memory, segregation of the developed DARP kernels, and overlapping data transfer, iv) a scheme to reduce branch divergence during the process of generating neighborhood solutions for DARP, and v) a concurrent kernel execution for overlapping data transfers between the GPU kernels and CPU functions.

The role of the GPU in the proposed framework is to simultaneously generate and evaluate various solutions during each iteration. The tasks of generating and evaluating such combinations of solutions are very time-intensive, therefore we use GPU for parallel processing. Our GPU approach is generic in the sense that we iteratively select the best admissible solution in CPU followed by GPU-based local search. Here, the choice of requests will be given as input and the evaluated neighborhood solutions will be the output. Therefore, any type of removal and re-insertion operators could be utilized to dislodge the requests involved during the local search operation. Moreover, any metaheuristic that utilizes local search operation could be integrated with the proposed approach. In this chapter, we integrate the proposed GPU framework with three metaheuristics: tabu search, variable neighborhood search, and adaptive large neighborhood search. The choice of metaheuristics can be justified by their simplicity, efficiency, and applicability for a wide range of problems [190][191][192].

Experiments are conducted on a current-generation GPU, the Tesla P100. Results are based on standard benchmark instances from the literature. Firstly, we demonstrate the speedup achieved by the proposed GPU approach in comparison with its conventional CPU counterpart. We also show the influence of the proposed device optimization strategies on computational speedup. Secondly, we compare the solution quality of our approach with the existing best-performing approaches. Finally, we discuss the effectiveness of our proposed GPU method for DARP under dynamic real-time settings.

The preliminary work of this chapter can be found in [6], in which we have described a basic implementation of tabu search on GPU to solve the dial-a-ride problem. In this current version, we have added the following improvements: i) new device-oriented optimization strategies to improve the computational speedup, with extended demonstrations on the effectiveness of each strategy, ii) generalized the GPU framework and integrated with metaheuristics such as tabu search, variable neighborhood search heuristic, and adaptive large neighborhood search, and finally compared the solution quality with existing approaches.

3.3 Formulation of Dial-A-Ride Problem

In this section, we describe the notations and mathematical formulation for dial-a-ride problem. Table 3.1 describes the terminologies used in this chapter.

In DARP, a set of n customer requests for vehicles, where each customer specifies a pickup and a drop-off location. Passengers also specify their preferred time windows during which they have to be served. All vehicles start from the depot and return to it once the transit service gets completed. In this problem, all the transportation requests are known in advance. In the following, we present the same formulation described in Chapter 2, while additionally formulate and discuss the choices/justifications of the objective function $f(x)$ in order to make this chapter sufficiently self-contained.

We assume that a fleet of m vehicles represented by set K must serve n customer requests. The customer requests lead to n pick up and n drop off nodes. The nodes form a directed graph (N, A) graph where $N = P \cup D \cup \{0, 2n + 1\}$, $P = \{1, \dots, n\}$ and $D = \{n + 1, \dots, 2n\}$. Nodes 0 and $2n + 1$ represent the origin and destination

Notation	Description
Depot	Initial and terminal node for each vehicle.
P	Location at which the passenger is picked up.
D	Location at which the passenger is dropped-off.
N	Set of all pick-up points, drop-off points and depot.
n	Total number of customers.
K	Fleet size or a set of vehicles serving the customer requests.
k	A vehicle from the fleet of size m .
E	Set of paths joining the nodes.
T'_k	Route duration of k^{th} vehicle.
T_k	Maximum route duration limit of k^{th} vehicle.
A_i^k	Time at which a vehicle k arrives at node i .
B_i^k	Time at which a vehicle k begins its service at node i .
C_i^k	Time at which a vehicle k ends its service at node i .
D_i^k	Time at which a vehicle k departs from node i .
W_i^k	Waiting time of a vehicle k at node i .
L_i^k	Ride time of the customer i in the vehicle k .
d_i	Service time of the vehicle at node i .
L	Maximum ride time limit of a passenger.
$c(x)$	Total distance traveled by all the vehicles to serve the customers.
Q_i^k	Total number of passengers on the vehicle k after visiting node i .
q_i	Load available at the i^{th} node.
S_k	Seating capacity of the k^{th} vehicle.
e_i	Earliest time at which the service can begin in the node i .
l_i	Latest time beyond which the node i cannot be serviced.
t_{ij}	Travel time between the node i and node j .
t^k	Current time of vehicle k .
x	A DARP solution that consists of m vehicle routes and schedules.
$f(x)$	Objective function of a DARP solution x .
x_{ij}^k	Binary decision variable; 1 if arc (i, j) is traversed by vehicle k and 0 otherwise.

TABLE 3.1: Terminology.

depots. Any arc (i, j) denotes a path from the i^{th} to the j^{th} node. The time to travel the arc (i, j) is t_{ij} and the cost of routing is c_{ij} . We adopt the dial-a-ride problem of [29], which is formulated as follows,

$$\begin{aligned}
 & \text{Min} f(x) \\
 & \text{subject to,}
 \end{aligned} \tag{3.1}$$

$$\sum_{j \in N} x_{0j}^k = 1 \quad \forall k \in K, \quad (3.2)$$

$$\sum_{i \in D} x_{i,2n+1}^k = 1 \quad \forall k \in K, \quad (3.3)$$

$$\sum_{j \in N} x_{ji}^k - \sum_{j \in N} x_{ij}^k = 0 \quad \forall i \in P \cup D, k \in K, \quad (3.4)$$

$$\sum_{k \in K} \sum_{j \in N} x_{ij}^k = 1 \quad \forall i \in P \cup D, \quad (3.5)$$

$$\sum_{j \in N} x_{ij}^k - \sum_{j \in N} x_{n+i,j}^k = 0 \quad \forall i \in P, k \in K, \quad (3.6)$$

$$Q_j^k \geq (Q_i^k + q_j)x_{ij}^k \quad \forall i \in N, \forall j \in N, \forall k \in K, \quad (3.7)$$

$$\max(0, q_i) \leq Q_i^k \leq \min(C_k, C_k + q_i) \quad \forall i \in N, \forall k \in K \quad (3.8)$$

$$B_j^k \geq (B_i^k + d_i + t_{ij})x_{ij}^k \quad \forall i \in N, \forall j \in N, \forall k \in K, \quad (3.9)$$

$$e_i \leq B_i^k \leq l_i \quad \forall i \in N, \forall k \in K, \quad (3.10)$$

$$L_i^k = (B_{n+i}^k - (B_i^k + d_i))x_{ij}^k \quad \forall i \in P, \forall k \in K, \quad (3.11)$$

$$t_{i,n+i} \leq L_i^k \leq L \quad \forall i \in P, \forall k \in K, \quad (3.12)$$

$$B_{2n+1}^k - B_0^k \leq T_k \quad \forall k \in K, \quad (3.13)$$

The optimization problem (5.1) minimizes routing cost $c(x) = \sum_{(i,j) \in A, k \in m} x_{ij}^k t_{ij}$, excess ride time $r(x) = \sum_{i \in n} (B_i - D_i - t_{i+i-})$, waiting time of passengers on board $l(x) = \sum_{i \in n} w_i (Q_i - q_i)$, route durations $g(x) = \sum_{k \in m} A_{2n+1}^k - D_0^k$, early arrival times at pickup and delivery nodes $e(x) = \sum_{i \in n} (e_i - A_i)^+$, and finally the number of unserved requests $k(x) = n - \sum_{k \in m, ij \in n} x_{ij}^k$. The parameters $w_1, w_2, w_3, w_4, w_5, \alpha$ are statically set according to the application requirement. This objective function has been first introduced in [29], which states that the cost of transportation of the customers is estimated to be the actual transportation cost and a 'cost of inadequate service'. The actual transportation cost represents the total vehicle travel time. The cost of inadequate service represents the excess user ride times and waiting times in the vehicle. Excess ride time is the difference between the total time spent by the customer inside a vehicle and the direct travel time between origin and destination of the customer, which provides a better estimate of user convenience.

The constraints specified in Eqs. (5.2)–(5.3) ensure that all the vehicle routes start and end at the depot. Eqs. (5.4)–(5.5) make it mandatory for every node to be visited by only one vehicle such that any vehicle reaching a particular node must exit it. Eq. (5.6) ensures that the customers are picked up and dropped off by the same vehicle. Additionally, every vehicle must not exceed its maximum capacity, and the maximum ride time constraints must hold for the vehicles and the customers. Eq. (5.7) updates the load of the k^{th} vehicle once it reaches node j from node i . The constraint in Eq. (5.8) ensures that the capacity of vehicle k is not exceeded at any of the nodes. The time consistency is maintained by Eq. (5.9), which ensures that node j can be serviced only after the arrival of the vehicle. Eq. (5.10) specifies the time window of the service at each location and makes sure that service begins only within the windows. The constraints in Eqs. (5.11)–(5.12) guarantee that the maximum ride time of a customer is not exceeded. Finally, the constraint on the route duration of any vehicle k is ensured by Eq. (5.13).

3.4 Proposed generic GPU framework

In this section, we introduce the building blocks of the GPU framework and explain how they are integrated to solve DARP. We observe the following points from our literature review on solving DARP: a) metaheuristics are preferred over exact methods due to their efficiency in attaining good solutions, despite the lack of assurance to achieve optimality, b) most metaheuristics are built upon a local search process. Therefore, we describe the general framework to solve DARP as follows: 1) for a given problem, construct an initial set of vehicle routes, 2) perform a local search process, 3) under the guidance of metaheuristic, choose the next best solution in the defined neighborhood, 4) repeat steps 2-3, until the stopping criterion is satisfied. From our preliminary results, it was identified that the local search process consumes most of the execution time during the program. Therefore, we focus on designing CUDA kernels that perform low-latency local search operation in GPU to efficiently solve DARP.

The proposed GPU framework consists of the class structure and DARP scheduling algorithm in Section 3.4.1, GPU-accelerated local search algorithm in Section 3.4.2, and CUDA kernel functions in Section 3.4.3. The generic GPU framework

presented in this section will be integrated with metaheuristic approaches in Section 3.6 to efficiently solve the DARP.

3.4.1 Class structure

Algorithm 1 struct GPU

```

1: int id
2: int start, gap, request, target
3: int *route, routesize, routecost
4: __device__ void dev_scheduler(dev_problem)
5: {
6:   Load constraint evaluation
7:   Time window constraint evaluation
8:   Route duration constraint evaluation
9:   Ride time constraint evaluation
10: }
```

Algorithm 2 GPU-based scheduler: dev_scheduler()

Require: Original route *route*, Problem *dev_problem*

Ensure: Update *routecost*

- 1: Create Route *R* // register memory
 $R \leftarrow route$ // global memory to register memory
 Allot $*A, *B, *C, *D, *W, *L, *q(x_k), *d(x_k), w(x_k), t(x_k)$ // register memory
 set $D_0 \leftarrow e_0$ // constant memory to register memory
 - 2: Compute A_i, w_i, B_i, D_i, y_i for each node i in route k
 If some $B_i > l_i$ or $y_i > Q$, goto step 8
 - 3: Compute F_0
 - 4: Set $D_0 = e_0 + \min(F_0, \sum_{0 < p < q} w_p)$
 - 5: Update A_i, w_i, B_i and D_i
 - 6: Compute all T_i^{ride}
 If all $T_i^{ride} < T^{ride}$ goto step 8 // faster read operation since the instance is stored in cached constant memory
 - 7: For every node j that is an origin
 - (a) Compute F_j
 - (b) Set $w_j := w_j + \min(F_j, \sum_{j < p < q} w_p)$;
 $B_j := A_j + w_j$;
 $D_j := B_j + d_j$
 - (c) Update T_i^{ride} for each request i whose destination is after j
 - (d) If all $T_i^{ride} \leq T^{ride}$ for all i whose destinations lie after j , return true
 // this step consumes more than two-third of evaluation time and it is slower in GPU due to the global memory. Therefore, we use the GPU register memory whenever desired, which is several times faster than the GPU global memory.
 - 8: Compute the value of *routecost*.
-

We first create a class that comprises data elements and methods. These methods are device-specific, i.e., they perform operations on the data elements in GPU. We implement the vehicle route as a pointer-to-array, while vertices, route size, and vehicle ID are declared as data elements. Algorithm 1 describes the GPU class. An object of this class can be instantiated to represent a single-vehicle route in GPU.

To evaluate a vehicle route, we design a GPU-based scheduling heuristic as described in Algorithm 2. The notations A, B, C, D, W, L represent vehicle arrival time, service start time, service end time, departure time, waiting time, and user ride time at vertices, respectively. For a DARP solution x , let $c(x)$ denote the total travel time of the fleet, and let $q(x), d(x), w(x)$ and $t(x)$ denote the total violation of load, duration, time window and ride time constraints, respectively.

$$q(x) = \sum_{\forall k} \max(q_{k,max} - Q_k, 0), \quad (3.14)$$

$$d(x) = \sum_{\forall k} \max(T'_k - T_k, 0), \quad (3.15)$$

$$w(x) = \sum_{\forall i} [\max(B_i - l_i, 0) + \max(B_{i+n} - l_{i+n}, 0)], \quad (3.16)$$

$$t(x) = \sum_{\forall i} \max(L_i - L, 0). \quad (3.17)$$

To set the beginning of service in the best possible way, such that route duration is minimum and ride time limits are respected wherever possible, the route evaluation scheme introduced by Cordeau and Laporte (2003) [7] is applied. It is based on the forward slack time concept of [193] in Eq.3.18.

$$F_0 = \min_{i \leq j \leq q} \{W_p + (\min_{i < p \leq j} \{l_j - B_j, L - L_j\})\}. \quad (3.18)$$

The slack at vertex j is the cumulative waiting time until j , plus the minimum of the difference between the end of the time window and the beginning of service at j , and the difference between the maximum user ride time and L_j . The forward time slack at vertex i is the minimum of all slacks between i and q . F_0 thus gives the maximum amount of time by which the departure from the origin depot can be delayed (with equal time window violations and smaller or equal ride time violations) to yield a modified route of minimal duration. Furthermore, Cordeau and Laporte [7] observed that delaying the departure by $\sum_{0 < p < q} W_p$ does not affect the arrival time at q , whereas delaying the departure by more will only increase

the arrival time at q by as much. Consequently, the departure from the depot should only be delayed by at most $\{F_0, \min_{0 < p < q} W_p\}$. The forward time slack can also be applied to delay B_i at each origin vertex such that the ride time of the corresponding request is reduced. This reasoning led Cordeau and Laporte (2003) [7] to an eight-step evaluation scheme. In their scheme, the steps (1–2) minimize time-window violations $d(x_k)$. In steps (3–6), route duration violations $w(x_k)$ are minimized without increasing time window violations $d(x_k)$. The load violations $q(x_k)$ are irreparable. Step (7) reduces ride-time violations, which is based on the forward slack time technique of [193].

Algorithm 2 is the proposed DARP solution evaluation function, which is based on [7]. The vehicle route is fixed throughout the execution of this algorithm for each input. A vehicle route is defined as a sequence of pickup and drop-off operations. Algorithm 2 takes a vehicle route as its input and produces optimized schedules for the given vehicle route as its output. An optimized schedule is the one that results in the minimum constraint violation for the given vehicle route. We denote M_1 and M_2 as the solution evaluation method of [7] and our Algorithm 2, respectively. M_1 and M_2 have the same scheduling procedure, i.e., both attain the same output for any given input. M_1 and M_2 run on different computational hardware, i.e., M_1 runs in CPU, and M_2 runs in GPU. We have designed M_2 in such a way that it leverages the memory performance of GPU. There are six different GPU memories in the hardware architecture of the Tesla P100 GPU. Each memory architecture has a specific advantage, and our contribution is to map the DARP-related variables to these various memory units. This process of memory-mapping is detailed in Table 3.2, illustrated by Figure 3.1, and elaborated in Section 3.5.1. In the description of M_2 , we have detailed how the allocation of different DARP-related variables to different GPU memories has improved the computational performance of M_2 in GPU. The role of the GPU in M_2 is to efficiently execute the scheduling algorithm in parallel. In the proposed GPU framework, many such instances of M_2 are run simultaneously in GPU to evaluate the schedules of various vehicle routes during each iteration.

Algorithm 3 GPU-accelerated Local Search**Require:** DARP Instance, Current Solution x **Ensure:** Crux of Evaluated Neighborhood Routes

- 1: Allocate GPU resources.
- 2: Initialize B, T, S, M .
- 3: **for all** requests $i \in req_list$ **do**
- 4: Copy data from CPU to GPU
- 5: *Preprocessor* Kernel $\lll B, T, S, M \ggg$ ($gpu[i]$)
- 6: *Insertion* Kernel $\lll B, T, S, M \ggg$ ($gpu[i]$)
- 7: *Evaluation* Kernel $\lll B, T, S, M \ggg$ ($gpu[i]$)
- 8: *Pipelining* Kernel $\lll B, T, S, M \ggg$ ($gpu[i]$)
- 9: Copy data from GPU to CPU
- 10: **end for**
- 11: Release GPU resources.

3.4.2 Local search operation

In this section, we present the GPU-accelerated local search algorithm, which will be integrated into the metaheuristics in Section 3.6. This algorithm is built as a host function that invokes CUDA kernels that run on the GPU. The implementation details are presented in Algorithm 3. This algorithm identifies all neighbors of the current solution, evaluates them, and chooses the best one for the next iteration. In this context, a neighbor refers to a solution generated by performing a neighborhood move onto the current solution, i.e., moving a pair of vertices that belong to a request from its current route to another route. Before launching CUDA kernels, the input arguments (i.e., current solution and DARP instance) are transferred from CPU to GPU; upon launching the kernels, parallel computation is performed in the GPU; finally, the output is copied back to the CPU. The grid size (B), block size (T), stream ID (S), and the GPU shared memory size (M) are initialized according to the current solution, which are detailed in Section 3.5.2. For every request, we launch a set of CUDA kernels to construct and evaluate the neighborhood solutions, namely 1) *Preprocessor*, 2) *Insertion*, 3) *Evaluation*, and 4) *Pipelining*. Algorithms 4-7 describe the implementation of the CUDA kernels.

3.4.3 CUDA kernels

This section describes the four new CUDA kernels that will be invoked from the GPU-accelerated local search algorithm presented in Section 3.4.2. In the *Preprocessor* kernel function, each thread is responsible to initialize data elements associated with a unique object instantiated with the GPU class. These elements will be used by subsequent kernels to construct and evaluate the solution. The procedure is described as follows: i) initialize the value of the variable idx as a function of threads and blocks, ii) for all values of idx less than N_{size} , where N_{size} is the number of solutions that is in the subset of neighborhood associated with request i , perform step iii, iii) for all nodes in the route $k \in K$, while ensuring the pair-precedence constraints, initialize the values of $start$, gap , $target$, and $routesize$. Algorithm 4 describes the implementation of the *Preprocessor* kernel.

Algorithm 4 Preprocessor Kernel

Require: Request req , Current Solution x

Ensure: Preprocessed gpu objects

```

1: int  $idx = threadIdx.x + blockIdx.x * blockDim.x$ ;
2: if  $idx < gpu[idx].N_{size}$  then
3:   for all ( $index$ ) in all routes  $k \in K$  do
4:     while ensuring the pair-precedence constraints do
5:       Initialize  $gpu[idx].start \leftarrow index$ 
6:       Initialize  $gpu[idx].gap \leftarrow offset$ 
7:       Initialize  $gpu[idx].target \leftarrow k$ 
8:       Initialize  $gpu[idx].routesize \leftarrow x[k].size + 2$ 
9:     end while
10:  end for
11: end if

```

In the *Insertion* kernel function, each thread is responsible to construct a unique neighborhood solution. The procedure is described as follows: i) initialize the value of idx as a function of threads and blocks, ii) for all values of idx less than N_{size} , initialize the value of $vehicle$ that represents the target vehicle ID in which the request req will be inserted, iii) for all the values of the variable $tid \in [0, gpu[idx].routesize]$, insert the pickup and drop-off vertices associated with request req in $vehicle$ at the $start$ and gap positions, respectively. As a result, all the neighborhood solutions with respect to the current solution will be constructed. The indexing scheme with $state1$ and $state2$ optimizes branch divergence as detailed in Section 3.5.4. Algorithm 5 describes the implementation of the *Insertion* kernel.

Algorithm 5 Insertion Kernel**Require:** Request req , Current Solution x **Ensure:** Constructed Neighborhood Routes

```

1: int  $idx = threadIdx.x + blockIdx.x * blockDim.x$ ;
2: bool  $state1, state2$ ; int  $node, vehicle$ ;
3: if  $idx < gpu[idx].N_{size}$  then
4:    $vehicle \leftarrow gpu[idx].target$ 
5:   for  $\forall tid \in [0, gpu[idx].routesize]$  do
6:      $node \leftarrow x[vehicle].route[tid]$ 
7:      $state1 \leftarrow tid \geq (gpu[idx].start)$ 
8:      $state2 \leftarrow tid \geq (gpu[idx].start + gpu[idx].gap)$ 
9:      $gpu[idx].route[tid + state1 + state2] \leftarrow node$ 
10:  end for
11: end if

```

In the *Evaluation* kernel function, each thread is responsible to evaluate the constructed neighborhood solutions. The procedure can be described as follows: i) initialize the value of idx as a function of threads and blocks, ii) for all values of idx less than N_{size} , launch the device function *dev_scheduler* (Algorithm 2), which schedules the vehicle routes and evaluates the value of the objective function. As described earlier, the Algorithm 2 reduces the constraint violations. The intuition behind separating the *Insertion* and *Evaluation* kernels (Section 3.5.5) is to overlap the launch latency associated with the GPU kernels that belong to different requests. Algorithm 6 describes the implementation of the *Evaluation* kernel.

Algorithm 6 Evaluation Kernel**Require:** Pointer to *dev_problem***Ensure:** Evaluated Neighborhood Routes

```

1: int  $idx = threadIdx.x + blockIdx.x * blockDim.x$ ;
2: if  $idx < gpu[idx].N_{size}$  then
3:    $gpu[idx].dev\_scheduler(dev\_problem)$ 
4: end if

```

In the *Pipelining* kernel function, each thread is responsible to copy the data from one object to another, which will be transferred to the CPU later. The objective of developing this kernel is to transfer only selective information back to the CPU, thereby reducing the CPU-GPU communication. This procedure is described as follows: i) initialize the value of idx as a function of threads and blocks, ii) for all values of idx less than N_{size} , copy a selective set of information about the solution into an object (i.e., data structure) in the GPU global memory. This selective information contains the value of request ID, current vehicle ID, new vehicle ID,

and the positions in which the request will be inserted. Following that, all these objects will be copied from GPU to CPU. The justification for this kernel lies in the fact that the peak bandwidth between the device memory and GPU is much higher than that of the CPU-GPU memory transaction. Algorithm 7 describes the implementation of the *Pipelining* kernel.

Algorithm 7 Pipelining Kernel

Require: Pointer to *temp_cost*, Current Solution *x*

Ensure: Crux of Evaluated Neighborhood Routes

```

1: int idx = threadIdx.x + blockIdx.x * blockDim.x;
2: if idx < gpu[idx].N_size then
3:   temp_cost[idx].reset();
4:   for  $\forall k \in [0, m]$  do
5:     if k == gpu[idx].target then
6:       temp_cost[idx]  $\leftarrow$  x[k].route_cost
7:     else
8:       temp_cost[idx]  $\leftarrow$  gpu[idx].route_cost
9:     end if
10:  end for
11:  temp_cost[idx].check_feasibility()
12:  temp[idx].vehicle  $\leftarrow$  gpu[idx].target
13:  temp[idx].request  $\leftarrow$  gpu[idx].request
14:  temp[idx].position  $\leftarrow$  gpu[idx].position
15: end if

```

The metaheuristics based on local search can be accelerated by parallel neighborhood exploration using Algorithm 3. The direction of the search can be refined by the metaheuristic procedures in an attempt to identify optimal solutions (Section 3.6).

3.5 Proposed Device Optimization Strategies

The GPU framework in the previous section does not fully exploit the potential of GPU, impeded by the issues of memory throughput, device occupancy, data transfer latency, thread divergence, and kernel launch latency. Therefore, in this section, we present a set of optimization strategies to circumvent the performance penalties associated with the GPU framework. In Section 3.5.1, we present the hierarchical memory optimization strategy to effectively map the DARP data structure into

various GPU memories. In Section 3.5.2, we present the adaptive kernel grid configuration to heuristically select the number of grids and blocks for the CUDA kernels. In Section 3.5.3, we present the strategies to reduce the CPU-GPU communication overhead. In Section 3.5.4, we present the branch refactoring method to optimize the control divergence during the execution of CUDA kernels. In Section 3.5.5, we present the concurrent kernel mechanism to enhance the overall kernel launch throughput.

3.5.1 Device Hierarchical Memory Optimization

In this section, we present the hierarchical memory optimization strategy to effectively map the DARP data structure into various GPU memories.

FIGURE 3.1: Memory mapping on the GPU.

The GPU hardware contains various types of memory resources with different sizes and access latencies. The scheduling heuristic of DARP is complex and consists of control-intensive tasks. Therefore, it is important to efficiently map the data structure onto the GPU to exploit its memory bandwidth. The data elements operated by the scheduling heuristic (Algorithm 2) of the GPU class, are classified based on their access type, access frequency, and size. There are data elements that are needed throughout the program, which are stored in the *global memory*. Although the memory latency is the highest, the *global memory* offers a larger capacity for data storage. Besides, the data elements that are no longer required

after the execution of a CUDA kernel are mapped onto the *register memory*, which is several hundred times faster than the *global memory*. The spilled data from the *register memory* goes into the *local memory*. *Constant memory* of the GPU is designed to cache data with very low read access latency, henceforth the DARP instance, which is static in nature, is stored in this type of memory. The other types of GPU memory such as *texture* and *shared memory* have their unique advantages, but they are not used in our program. Table 3.2 summarizes all the GPU memory types and details how we perform the device hierarchical memory optimization.

Figure 3.1 illustrates the data mapping onto the GPU and shows how each GPU thread is mapped to a CUDA core to simultaneously explore a unique solution in the neighborhood of the current solution.

3.5.2 Adaptive Kernel Grid Configuration

In this section, we present the adaptive kernel grid configuration to heuristically select the number of grids and blocks for the CUDA kernels.

Kernel grid configuration consists of optimally segregating the tasks with the number of threads and blocks, for efficient parallel computation. To maximize GPU exploitation, it is important to choose the right combination. Since a brute force search may take an enormous amount of time, we develop a kernel-grid heuristic based on the concept of warps. N_{size} defines neighborhood size that can be calculated in advance.

Algorithm 8 shows the kernel-grid heuristic, which is explained as follows: 1) Randomly set grid size B_r and block size T_r , and initialize the values of B_{opt} , T_{opt} , t_{best} and $step_{size}$. Here, the value of $step_{size}$ is set as the warp size, which is 32, thereby avoiding the scenario of branch divergence. 2) For all the values of T in $(0, 1024]$ interleaved by the incremental factor $step_{size}$, perform step-3. 3) For each value of T , set the number of blocks B , and launch the kernel functions to record the kernel run time t_{kernel} . Then, update the value of the best number of threads T_{opt} and the best kernel time lapsed t_{kernel} . 4) After a fixed number of iterations, set the grid size as B_{opt} and the block size as T_{opt} . This procedure can attain a good kernel configuration regardless of the GPU architecture.

GPU memory resource				DARP memory mapping			
Memory type	Read access latency	Size		Data mapping	Access type	Access frequency	Size
Global	High	High		Data with program life-time	Read/Write	High	High
Register	Very low	Very low		Data with kernel life-time	Read/Write	Very high	Medium
Local	Mid-high	Medium		Register-spilled data	Read/Write	Very high	Low
Shared	Low	Small		-	-	-	-
Constant	Low (cached)	Medium		DARP Instance	Read only	Medium	Low
Texture	Low (cached)	Medium		-	-	-	-

TABLE 3.2: Device hierarchical memory optimization

Algorithm 8 Adaptive kernel grid configuration

```

Kernel_Func <<< B, T, S, W >>> ()
 $B_r \leftarrow \text{rand}(1, 65535)$ ;  $T_r \leftarrow \text{rand}(1, 1024)$ 
 $B_{opt} \leftarrow (N_{size}/T_{opt}) + 1$ ;  $T_{opt} = 32 \leftarrow \text{rand}(1, 1024)$ 
Initialize  $t_{best} \leftarrow 0$ 
 $step_{size} \leftarrow 32$ 
for  $\forall T \in (0, 1024, step_{size}]$  do
     $B \leftarrow \text{round}(N_{size}/T) + 1$ 
    Launch Kernel_Func <<< B, T, S, W >>> ()
    Record kernel time  $t_{kernel}$ 
    if  $t_{best} \leftarrow 0 \parallel t_{kernel} < t_{best}$  then
         $t_{best} \leftarrow t_{kernel}$ 
         $T_{opt} \leftarrow T$ 
    end if
end for
Set  $T \leftarrow T_{opt}$ ;  $B \leftarrow B_{opt}$ 

```

3.5.3 Optimizing Data Transfer

In this section, we present the strategies to reduce the CPU-GPU communication overhead.

Whenever a program encounters a CPU-GPU copy instruction in CUDA, the control returns to the CPU only after completing the memory transaction, which creates an unnecessary overhead. Therefore, we have designed the following strategies to optimize data transfer between CPU and GPU: 1) *asynchronous data transfer*: dual direct memory access engine of the GPU allows data transfer over the PCIe bus from GPU to CPU, while the CUDA cores are busy crunching the other data. As a result, the program control immediately returns to the next instruction on the CPU side, allowing overlapped execution of both the CPU module and the CPU-GPU data transfer. 2) *batch data transfer*: due to the overhead associated with a single data transfer request, it is preferred to group individual data transfers together to perform contiguous copying. 3) *page-locked data transfer*: host data allocations are pageable by default. After initiating the data transfer, the data elements from pageable memory are moved to pinned memory, before getting transferred to the *global memory*. Therefore, we directly allocate the operating data in the CPU pinned memory.

FIGURE 3.2: Optimized data transfer between CPU and GPU.

3.5.4 Reducing Control Divergence

In this section, we present the branch refactoring method to optimize the control divergence during the execution of CUDA kernels.

In CUDA, each warp comprises 32 threads that execute a single instruction in one cycle. If multiple instructions are associated with different threads from the same warp, then the executions are serialized. Divergent branches cause inefficient use of computational resources; hence it should be avoided by executing the same set of instructions in all the threads associated with a single warp.

Algorithm 9 Insertion Kernel (with control divergence)

```

for  $i = 0; i < limit; i++$  do
  if  $i \leftarrow start$  then
     $gpu[idx].route[i] \leftarrow gpu[idx].start$ 
  else if  $i \leftarrow gap$  then
     $gpu[idx].route[i] \leftarrow gpu[idx].end$ 
  else
     $gpu[idx].route[i] \leftarrow route[i]$ 
  end if
end for

```

Algorithm 9 illustrates the naive version of the *Insertion* kernel. This CUDA kernel is responsible for the operation of inserting pickup and drop-off vertices that belong to a request i into a route k at positions $start$ and gap , respectively, while initializing other nodes along the route k . The naive idea is to directly implement

Algorithm 10 Insertion_Kernel (without control divergence)

```

 $gpu[idx].route[i] \leftarrow gpu[idx].start$ 
 $gpu[idx].route[i] \leftarrow gpu[idx].end$ 
for  $i = 0; i < limit; i++$  do
     $bool\ state\_1 \leftarrow i \geq start$ 
     $bool\ state\_2 \leftarrow i \geq gap$ 
     $gpu[idx].route[i + state\_1 + state\_2] \leftarrow route[i]$ 
end for

```

them using if-else conditions. However, performance penalties may be incurred due to branch divergence. Therefore, we introduce a method to reduce the divergence through Boolean operators and an array indexing scheme as described in Algorithm 10. In this method, the Boolean variables, $state_1$ and $state_2$, keep track of $start$ and gap , respectively. Finally, the index of the route k for each GPU object is calculated based on the summation of the $state$ variables and the current index value i . As a result, the program control does not diverge during the execution of the *Insertion* kernel. The branch refactoring method was first applied for branch and bound algorithm in GPU for flow shop scheduling problem [188]. We extend upon this idea to reduce the control divergence for the application of DARP.

3.5.5 Enhancing Kernel Launch Throughput

In this section, we present the concurrent kernel mechanism to enhance the overall kernel launch throughput.

FIGURE 3.3: Concurrent kernel execution with slicing.

For efficient CPU-GPU heterogeneous computing, it is crucial to always keep the computational cores busy. In the baseline Algorithm 3, all the kernels operate on a default *stream*. A *stream* is a sequence of operations that execute in issue-order on the GPU. The CPU places issue-orders in the *stream* while retaining the program control. If more than one kernel is launched in a stream, the kernel executions are serialized. Figure 3.3 illustrates the proposed method of concurrent kernel execution with kernel slicing. Assume, the kernels in Section 3.4.3 are numbered from 1 to 4. Let $Ka(b)$ indicates the kernel number a placed on stream b . a ranges from 1 to 4 corresponding to the described kernels in Section 3.4.3; b ranges from 1 to n , where n is the total number of requests. The series of kernels $K1(1), \dots, K1(n)$ corresponding to stream $1, \dots, n$ are launched simultaneously. Multiple kernels that use the same stream operate sequentially. Thus, in our implementation, a set of kernels that belongs to the same request is continuously executed regardless of the state of other kernels.

3.6 Integrating the proposed GPU framework with Metaheuristics

In this section, we detail the metaheuristics that are integrated with the GPU framework to solve DARP. First, we provide a brief overview of the GPU-based metaheuristic approach (Section 3.6.1). Afterwards, we describe the neighborhood moves (Section 3.6.2), acceptance criteria to decide the best solution among the neighborhood solutions (Section 3.6.3), allowance of infeasible solutions during the search operation (Section 3.6.4), and implementation details of the metaheuristics (Section 3.6.5) such as tabu search (Section 3.6.5.1), variable neighborhood search (Section 3.6.5.2), and adaptive large neighborhood search (Section 3.6.5.3).

3.6.1 Overview of the GPU-based metaheuristics

In this section, we provide a brief overview of the GPU-based metaheuristic approach. To solve DARP, we iteratively explore the neighborhood solutions of an initial solution until a stopping criterion is met. *DARP instance* will be the input. *Best solution found* is the output, which must be both feasible and should have the least *Cost* or objective function value among all the explored solutions. Firstly, an *Initial solution* for the given DARP instance will be constructed and it will be set as the *Current solution*. *Current solution* is the solution present at the current iteration. Then, the process of iteratively exploring the neighborhood of a *Current solution* is performed using GPU. The ways to generate *neighborhood solutions* from the *Current solution*, and acceptance criterion to select the best admissible solution on each iteration are based on the integrated metaheuristic. During each iteration, the best admissible solution among the *neighborhood solutions* is selected in CPU. For search diversification, the acceptance criteria may allow infeasible solutions to be selected. However, the *Best solution found* must be always feasible and updated after every iteration.

3.6.2 Neighborhood move

In this section, we provide a description of the neighborhood move. The process of moving a customer request from one vehicle into another is called as *neighborhood*

move. All possible solutions resulting from moving a request into all available vehicles are called as *neighborhood solutions*. For GPU computation, the neighborhood solutions are divided into batches that are executed concurrently in GPU. Each batch simultaneously evaluates the process of moving a unique request into all vehicles. Inside each batch, the GPU threads simultaneously evaluate the process of moving a unique request into a unique vehicle. The choice of which customer requests need to be relocated is chosen by the integrated metaheuristic.

3.6.3 Acceptance criteria to iteratively select the best solution

In this section, we describe the acceptance criteria to iteratively select the best admissible solution during the optimization process. There are two strategies for iteratively accepting the best solution among the neighborhood namely, first-improvement and best-improvement. In the first-improvement strategy, the neighborhood solutions are evaluated one-by-one, once a better solution is found, the control will move to the next iteration. In the best-improvement strategy, all the neighborhood solutions need to be evaluated and the best one will be found before proceeding to the next iteration. We have decided to use the best-improvement strategy based on the following reasons: 1) complete neighborhood exploration has the maximum potential to find the best solution during the search [194], and 2) since we use GPU that has thousands of cores, it is faster to evaluate all the neighborhood solutions simultaneously.

3.6.4 Allowance of infeasible solutions

In this section, we describe the process to allow infeasible solutions during the optimization process. To escape local optima, infeasible solutions are allowed during the search process through a relaxation mechanism [7]. Solutions are evaluated by a modified objective function $f_{new}(x) = f(x) + \alpha' * q(x) + \beta' * d(x) + \gamma' * w(x) + \phi' * t(x)$, where $\alpha', \beta', \gamma', \phi'$ are self-adjusting positive parameters. During every iteration, these parameters are dynamically increased and decreased by a factor of $(1 + \delta')$, if the current solution is feasible and infeasible, respectively. A DARP solution is feasible if it does not violate any given constraints in the formulation. We use

the same parameters and relaxation mechanism described in [7] to facilitate the exploration of solution space.

3.6.5 Implementation details

In this section, we describe the implementation details of the proposed GPU framework with three metaheuristic approaches: tabu search (TS), variable neighborhood search (VNS), and adaptive large neighborhood search (ALNS). After integrating the proposed GPU framework with TS, VNS, and ALNS, we describe the resultant methods as GPU-based tabu search (G-TS) (Section 3.6.5.1), GPU-based variable neighborhood search (G-VNS) (Section 3.6.5.2), and GPU-based adaptive large neighborhood search (G-ALNS) (Section 3.6.5.3), respectively, for efficiently solving DARP.

3.6.5.1 GPU-based Tabu Search Heuristic

In this section, we describe the implementation details of the GPU-accelerated tabu search (G-TS) algorithm for DARP.

Tabu search heuristic was first introduced in [162] and applied to solve DARP in [7]. It works based on the concept of forbidding recently visited neighborhood moves for a limited number of iterations to avoid getting trapped at local optimal solutions. It is often regarded as a higher-level procedure that guides lower-level heuristics. In our implementation, the lower-level heuristic is the GPU-accelerated local search method. Therefore, we extend the tabu search of [7] with the proposed GPU framework to form G-TS. We have reported the preliminary results in [6].

In tabu search, all requests are removed and reinserted iteratively. A request that is removed from a route must be reinserted in a different route. During the search, we allow infeasible solutions as described by [7]. In TS, recently performed neighborhood moves will be updated in an array consisting of *tabu moves*. Such moves cannot be repeated for a subsequent number of iterations. During the process of selecting the best admissible solution in a neighborhood, *tabu moves* are not considered. If none of the neighborhood solutions is feasible, then the best infeasible solution is selected as the solution that has the least value of the objective function.

Algorithm 11 Algorithm: GPU-based TS

Require: DARP Instance**Ensure:** *Best solution found* and $Cost_{best}$

- 1: Parameterization and allocation of GPU resources.
 - 2: Copy DARP instance from CPU to GPU.
 - 3: Construct an *Initial solution* in CPU
 - 4: Set *Current solution* as *Initial solution* in CPU
 - 5: **repeat**
 - 6: Update *req_list* based on TS in CPU
 - 7: Copy *Current solution* from CPU to GPU
 - 8: // *Evaluating neighborhoods in multiple concurrent batches*
 - 9: **Explore neighborhood in GPU** i.e. launch *GPU-accelerated Local Search()*
 - 10: Selectively copy data from GPU to CPU
 - 11: Update the overall cost of each neighborhood solution in CPU
 - 12: Select best solution among the neighborhood solutions based on TS in CPU
 - 13: Set the newly selected solution as *Current solution* in CPU
 - 14: Update $Cost_{best}$ of the *Best solution found* in CPU
 - 15: Adjust *tabu coefficients* (relaxation mechanism)
 - 16: **until** stopping criterion
 - 17: Release GPU resources.
 - 18: **return** *Best solution found*
-

However, if any neighborhood solution is better than the *Best solution found*, then the *tabu move* can be overridden.

Algorithm 11 describes the implementation of our GPU-based tabu search algorithm to solve the DARP. The procedure can be described as follows: 1) allocate device resources, and initialize the *tabu coefficients* (relaxation mechanism) based on [7], 2) construct initial solution (as per Section 3.7), 3) copy *Current solution* from CPU to GPU, 4) construct and evaluate neighborhood solutions in GPU, 5) selectively copy data from GPU to CPU, 6) identify the best admissible solution and set it as the *Current solution* for next iteration, 7) repeat steps 3-6 until the stopping criterion is fulfilled, and 9) output the *Best solution found* and release device resources.

3.6.5.2 GPU-based Variable Neighborhood Search

In this section, we describe the implementation details of the GPU-accelerated variable neighborhood search (G-VNS) algorithm for DARP.

In general, VNS starts from an initial solution. Then, during every iteration, a solution x' is selected from a set of neighborhood solutions of a *Current solution* x by shaking procedure. A subsequent local search step may be applied to the selected solution x' to yield x'' . If x'' is better than x , it replaces x and the search continues with the first neighborhood $k = 1$. If x'' is worse, x is not replaced and the next (larger) neighborhood is used in the subsequent iteration $k = k + 1$. Whenever k reaches the maximum number of neighborhoods k_{max} , the search continues with the first neighborhood $k = 1$. This is repeated until the stopping criterion is met.

Previously, VNS was applied to DARP in [9] which has complicated shaking procedures and design decisions. Instead, we employ the two simpler shaking procedures from the literature namely, 1) ejection neighborhood (random and worst) [77], and 2) zero-split neighborhood [9]. During every iteration, we randomly choose the type of shaking procedure. In the ejection neighborhood, a set of requests is chosen, removed from the current solution, and reinserted at the best possible position among all vehicles. The requests can be removed in two ways: i) random removal of requests, and ii) those requests that most degrade the value of the objective function can be removed. The number of neighborhoods, i.e., requests to be relocated k varies from 1 to 7. In a zero-split neighborhood, a sequence of requests is moved from one vehicle to another. A request that is removed from its current route can be reinserted into any route. Sometimes, sequences of requests lie between two arcs where the vehicle load is equal to zero, i.e., the vehicle is empty. Every vehicle may contain such multiple sequences of requests. In this neighborhood, a randomly chosen such sequence of requests will be removed from its current vehicle. All requests that are part of this sequence are then reinserted one-by-one at the best position into the vehicles. A thresholding criterion is employed to accept the solution resulting from the shaking process. The local search step is carried out only when a new value of the *Best solution found* is found in that iteration. During the local search phase, each request is removed from its corresponding route and reinserted at the best possible positions of the same route. Additionally, we employ a restart mechanism, in which, if the *Current solution* does not improve for more than 200 iterations, we set the *Current solution* as the *Best solution found* and restart the search for intensification. We allow the infeasible solutions to be accepted during the optimization process for diversification.

Algorithm 12 describes the implementation of our GPU-based VNS method for

Algorithm 12 Algorithm: GPU-based VNS

Require: DARP Instance**Ensure:** *Best solution found* and $Cost_{best}$

- 1: Parameterization and allocation of GPU resources.
 - 2: Copy DARP instance from CPU to GPU.
 - 3: Construct an *Initial solution* in CPU
 - 4: Set *Current solution* as *Initial solution* in CPU
 - 5: **repeat**
 - 6: Update *req_list* based on VNS in CPU
 - 7: Copy *Current solution* from CPU to GPU
 - 8: // *Evaluating neighborhoods in a single batch*
 - 9: **Explore neighborhood in GPU** i.e. launch *GPU-accelerated Local Search()*
 - 10: Selectively copy data from GPU to CPU
 - 11: Update the overall cost of each neighborhood solution in CPU
 - 12: Select best solution among the neighborhood solutions based on VNS in CPU
 - 13: Set the newly selected solution as *Current solution* in CPU
 - 14: Update $Cost_{best}$ of the *Best solution found* in CPU
 - 15: **if** No improvement in $Cost_{current}$ for a predefined number of iterations **then**
 - 16: Set the *Current solution* as the *Best solution found*
 - 17: **end if**
 - 18: **until** stopping criterion
 - 19: Release GPU resources.
 - 20: **return** *Best solution found*
-

DARP. The procedure can be described as follows: 1) allocate device resources, 2) construct initial solution (as per Section 3.7), 3) copy *Current solution* from CPU to GPU, 4) construct and evaluate neighborhood solutions in GPU, 5) selectively copy data from GPU to CPU, and select a best admissible solution in CPU, 6) trigger restart mechanism in CPU if required, 7) record the cost of the best solution $Cost_{best}$ in CPU, 8) repeat steps 3-7 until the stopping criterion, 9) print the *Best solution found*, and release the memory resources.

3.6.5.3 GPU-based Adaptive Large Neighborhood Search

In this section, we describe the implementation details of the GPU-accelerated adaptive large neighborhood search (G-ALNS) algorithm for DARP.

The general idea of LNS by Shaw (1998) [195] is to repeatedly remove requests from the solution and to reinsert them at more profitable positions. Later, an adaptive

weight adjustment scheme to LNS was introduced by Ropke et al. (2006) [164] to develop ALNS for vehicle routing problems with time-windows. In this section, we extend this ALNS approach with GPU-acceleration to solve disruptive DARP-VB. The idea of exploring neighborhoods using GPU for DARP was introduced in our preliminary conference paper in Ramesh et al. (2018a) [6], which was applied for a Tabu Search algorithm. In this section, we develop a GPU-accelerated Adaptive Large Neighborhood Search (G-ALNS) algorithm to solve the disruptive DARP-VB.

Algorithm 13 Algorithm: GPU-based ALNS

Require: DARP Instance

Ensure: *Best solution found*

```

1: CurrentSolution  $\leftarrow$  InitialSolution
2: while the termination condition is not satisfied do
3:   Select ALNS operators according to their past scores
4:   Apply Destroy operator to the CurrentSolution
5:   DataTransfer from CPU to GPU
6:   // Evaluating neighborhoods in a single batch
7:   Apply Repair operator i.e., Explore neighborhood in GPU /launch
   GPU-accelerated Local Search()
8:   DataTransfer from GPU to CPU
9:   Select best solution among the neighborhood solutions based on SA-
   acceptance criterion of ALNS in CPU
10:  Set the newly selected solution as Current solution in CPU
11:  Update  $Cost_{best}$  of the Best solution found in CPU
12: end while
13: return BestSolution

```

Algorithm 3 details the neighborhood exploration in GPU. Algorithm 13 details the proposed G-ALNS framework. The algorithm starts with an initial solution. For each iteration, a pair of destroy and repair ALNS operators are adaptively selected from a pool of operators using a roulette wheel mechanism [164]. First, a destroy operator is applied to remove some requests from the current solution. Then, a repair operator explores the neighborhood using CUDA kernels (Section 3.4.3) in GPU and selects an appropriate solution in CPU based on simulated annealing acceptance criterion. The weights of ALNS operators are adjusted after every segment according to predefined rules. For a detailed description of the ALNS heuristic, the readers are referred to the work of Ropke et al. 2006 [164].

We implemented the following neighborhood search operators (destroy and repair operators) for ALNS.

- Destroy operators
 - *Worst removal*: This operator iteratively removes requests that cause the biggest detour in the current solution.
 - *Random removal*: This operator randomly selects the requests to be removed among the planned requests.
 - *Related removal (or Shaw removal)*: This operator aims to remove requests that are alike, so that their positions of service can somehow be interchanged.
- Repair operators
 - *Best insertion*: At each iteration, the best insertion cost is computed for each unplanned request and the request with the lowest insertion cost is inserted at its best position. The heuristic stops when all requests are routed or none can be inserted anymore.
 - *Regret insertion*: This heuristic is based on the notion of regret, which is used by Potvin [196] for the Vehicle Routing Problem with Time Windows (VRPTW), and extended by Ropke [164]. It investigates the cost of inserting a request on the k best routes, and finally inserts it on a route whose cost difference between inserting it into the best route and the $k-1$ best routes is largest. In our implementation, we consider regret- k heuristics with values of k between 1 and 5.

In addition to the above-mentioned operators, we perform intra-route optimization whenever a new best solution is found. In intra-route optimization, we remove and re-insert all requests in the best position on their respective routes. Furthermore, we invoke tabu search for every 200 iterations with a stopping criterion of 100 iterations. We summarize the implementation of our GPU-based ALNS method for DARP in Algorithm 12. The procedure can be described as follows: 1) allocate device resources, 2) construct initial solution (as per Section 3.7), 3) copy *Current solution* from CPU to GPU, 4) construct and evaluate neighborhood solutions in GPU, 5) selectively copy data from GPU to CPU, and select a best admissible

solution in CPU, 6) trigger restart mechanism in CPU if required, 7) record the cost of the best solution $Cost_{best}$ in CPU, 8) repeat steps 3-7 until the stopping criterion, 9) print the *Best solution found*, and release the memory resources.

To benchmark the efficiency of our GPU-based ALNS for the DARP that require quick solutions, it is important to re-implement the same ALNS in CPU since Masson (2014) [77] did not report the following: i) solutions at various time instances < 30 sec, ii) algorithmic parameters, iii) CPU type, iv) style of hash table implementation and v) the notion of the critical vertex. In our approach, we use a separate chaining approach with hash function $f(x) \bmod 199$ to avoid hash collisions, and the notion of the critical vertex is not used since it has the maximum potential to find the best solution during the search operation.

In the next section, we evaluate the performance of the proposed GPU-based solution methodology and compare the results with the existing approaches for solving DARP.

3.7 Simulation Results

In this section, we evaluate the performance of the proposed GPU-based methodology on DARP benchmark instances and compare the results with the conventional CPU approaches, conventional GPU-based approach, and existing best-performing sequential approaches. In Section 3.7.1, we describe the simulation settings and the details of the DARP instances. In Section 3.7.2, we analyze the effect of various components (presented in Section 3.5) on computational speedup for the proposed GPU methodology. In Section 3.7.3, we present the solution quality attained by the proposed GPU-based methodology and compare the results with existing best-performing approaches for solving DARP.

3.7.1 Simulation settings

In this section, we describe the program development environment, DARP benchmark instances, and construction heuristic to generate an initial DARP solution.

The GPU-accelerated algorithms are implemented in CUDA C and conventional CPU-based algorithms are implemented in C++ language. All simulations have been carried out on a computer running 2.1 GHz Intel Xeon E5-2620 v4 processor and NVidia Tesla P100-PCI-E-16GB GPU, interconnected with a PCI-e controller. We have used Visual studio 2015 with CUDA 8.0 toolkit in Windows 8.1 as the integrated development environment.

The DARP test instances are obtained from [7]. In these instances, the number of requests ranges from 24 to 144, and the number of vehicles ranges from 3 to 13. Each request is randomly generated in the Euclidean space of $[-10, 10]^2$. For R1a-R10a, the time window length is 15 min. For R1b-R10b, the time window length is 45 min. Each instance has a different geographical distribution of pick-up and drop-off locations. The service time for each vehicle at any location is set as 10 min. The capacity of each vehicle is 6, with a maximum route duration limit of 480 min. The maximum passenger ride time limit is 90 min. All vehicles are considered to travel at a constant speed in the road network. All vehicles start from the depot and return to the depot after serving the last assigned request. These DARP instances (consisting of the simulation network and properties of the problem) are well-studied in the literature and publicly available online. As referenced by original authors of these instances [7], we describe that the instances are generated and calibrated according to realistic assumptions and in alignment with several North American transporters [28], the information regarding time window widths, vehicle capacity, route duration, and maximum ride time was provided by the Montreal Transit Commission (MTC) of Canada, the vehicle speed in the road network is considered to be constant, i.e., traffic congestion is not taken into account, and the transport network is considered as an undirected complete graph within the Euclidean space $[-10, 10]^2$ km.

To ensure a fair comparison between CPU and GPU methods, all algorithms start with an initial solution generated by the same construction heuristic, described as follows: i) create m set of empty routes, ii) for each route $k \in m$, randomly initialize the maximum number of pick up and drop-off vertices that can be inserted, iii) create a *req_list* that comprises n requests in a random order, iv) for each unserved request $i \in req_list$, randomly insert its pick up and drop-off vertices while ensuring the pair-precedence constraints.

Notation	Description
TS	Described tabu search heuristic methodology
VNS	Described variable neighborhood search methodology
ALNS	Described adaptive large neighborhood search methodology
C_1	TS with conventional GPU methodology
C_1+opt_1	TS with conventional GPU methodology with device hierarchical memory optimization in section 3.5.1
C_1+opt_2	TS with conventional GPU methodology with adaptive kernel grid configuration in section 3.5.2
C_1+opt_3	TS with conventional GPU methodology with optimizing data transfer in section 3.5.3
C_1+opt_4	TS with conventional GPU methodology with optimizing thread divergence in section 3.5.4
C_1+opt_5	TS with conventional GPU methodology with concurrent kernels in section 3.5.5
P_1	TS with proposed GPU methodology (described GPU framework + all optimization strategies in section 3.5)
P_1-opt_1	TS with proposed GPU methodology without device hierarchical memory optimization in section 3.5.1
P_1-opt_2	TS with proposed GPU methodology without adaptive kernel grid configuration in section 3.5.2
P_1-opt_3	TS with proposed GPU methodology without optimizing data transfer in section 3.5.3
P_1-opt_4	TS with proposed GPU methodology without optimizing thread divergence in section 3.5.4
P_1-opt_5	TS with proposed GPU methodology without concurrent kernels in section 3.5.5
C_2	VNS with conventional GPU methodology
C_2+opt_1	VNS with conventional GPU methodology with device hierarchical memory optimization in section 3.5.1
C_2+opt_2	VNS with conventional GPU methodology with adaptive kernel grid configuration in section 3.5.2
C_2+opt_3	VNS with conventional GPU methodology with optimizing data transfer in section 3.5.3
C_2+opt_4	VNS with conventional GPU methodology with optimizing thread divergence in section 3.5.4
C_2+opt_5	VNS with conventional GPU methodology with concurrent kernels in section 3.5.5
P_2	VNS with proposed GPU methodology (described GPU framework + all optimization strategies in section 3.5)
P_2-opt_1	VNS with proposed GPU methodology without device hierarchical memory optimization in section 3.5.1
P_2-opt_2	VNS with proposed GPU methodology without adaptive kernel grid configuration in section 3.5.2
P_2-opt_3	VNS with proposed GPU methodology without optimizing data transfer in section 3.5.3
P_2-opt_4	VNS with proposed GPU methodology without optimizing thread divergence in section 3.5.4
P_2-opt_5	VNS with proposed GPU methodology without concurrent kernels in section 3.5.5
C_3	ALNS with conventional GPU methodology
C_3+opt_1	ALNS with conventional GPU methodology with device hierarchical memory optimization in section 3.5.1
C_3+opt_2	ALNS with conventional GPU methodology with adaptive kernel grid configuration in section 3.5.2
C_3+opt_3	ALNS with conventional GPU methodology with optimizing data transfer in section 3.5.3
C_3+opt_4	ALNS with conventional GPU methodology with optimizing thread divergence in section 3.5.4
C_3+opt_5	ALNS with conventional GPU methodology with concurrent kernels in section 3.5.5
P_3	ALNS with proposed GPU methodology (described GPU framework + all optimization strategies in section 3.5)
P_3-opt_1	ALNS with proposed GPU methodology without device hierarchical memory optimization in section 3.5.1
P_3-opt_2	ALNS with proposed GPU methodology without adaptive kernel grid configuration in section 3.5.2
P_3-opt_3	ALNS with proposed GPU methodology without optimizing data transfer in section 3.5.3
P_3-opt_4	ALNS with proposed GPU methodology without optimizing thread divergence in section 3.5.4
P_3-opt_5	ALNS with proposed GPU methodology without concurrent kernels in section 3.5.5
E_1	Existing results based on VNS reported by Parragh et al. [9]
E_2	Existing results based on TS of Cordeau and Laporte (2003) [7] reported by Kirchler et al. [67]
E_3	Existing results based on Granular TS variant, GTS_P reported by Kirchler et al. [67]
E_4	Existing results based on Granular TS variant, GTS_RC reported by Kirchler et al. [67]

TABLE 3.3: Representation of the methodologies.

3.7.2 Effect of different components on computational speedup for the proposed GPU methodology

In this section, we analyze the effect of various components (presented in Section 3.5) on computational speedup for the proposed GPU methodology.

Table 3.3 describes the notations of the various solution methodologies. The proposed GPU algorithms are represented by $P \in P_1, P_2, P_3$ series, conventional GPU methodologies are denoted by $C \in C_1, C_2, C_3$ series, and existing best-performing

sequential methods are mentioned by $E \in E_1, E_2, E_3, E_4$ series. The device optimization strategies are represented by $opt_1, opt_2, opt_3, opt_4, opt_5$, where opt_1 denotes device hierarchical memory optimization, opt_2 denotes adaptive kernel grid configuration, opt_3 denotes data transfer optimization, opt_4 denotes reducing thread divergence, opt_5 denotes concurrent kernel mechanism. For example, $P - opt_1$ represents the proposed GPU methodology without optimization strategy opt_1 . In another example, $C + opt_1$ represents the conventional GPU methodology with the optimization strategy opt_1 . The conventional GPU methodology with all the five device optimization strategies is equivalent to the proposed GPU methodology. $C_1 + opt_1, opt_2, opt_3, opt_4, opt_5$ is equivalent to P_1 . $C_2 + opt_1, opt_2, opt_3, opt_4, opt_5$ is equivalent to P_2 . $C_3 + opt_1, opt_2, opt_3, opt_4, opt_5$ is equivalent to P_3 .

In terms of computational speedup, the effect of including each device optimization strategy into the conventional GPU methodology, and the effect of excluding each device optimization strategy from the proposed GPU methodology are shown in Tables 3.4, 3.5, and 3.6. Initially, the computational time for the GPU-based methods and their CPU counterparts on DARP instances are recorded after 10^2 number of iterations. Then, the computational speedup is calculated as follows.

$$speedup = \frac{\text{Execution time of CPU only method}}{\text{Execution time of GPU-based method}}. \quad (3.19)$$

Each experiment is repeated five times to calculate the average speedup. The fundamental algorithmic behavior of the GPU-based methods and their CPU counterparts remains the same except for the computational speed. The existing methods $E \in E_1, E_2, E_3, E_4$ provide results for 13 test instances. For the remaining 7 instances, detailed information about the solution at different computation times are not available. Therefore, we also restrict our tests to those 13 test instances.

Tables 3.7, 3.8, and 3.9 show the percentage difference from the best computational speedup attained for the GPU-based solution methodologies when compared to their respective CPU counterparts.

With the conventional GPU methodologies C , the inclusion of strategies opt_1, opt_2, opt_5 leads to the largest increase in speedup, and the inclusion of strategy opt_4 leads to the lowest increase in speedup. With the proposed GPU methodologies P , the exclusion of strategy opt_3 leads to the largest decrease in speedup, and exclusion

of strategy opt_4 leads to the smallest decrease in speedup. Therefore, when designing a GPU-based algorithm for DARP, our results suggest the following: 1) it is preferred to first include the strategies opt_1, opt_2, opt_5 followed by opt_3, opt_4 , 2) it is most advisable to remove the strategy opt_4 and least preferable to remove strategy opt_3 from the GPU methodology during the process of algorithmic simplifications. Furthermore, it can be observed that the speedup is the largest when all device optimization strategies ($opt_1, opt_2, opt_3, opt_4, opt_5$) are included in the GPU methodology. As a result, the GPU-based tabu search algorithm P_1 , GPU-based variable neighborhood search P_2 , and GPU-based adaptive large neighborhood search P_3 exhibit the highest computational speedup. Therefore, we have considered only these three methods for subsequent experiments.

3.7.3 Comparison of solution quality of our approach with the existing results

In this section, we present the solution quality attained by the proposed GPU-based methodology and compare the results with existing best-performing approaches for solving DARP.

Tables 3.10, 3.11, and 3.12 compare the solution quality of the proposed GPU approaches $P \in P_1, P_2, P_3$, in terms of gap defined in Eq. 3.20, with the best-performing sequential approaches $E \in E_1, E_2, E_3, E_4$ from the literature. For each instance, two of the best values of gap are highlighted in bold text. Each experiment is conducted five times, and average values of gap are reported. Since all the existing best-performing algorithms E report "mean" results rather than median, we follow the same trend and report mean/average results to ensure a fair comparison with existing results. For uniformity, the objective function values reported in E are converted into gap values in the Tables 3.10, 3.11, and 3.12. The definition for gap is given by:

$$gap (\%) = \frac{Cost - BKS}{BKS} \times 100, \quad (3.20)$$

where BKS denotes the best-known solutions or objective function value, and $Cost$ denotes the best value of objective function attained for a given solution method. The solution quality is measured by the value of gap . Lesser the value of gap represents a better solution quality. A negative value of gap denotes a quality of

Case	n	m	GPU-based Tabu Search Heuristic											
			C ₁	C ₁ +opt ₁	C ₁ +opt ₂	C ₁ +opt ₃	C ₁ +opt ₄	C ₁ +opt ₅	P ₁ -opt ₁	P ₁ -opt ₂	P ₁ -opt ₃	P ₁ -opt ₄	P ₁ -opt ₅	P ₁
R1a	24	3	2.89	3.62	4.04	3.52	2.99	11.68	20.63	19.11	4.43	25.71	5.75	26.62
R2a	48	5	4.95	6.95	9.44	6.03	5.29	21.67	34.95	35.51	10.85	45.29	13.48	51.44
R3a	72	7	7.14	11.37	13.70	8.30	7.62	26.25	33.78	48.73	17.20	61.50	21.16	67.66
R4a	96	9	12.85	16.34	22.26	15.21	17.45	50.14	67.35	67.80	56.40	68.21	25.86	69.81
R5a	120	11	11.87	18.65	18.59	13.07	12.82	28.71	29.87	57.94	26.06	63.11	29.78	68.22
R6a	144	13	25.21	26.80	29.11	26.90	25.93	45.69	55.56	52.73	53.39	60.11	33.74	60.69
R7a	36	4	5.36	4.98	7.06	6.85	5.70	18.66	34.48	36.34	23.71	36.64	8.55	37.81
R8a	72	6	14.55	17.20	20.59	17.89	17.20	48.57	72.72	64.56	60.13	71.80	25.34	73.61
R9a	108	8	12.32	19.19	19.77	14.04	13.14	24.15	26.62	58.58	29.26	65.70	33.73	65.97
R10a	144	10	16.98	25.39	23.46	18.81	17.51	23.71	25.19	51.82	34.65	53.44	39.84	55.55
R1b	24	3	2.29	2.92	3.34	2.83	2.46	9.10	17.76	13.93	3.66	19.60	4.69	21.07
R2b	48	5	4.71	6.57	9.00	5.66	5.18	19.85	29.39	33.40	10.20	43.97	12.41	47.37
R3b	72	7	10.48	12.48	17.48	16.01	12.28	44.25	66.44	54.86	51.96	62.66	20.55	68.85
R4b	96	9	19.65	20.40	27.71	21.93	22.14	60.13	81.86	74.96	70.24	84.41	31.62	84.50
R5b	120	11	12.32	18.24	19.79	14.06	13.26	27.47	30.20	62.60	27.22	65.52	31.13	66.68
R6b	144	13	14.91	21.85	22.31	16.18	15.25	27.19	30.32	61.09	31.13	66.24	34.87	66.63
R7b	36	4	3.48	5.01	6.20	4.11	3.74	14.83	25.43	22.83	7.02	34.04	8.78	35.56
R8b	72	6	15.81	20.99	23.11	19.88	17.83	57.01	79.72	79.70	65.52	75.73	27.32	80.24
R9b	108	8	15.96	24.07	25.63	17.60	17.17	33.70	36.45	81.00	37.12	90.05	42.09	92.63
R10b	144	10	16.53	25.13	23.43	18.93	17.53	24.14	25.28	52.69	34.77	53.48	39.68	56.21
Avg.	86	8	11.51	15.41	17.30	13.39	12.63	30.84	41.20	51.48	32.75	57.36	24.51	59.86

TABLE 3.4: Effect of different components on computational speed up for the G-TS methodology.

Case	n	m	GPU-based Variable Neighborhood Search													
			C_2	C_2+opt_1	C_2+opt_2	C_2+opt_3	C_2+opt_4	C_2+opt_5	P_2-opt_1	P_2-opt_2	P_2-opt_3	P_2-opt_4	P_2-opt_5	P_2	P_2	P_2
R1a	24	3	3.97	5.06	5.35	4.49	4.34	4.72	7.54	8.55	6.69	7.99	8.09	10.19		
R2a	48	5	4.69	7.31	6.46	5.68	5.75	5.88	6.41	6.74	6.88	7.56	7.35	12.06		
R3a	72	7	8.24	9.77	10.57	10.32	8.88	13.06	8.96	10.90	9.03	11.48	11.59	15.33		
R4a	96	9	28.60	30.20	34.34	29.80	28.34	29.90	48.59	30.39	36.07	49.19	41.66	53.99		
R5a	120	11	11.34	13.40	14.01	14.55	15.22	14.40	7.01	9.04	7.71	7.26	10.28	17.76		
R6a	144	13	29.70	33.00	37.90	40.17	32.36	35.76	51.97	43.15	39.09	51.51	43.36	52.57		
R7a	36	4	5.44	5.32	7.53	7.29	5.91	7.22	11.38	9.06	9.65	11.99	11.00	12.78		
R8a	72	6	15.47	16.26	23.55	20.96	17.14	19.68	32.39	25.39	27.84	29.32	28.44	34.79		
R9a	108	8	21.58	29.45	30.24	24.51	22.32	24.44	36.56	41.31	41.26	44.74	41.88	48.12		
R10a	144	10	27.65	33.18	32.89	39.46	28.44	29.85	18.87	18.50	10.51	25.88	24.36	50.81		
R1b	24	3	4.53	6.41	6.46	5.74	4.93	5.98	8.57	9.38	8.95	10.29	10.79	11.95		
R2b	48	5	5.62	6.75	7.94	6.80	6.29	7.02	8.03	10.20	8.44	11.29	10.23	15.03		
R3b	72	7	17.16	19.40	25.65	22.36	20.85	21.23	34.36	23.44	27.86	34.08	31.34	37.74		
R4b	96	9	21.76	25.34	34.00	31.02	28.36	33.13	36.88	30.44	39.41	45.50	42.18	47.79		
R5b	120	11	6.93	11.04	9.77	7.55	9.35	9.60	6.19	7.87	6.30	8.96	7.22	15.33		
R6b	144	13	8.40	10.11	11.66	10.00	11.19	13.04	5.15	5.71	6.05	7.18	5.34	16.92		
R7b	36	4	5.67	7.46	9.04	6.87	6.70	6.96	10.02	11.86	9.44	11.99	12.46	16.42		
R8b	72	6	24.68	27.78	33.79	27.38	24.88	33.94	41.60	28.32	37.30	44.96	40.48	47.10		
R9b	108	8	9.50	12.44	12.09	12.97	12.79	13.30	6.12	7.39	7.05	8.37	9.10	15.58		
R10b	144	10	28.95	34.53	32.66	31.06	29.13	30.88	40.36	22.12	21.08	35.05	21.59	52.73		
Avg.	86	8	14.49	17.21	19.30	17.95	16.16	18.01	21.35	17.99	18.33	23.23	20.94	29.25		

TABLE 3.5: Effect of different components on computational speed up for the G-VNS methodology.

Case	n	m	GPU-based Adaptive Large Neighborhood Search											
			C ₃	C ₃ +opt ₁	C ₃ +opt ₂	C ₃ +opt ₃	C ₃ +opt ₄	C ₃ +opt ₅	P ₃ -opt ₁	P ₃ -opt ₂	P ₃ -opt ₃	P ₃ -opt ₄	P ₃ -opt ₅	P ₃
R1a	24	3	6.09	7.53	9.25	7.04	7.22	7.75	9.20	8.34	8.46	8.59	7.63	11.70
R2a	48	5	10.27	10.55	14.02	11.16	11.60	10.91	12.18	10.50	11.26	11.65	10.66	16.76
R3a	72	7	13.97	14.73	22.57	14.12	17.87	20.04	20.84	15.17	20.43	19.92	17.53	28.35
R4a	96	9	24.46	25.37	32.82	27.34	28.13	29.34	32.69	26.27	30.20	30.39	27.86	41.90
R5a	120	11	26.03	27.21	35.90	28.51	31.68	32.45	37.72	35.27	34.00	35.63	30.83	46.31
R6a	144	13	26.29	29.57	35.74	30.67	33.08	32.51	38.96	35.94	36.59	39.53	33.85	48.19
R7a	36	4	7.55	8.14	12.66	8.15	9.91	9.78	10.94	9.26	10.66	10.85	9.93	15.68
R8a	72	6	14.95	15.18	27.80	18.49	18.27	17.92	24.51	19.74	24.50	23.17	21.51	33.05
R9a	108	8	26.11	30.54	38.87	30.76	30.53	30.88	38.84	37.94	39.12	38.94	33.40	50.51
R10a	144	10	34.87	41.98	49.87	39.39	42.67	43.12	52.94	49.53	51.14	51.39	45.71	63.74
R1b	24	3	5.37	6.04	7.58	6.74	6.22	6.42	7.74	6.95	7.90	8.12	6.96	10.32
R2b	48	5	10.83	11.64	17.21	11.86	12.05	12.10	14.55	11.32	14.02	13.72	13.00	20.23
R3b	72	7	18.20	20.16	27.63	19.31	20.34	20.92	25.74	18.66	24.12	25.33	22.09	35.44
R4b	96	9	20.33	21.67	29.94	22.91	24.98	27.05	28.73	24.96	28.13	30.04	25.58	40.08
R5b	120	11	26.04	29.71	37.27	28.59	30.91	31.78	37.87	32.38	34.96	35.53	31.10	47.22
R6b	144	13	37.63	40.80	46.38	41.30	38.60	42.09	49.75	48.74	43.79	47.49	41.25	61.03
R7b	36	4	5.14	6.27	9.04	6.34	6.74	6.33	8.06	5.90	7.71	7.60	6.83	10.87
R8b	72	6	17.23	21.68	31.41	20.28	22.09	22.19	26.97	22.04	26.66	25.93	24.13	37.06
R9b	108	8	23.63	28.42	35.77	30.58	27.80	29.28	34.80	25.95	33.01	33.50	30.05	43.47
R10b	144	10	32.41	38.91	43.86	40.32	37.97	42.28	45.63	44.72	44.23	43.07	40.90	58.74
Avg.	86	8	19.37	21.80	28.28	22.19	22.93	23.76	27.93	24.48	26.54	27.02	24.04	36.03

TABLE 3.6: Effect of different components on computational speed up for the G-ALNS methodology.

Case	n	m	TS											
			P_1	P_1-opt_5	P_1-opt_4	P_1-opt_3	P_1-opt_2	P_1-opt_1	C_1+opt_5	C_1+opt_4	C_1+opt_3	C_1+opt_2	C_1+opt_1	C_1
R1a	24	3	0	78	03	83	28	22	56	89	87	85	86	89
R2a	48	5	0	74	12	79	31	32	58	90	88	82	86	90
R3a	72	7	0	69	09	75	28	50	61	89	88	80	83	89
R4a	96	9	0	63	02	19	03	04	28	75	78	68	77	82
R5a	120	11	0	56	07	62	15	56	58	81	81	73	73	83
R6a	144	13	0	44	01	12	13	08	25	57	56	52	56	58
R7a	36	4	0	77	03	37	04	09	51	85	82	81	87	86
R8a	72	6	0	66	02	18	12	01	34	77	76	72	77	80
R9a	108	8	0	49	00	56	11	60	63	80	79	70	71	81
R10a	144	10	0	28	04	38	07	55	57	68	66	58	54	69
R1b	24	3	0	78	07	83	34	16	57	88	87	84	86	89
R2b	48	5	0	74	07	78	31	38	58	89	88	81	86	90
R3b	72	7	0	70	09	25	20	04	36	82	77	75	82	85
R4b	96	9	0	63	00	17	11	03	29	74	74	67	76	77
R5b	120	11	0	53	02	59	06	55	59	80	79	70	73	82
R6b	144	13	0	48	01	53	8	54	59	77	76	67	67	78
R7b	36	4	0	75	04	80	36	28	58	89	88	83	86	90
R8b	72	6	0	66	06	18	01	01	29	78	75	71	74	80
R9b	108	8	0	55	03	60	13	61	64	81	81	72	74	83
R10b	144	10	0	29	05	38	06	55	57	69	66	58	55	71
Avg.				61	4	50	16	31	50	80	79	72	75	82

TABLE 3.7: Percentage deviation from the best speed up for the G-TS methodology.

solution better than *BKS*. We attempt to minimize the value of *gap*. To calculate the average *gap*, we calculate the mean value of *gap* attained for five simulation runs. To consider DARP under dynamic real-life setting, Kirchler et al. (2013) [67] has reported the solution quality at 30 sec, 60 sec, and 180 sec. We applied the same CPU time for the optimization process. We report solution quality for two objective functions $f' = f$, and $f'' = f' - w_5 * e(s)$, and used same weights for the objective function f as proposed in [29][9][67]: $w_1 = 8, w_2 = 3, w_3 = 1, w_4 = 1, w_5 = n$, and the coefficient α was set to 10000. *BKS* for all instances with objective functions f' and f'' are provided by [67][9]. We have included results for two objective functions to test the flexibility of our optimization algorithms. We get a solution that has better user comfort (lesser user excess ride times) when we use the objective function f' rather than f'' because we include the user excess ride

Case	n	m	VNS												
			P_2	P_2-opt_5	P_2-opt_4	P_2-opt_3	P_2-opt_2	P_2-opt_1	C_2+opt_5	C_2+opt_4	C_2+opt_3	C_2+opt_2	C_2+opt_1	C_2	
R1a	24	3	0	21	22	34	16	26	54	57	56	48	50	61	24
R2a	48	5	0	39	37	43	44	47	51	52	53	46	39	61	48
R3a	72	7	0	24	25	41	29	42	15	42	33	31	36	46	72
R4a	96	9	0	23	09	33	44	10	45	48	45	36	44	47	96
R5a	120	11	0	42	59	57	49	61	19	14	18	21	25	36	120
R6a	144	13	0	18	02	26	18	01	32	38	24	28	37	43	144
R7a	36	4	0	14	06	25	29	11	43	54	43	41	58	57	36
R8a	72	6	0	18	16	20	27	07	43	51	40	32	53	56	72
R9a	108	8	0	13	07	14	14	24	49	54	49	37	39	55	108
R10a	144	10	0	52	49	79	64	63	41	44	22	35	46	46	144
R1b	24	3	0	10	14	25	21	28	50	59	52	46	46	62	24
R2b	48	5	0	32	25	44	32	47	53	58	55	47	55	63	48
R3b	72	7	0	17	10	26	38	09	44	45	41	32	49	55	72
R4b	96	9	0	12	05	28	36	23	31	41	35	29	47	54	96
R5b	120	11	0	53	42	59	49	60	37	39	51	36	28	55	120
R6b	144	13	0	68	58	64	66	70	23	34	41	31	40	50	144
R7b	36	4	0	24	27	42	28	39	58	59	58	45	55	65	36
R8b	72	6	0	14	05	21	40	12	28	47	42	28	41	48	72
R9b	108	8	0	42	46	55	53	61	15	18	17	22	20	39	108
R10b	144	10	0	69	34	60	58	23	41	45	41	38	35	45	144
Avg.			0	30	25	39	38	33	39	45	41	36	42	52	

TABLE 3.8: Percentage deviation from the best speed up for the G-VNS methodology.

time factor into account.

Since we do not find existing GPU-based results for DARP, we compare the solution quality of our approach P_1, P_2 with the best-performing sequential approaches E_1, E_2, E_3, E_4 . From Tables 3.10, 3.11, and 3.12, the proposed GPU methodologies P attain better solutions than E for all the tested instances, and better than BKS for most of the instances within a short amount of time. It can be noted that the method P_1 outperforms P_2 in terms of average *gap* over the tested instances. In P_2 , the process of inserting selected customers into all vehicles is evaluated using GPU in a single batch. In a single batch, multiple GPU threads will be running simultaneously, in which each GPU thread evaluates the process of inserting a unique customer into a unique vehicle. In P_1 , the processes of inserting each customer into all vehicles are evaluated using GPU in multiple concurrent batches, i.e., the

Case	n	m	ALNS											
			C_3	C_3+opt_1	C_3+opt_2	C_3+opt_3	C_3+opt_4	C_3+opt_5	P_3-opt_1	P_3-opt_2	P_3-opt_3	P_3-opt_4	P_3-opt_5	P_3
R1a	24	3	48	36	21	40	38	34	21	29	28	27	35	0
R2a	48	5	39	37	16	33	31	35	27	37	33	30	38	0
R3a	72	7	51	48	20	50	37	29	27	46	28	30	38	0
R4a	96	9	42	39	22	35	33	30	22	37	28	27	34	0
R5a	120	11	44	41	22	38	32	30	19	24	27	23	33	0
R6a	144	13	45	39	26	36	31	33	19	25	24	18	30	0
R7a	36	4	52	48	19	48	37	38	30	41	32	31	37	0
R8a	72	6	55	54	16	44	45	46	26	40	26	30	35	0
R9a	108	8	48	40	23	39	40	39	23	25	23	23	34	0
R10a	144	10	45	34	22	38	33	32	17	22	20	19	28	0
R1b	24	3	48	42	27	35	40	38	25	33	24	21	33	0
R2b	48	5	46	42	15	41	40	40	28	44	31	32	36	0
R3b	72	7	49	43	22	46	43	41	27	47	32	29	38	0
R4b	96	9	49	46	25	43	38	33	28	38	30	25	36	0
R5b	120	11	45	37	21	39	35	33	20	31	26	25	34	0
R6b	144	13	38	33	24	32	37	31	18	20	28	22	32	0
R7b	36	4	53	42	17	42	38	42	26	46	29	30	37	0
R8b	72	6	54	42	15	45	40	40	27	41	28	30	35	0
R9b	108	8	46	35	18	30	36	33	20	40	24	23	31	0
R10b	144	10	45	34	25	31	35	28	22	24	25	27	30	0
Avg.			47	41	21	39	37	35	24	35	27	26	34	0

TABLE 3.9: Percentage deviation from the best speed up for the G-ALNS methodology.

process of inserting any customer into all vehicles is parallelized with concurrent kernels, therefore resulting in rapid exploration of a much larger neighborhood. Furthermore, the GPU-based ALNS P_3 exhibits better solution quality when compared to both the methodologies GPU-based tabu search P_1 and GPU-based variable neighborhood search P_2 . Figures 3.4–3.15 illustrate the difference in solution quality attained by the proposed GPU-based methodologies when compared to the existing sequential CPU-based methods for solving DARP. Since we have already presented detailed results in the Tables 3.10–3.12, the plotted graphs are only for illustrative purposes. For every graph, the scale of y-axis is varied to visualize the $gap(\%)$ value attained by various algorithms on each test instance at different time stamps. Furthermore, our goal is to not compare the results for different objective functions f' and f'' , but compare the results of various algorithms for different

objective functions. For all the tested instances, we have provided the *BKS* and *gap* values attained by the methods for both the objective function f' and f'' for the sake of comparison. In summary, the proposed methodologies outperform the existing methods in terms of solution quality within a short time.

Case	BKS	E_1	E_2	E_3	E_4	P_1	P_2	P_3
f'								
R1a	3152.22	0.13	0.48	2.66	1.02	0.00	0.00	0.00
R2a	6027.29	151.51	29.79	5.07	7.52	-3.16	-3.64	-0.08
R3a	9952.00	50.25	1.45	17.12	1.74	-1.74	0.17	-1.48
R5a	13050.30	123.61	67.57	169.20	8.34	-1.88	0.98	1.24
R9a	13912.96	Inf	91.33	168.93	50.81	11.79	21.09	24.93
R10a	18259.40	Inf	33.92	101.04	14.94	6.63	19.42	8.37
R1b	2874.74	0.00	1.44	1.13	2.80	0.00	0.00	0.00
R2b	4969.28	3.47	0.35	17.95	1.98	-0.65	-0.26	-0.34
R5b	11747.20	33.31	90.71	109.01	8.59	0.48	1.62	1.56
R6b	15000.50	51.57	100.24	127.68	8.61	0.09	1.90	1.54
R7b	4365.98	5.72	0.49	5.08	1.06	-1.10	-0.61	-0.85
R9b	12265.50	30.26	46.32	84.76	4.68	-0.54	1.45	-0.25
R10b	16391.00	31.30	190.87	232.66	34.05	5.22	12.57	6.41
f''								
R1a	3118.33	0.14	0.91	0.38	1.23	0.00	0.00	0.00
R2a	5546.55	1.38	3.29	4.78	3.05	-0.03	1.09	-0.51
R3a	9632.47	2.91	3.66	23.85	3.91	0.53	1.56	1.17
R5a	12642.77	11.47	59.16	64.40	7.91	1.10	2.83	1.43
R9a	13301.65	Inf	83.04	101.83	50.78	10.10	28.59	13.65
R10a	17459.16	Inf	37.74	105.62	12.09	5.91	10.42	8.50
R1b	2874.74	0.00	1.35	2.67	3.48	0.00	0.00	0.00
R2b	4929.08	1.32	2.65	16.14	2.63	0.09	0.58	0.05
R5b	11635.79	11.82	83.45	66.09	7.63	1.24	2.58	1.08
R6b	14927.1	12.00	91.74	69.22	10.05	0.81	1.95	1.14
R7b	4290.11	0.83	1.82	2.84	0.47	-0.53	-0.36	-0.52
R9b	12021.8	10.19	48.52	78.77	7.39	1.46	1.64	1.18
R10b	16238.27	20.25	148.07	235.76	45.83	4.02	11.30	6.27
Avg. <i>gap</i>	0.00	Inf	46.94	69.79	11.64	1.53	4.50	2.87

TABLE 3.10: Comparison of solution quality (*gap*) of our approach (P) when compared to the existing methods (E) after 30 s of CPU time.

FIGURE 3.4: (A) Comparison of solution quality (gap) with objective function f' between Proposed and Existing methods on DARP instances after 30 sec of CPU time.

FIGURE 3.5: (B) Comparison of solution quality (gap) with objective function f' between Proposed and Existing methods on DARP instances after 30 sec of CPU time.

FIGURE 3.6: (C) Comparison of solution quality (gap) with objective function f'' between Proposed and Existing methods on DARP instances after 30 sec of CPU time.

FIGURE 3.7: (D) Comparison of solution quality (gap) with objective function f'' between Proposed and Existing methods on DARP instances after 30 sec of CPU time.

FIGURE 3.8: (A) Comparison of solution quality (gap) with objective function f' between Proposed and Existing methods on DARP instances after 60 sec of CPU time.

FIGURE 3.9: (B) Comparison of solution quality (gap) with objective function f' between Proposed and Existing methods on DARP instances after 60 sec of CPU time.

FIGURE 3.10: (C) Comparison of solution quality (gap) with objective function f'' between Proposed and Existing methods on DARP instances after 60 sec of CPU time.

FIGURE 3.11: (D) Comparison of solution quality (gap) with objective function f'' between Proposed and Existing methods on DARP instances after 60 sec of CPU time.

FIGURE 3.12: (A) Comparison of solution quality (gap) with objective function f' between Proposed and Existing methods on DARP instances after 180 sec of CPU time.

FIGURE 3.13: (B) Comparison of solution quality (gap) with objective function f' between Proposed and Existing methods on DARP instances after 180 sec of CPU time.

FIGURE 3.14: (C) Comparison of solution quality (gap) with objective function f'' between Proposed and Existing methods on DARP instances after 180 sec of CPU time.

FIGURE 3.15: (D) Comparison of solution quality (gap) with objective function f'' between Proposed and Existing methods on DARP instances after 180 sec of CPU time.

Case	BKS	E_1	E_2	E_3	E_4	P_1	P_2	P_3
f'								
R1a	3152.22	0.13	0.48	2.66	0.84	0.00	0.00	0.00
R2a	6027.29	154.01	11.72	5.07	5.30	-3.12	-3.80	-4.73
R3a	9952.00	56.46	0.62	7.86	1.23	-2.45	-1.06	-1.75
R5a	13050.30	93.20	24.28	129.97	6.09	-2.60	0.09	-2.41
R9a	13912.96	18.11	46.11	82.21	50.81	10.68	19.23	23.99
R10a	18259.40	Inf	19.60	94.47	9.93	-2.84	12.26	-1.85
R1b	2874.74	0.00	1.44	1.13	1.18	0.00	0.00	0.00
R2b	4969.28	1.68	0.35	4.80	1.62	-1.12	-0.35	-0.54
R5b	11747.20	26.96	59.08	98.36	6.90	-1.29	1.33	-0.05
R6b	15000.50	45.86	78.62	114.07	5.77	-1.33	1.52	-0.39
R7b	4365.98	5.35	0.00	5.08	1.06	-1.40	-1.18	-1.37
R9b	12265.50	33.78	17.62	74.83	1.92	-1.80	0.88	-1.15
R10b	16391.00	19.73	69.52	224.47	26.57	-1.20	9.44	1.63
f''								
R1a	3118.33	0.14	0.00	0.02	1.23	0.00	0.00	0.00
R2a	5546.55	0.99	3.29	3.58	2.11	-0.20	1.09	-0.68
R3a	9632.47	1.82	2.12	8.60	3.10	-0.01	1.01	0.69
R5a	12642.77	9.02	18.83	55.39	6.00	0.01	1.92	0.39
R9a	13301.65	18.26	81.41	51.40	34.99	7.67	26.54	11.39
R10a	17459.16	Inf	13.18	95.96	10.27	-0.40	7.28	1.62
R1b	2874.74	0.00	1.35	2.67	3.48	0.00	0.00	0.00
R2b	4929.08	1.02	2.65	4.43	2.63	-0.42	-0.12	-0.06
R5b	11635.79	9.43	54.02	59.64	6.01	-0.85	1.80	0.06
R6b	14927.1	10.29	72.22	64.46	5.96	-0.88	1.29	-0.70
R7b	4290.11	0.45	1.82	2.84	0.38	-0.60	-0.48	-0.52
R9b	12021.8	8.39	19.60	76.41	3.78	0.11	1.24	0.40
R10b	16238.27	16.07	67.18	207.99	45.65	0.82	7.43	4.07
Avg. <i>gap</i>	0.00	Inf	25.66	56.86	9.42	-0.12	3.36	1.08

TABLE 3.11: Comparison of solution quality (*gap*) of our approach (P) when compared to the existing methods (E) after 60 s of CPU time.

3.8 Discussion

On-demand mobility has become a topic of interest due to its wide range of applications including medical emergency transit, public transportation, airport, and seaport operations. In the last few years, the general public has developed a significant interest in MoD transportation. Dial-a-ride problem is a form of MoD, which is NP-hard in nature. In today's world, DARP often occurs in "real-time" rather than in advanced reservation mode, and hence attaining quicker solutions

Case	BKS	E_1	E_2	E_3	E_4	P_1	P_2	P_3
f'								
R1a	3152.22	0.04	0.48	2.67	0.48	0.00	0.00	0.00
R2a	6027.29	116.09	3.46	3.37	1.71	-4.29	-4.24	-4.73
R3a	9952.00	50.21	0.13	5.08	0.10	-2.55	-1.76	-2.66
R5a	13050.30	68.90	3.01	50.34	1.76	-2.82	-0.72	-3.27
R9a	13912.96	12.38	29.01	21.98	48.88	6.32	6.55	2.37
R10a	18259.40	35.67	11.59	75.56	5.84	-4.42	5.40	-2.85
R1b	2874.74	0.00	1.18	1.13	1.18	0.00	0.00	0.00
R2b	4969.28	-0.48	0.35	3.43	1.29	-1.15	-0.58	-1.12
R5b	11747.20	15.60	13.83	50.90	3.95	-1.73	0.35	-1.44
R6b	15000.50	10.42	36.55	104.63	2.57	-1.68	0.24	-1.42
R7b	4365.98	4.80	0.00	5.08	1.06	-1.40	-1.40	-1.40
R9b	12265.50	7.44	2.61	65.38	1.60	-2.04	-0.64	-2.24
R10b	16391.00	19.86	10.00	191.33	6.12	-2.14	4.68	-2.25
f''								
R1a	3118.33	0.05	0.00	0.00	0.21	0.00	0.00	0.00
R2a	5546.55	-0.62	1.50	2.38	1.99	-0.34	0.00	-0.83
R3a	9632.47	0.39	1.86	3.00	3.10	-0.16	0.57	0.69
R5a	12642.77	5.42	2.75	28.14	3.59	-0.32	1.16	-0.49
R9a	13301.65	8.25	29.09	47.74	27.97	7.67	9.09	8.65
R10a	17459.16	12.53	7.54	81.76	5.50	-1.51	4.12	-1.62
R1b	2874.74	0.00	1.07	2.67	1.46	0.00	0.00	0.00
R2b	4929.08	0.33	0.94	1.78	1.97	-0.48	-0.35	-0.53
R5b	11635.79	5.18	10.60	51.22	2.96	-0.98	0.60	-0.97
R6b	14927.1	7.13	31.82	54.32	3.86	-1.43	0.49	-1.55
R7b	4290.11	-0.17	0.21	2.14	0.38	-0.60	-0.60	-0.60
R9b	12021.8	4.92	4.23	61.81	3.39	-0.12	0.54	-0.47
R10b	16238.27	10.65	8.58	193.24	10.82	-0.91	2.96	-1.19
Avg. <i>gap</i>	0.00	15.19	8.17	42.73	5.53	-0.66	1.02	-0.77

TABLE 3.12: Comparison of solution quality (*gap*) of our approach (P) when compared to the existing methods (E) after 180 s of CPU time.

can be critical. GPU computing has turned out to be a fundamental pillar in modern science. In this chapter, we have proposed a GPU-based solution methodology to solve DARP under dynamic real-life settings. First, we have introduced the building blocks of the proposed GPU framework including CUDA kernels and local search operation. Secondly, we have proposed five device optimization strategies tailored for the described problem to improve computational speedup using GPU. Thirdly, we have integrated the proposed framework with three metaheuristics: tabu search, variable neighborhood search, and adaptive large neighborhood

search. Simulations are conducted on DARP benchmark instances based on the Montreal Transit Commission from the literature. The proposed GPU approach with the device optimization strategies attains higher speedup when compared to the conventional GPU approach. The effect of inclusion and exclusion of each device optimization strategy on computational speedup has also been reported. Overall, the proposed GPU approach attains better average solution quality when compared to the existing best-performing sequential approaches for DARP.

The source of the computational efficiency is the proposed GPU framework along with the device optimization strategies. The computational speedup of the GPU framework varies depending on the type of integrated metaheuristic. For example, in tabu search, the neighborhood exploration represents relocating "all requests" into their potential positions in all vehicles. Meanwhile, in ALNS/VNS, the neighborhood exploration represents relocating a "selected number of requests" into their potential positions in all vehicles. Therefore, it can be understood that a higher speedup can be expected when integrating TS with our GPU framework due to its higher arithmetic intensity in GPU when compared to ALNS/VNS. The difference between the realization of proposed and existing parallel computing algorithms in GPU is that we introduce various device optimization strategies to demonstrate a thorough investigation of the effect of GPU in accelerating DARP solution algorithms. Besides, we do not find any existing GPU-based algorithms for DARP in the literature as discussed in Section 3.2.2. We do not find convergence issues. The considered DARP instances have NP-hard computational complexity and never solved optimally. Therefore, we show that a better convergence and solution quality are achieved by our parallel algorithms in comparison with existing best-performing sequential approaches.

We conclude that accelerating time-critical neighborhood explorations in GPU along with a guided search of metaheuristic can be significant in efficiently solving DARP.

Following are some potential future research directions: 1) introducing GPU-based neighborhood reduction techniques to address very large scale problems, 2) solving the richer variants of DARP with heterogeneous demand, fleet, dynamic traffic conditions, 3) applying the proposed framework for emergency transportation services (e.g. medical ambulance, fire and rescue operations), where attaining quick

solutions is critical, 4) enabling the GPU memory coalescing techniques, 5) utilizing multi-core CPU alongside the proposed GPU framework to further enhance the computational speedup.

Chapter 4

Disruption Management Framework for Dial-A-Ride Problem

Summary - Mobility-on-demand has been gaining more attention from the research community to offer smart and efficient transportation services to people.

Despite the advancements in vehicular technologies, vehicle breakdown remains as one of the major contributors to the disruption in fleet operations, which may inflict large recovery costs and damage service providers' reputation. However, modeling such dynamic events for dial-a-ride systems has not been addressed until now, which is the research gap we attempt to address in this chapter.

The following are the major contributions of this chapter: a formulation of disruptive dial-a-ride problem with vehicle breakdown (DDARP-VB) with several realistic constraints, and a disruption management framework, which consists of a combination of online optimization method based on GPU-based adaptive large neighborhood search (G-ALNS) and a fleet size minimization (FSM) strategy that leads to reductions in operational costs.

From our simulations, the proposed G-ALNS algorithm performs significantly faster than its CPU counterpart and produces better solutions when compared to the existing GPU approach for DARP. Overall, our FSM strategy leads to a reduction in fleet idle time under normal operations and a reduction in operational cost under disruption.

The preliminary results of this chapter are published as a conference paper in Ramesh et al. 2019 [124], and the journal version is published as Ramesh et al. 2020b [131] in the IEEE Intelligent Transportation Systems Magazine.

4.1 Introduction

Smart urban mobility [197][198][199] is one of the prime areas of focus for most of the countries nowadays to fulfill people's daily transportation needs, enhance travel comfort, and provide superior user convenience. Rapid advancements in the areas of autonomous vehicle technology, parallel computing, optimization techniques for vehicle routing, work as the driving forces to achieve this objective. Many day-to-day services such as emergency vehicles, door-to-door people transport, goods delivery, etc. need efficient fleet management solutions developed using these emerging technologies to provide satisfactory service to the customers.

Dial-a-ride problem (DARP) [7], a special class of vehicle routing problem (VRP), belongs to the category of people transportation, offers pick-up and drop-off services to a group of users between their preferred origin and destination locations, while minimizing the total travel cost subject to a set of constraints. Some of these constraints include vehicle load, route duration, passenger ride time, and time windows associated with pick-up and drop-off. Several optimization algorithms exist in the literature to provide optimal [28][11] or near-optimal [103] solutions to solve this problem. However, in many real-life scenarios, fleet operations are often hindered by the occurrence of many unanticipated stochastic events, such as vehicle breakdown, accidents, drastic weather conditions, etc., which in turn diminish the solution quality. According to [5], with advances in communications technologies, the dial-a-ride (DAR) system evolves from being based on advanced reservations to a real-time vehicle dispatch system. Thus, fast on-line algorithms are needed to be able to insert new trips into routes that are already on-going. To achieve the necessary computational speeds, parallel computation is a promising research direction. The technological advancements are constantly creating new trends to operate DAR services. Particularly, GPU technology has evolved significantly in the past decade with the introduction of the Compute Unified Device Architecture (CUDA) programming model. Much progress has been made on computing technologies over the years, with the invention of high-performance parallel-computing

hardware and its usage in non-graphical computationally intensive general-purpose applications.

Besides designing optimal routes, another important problem in the fleet management system is the minimum fleet problem, which in general, focuses on determining a minimum number of vehicles required to provide an uninterrupted and high-quality transportation service to the users [16]. According to a recent study on fleet size minimization for taxi services, in New York City alone, taxis travel around the city without serving any passengers for about 40% of the time [17]. With the increased need for transportation, several other issues such as traffic congestion, carbon emissions per vehicle, passenger ride times, etc. also increase significantly. Furthermore, the fleet depreciation loss, maintenance cost, supporting infrastructure cost, etc. pose an additional burden on the service providers. Thus, there is a great need for fleet size minimization in DAR systems.

Regardless of how technology advances, vehicle breakdowns lead to major disruption of transportation services across the globe. In the US alone, a massive 32 million car breakdowns are recorded [18] in 2015 and the numbers are rising further every year. In the context of dial-a-ride systems, breakdowns result in an inflated operational cost and damaged reputation for the service providers (operators of taxis, private buses, etc.). Therefore, the development of efficient fleet management strategies to tackle vehicle breakdown is vital to provide uninterrupted transportation service to the passengers.

Although the problem of solving vehicle routing with a breakdown is very important in many real-world applications, further research needs to be carried out to address this problem in greater detail. However, only a handful of methods are available in the literature to solve this problem. A Tabu Search algorithm [200] for disruption management is proposed to generate new route plans for the delivery problem. Optimization models based on networks for complex aircraft re-routing and crew scheduling are detailed in [201] and [202]. [203] discusses different formulations and existing methods in the literature to address disruption management in road freight transport systems. A saving based heuristic [16] is proposed to evaluate the reliability of vehicle routing schemes in the vaccine delivery with a vehicle breakdown. A disruption management mathematical model [17] and a Lagrange heuristic are developed to address the urgency vehicle routing problem. An improved genetic algorithm [204] was proposed to address VRP with breakdowns.

4.1.1 Our Contributions

In this section, we describe the main contributions of this chapter.

- According to Eglese et al. (2018) [203] - a survey on disruptive VRP, there is no existing model for DARP with vehicle breakdowns. The most relevant studies on vehicle breakdown in VRP-context can be found in Li et al. (2009) [205] and Qu et al. (2011) [200]. In this chapter, we extend the formulation of VRP-VB by Qu (2011) and DARP by Cordeau and Laporte (2003), to model DDARP-VB, with new additional realistic assumptions such as immediate fleet re-routing after a vehicle breakdown, the decision to serve all the customers to avoid damage on service providers' reputation, design of a realistic operational cost function, and dynamic updates in travel times between the newly generated nodes, i.e., location of vehicles during vehicle breakdown and all the other nodes in the network.
- According to Psaraftis et al. (2016) [171] - a survey on dynamic VRP, the future development of parallel algorithms may help in further reducing the computational time, especially for dynamic problems. In literature, the dial-a-ride problem with traffic or demand uncertainty has been addressed. The common methodology is to forecast the customer demand [19] or traffic uncertainty [20] to perform anticipative routing. However, in the case of vehicle breakdown, it is not possible to rely on prediction outputs. Therefore, there is a great need to develop a technique for a recourse action as soon as possible in an event of a vehicle breakdown. At this point, ALNS is one of the outstanding algorithms for DARP. It was developed by Ropke and Pisinger (2006) for VRP-TW. In this chapter, we extend this basic ALNS approach with GPU acceleration to address the real-time dynamic rescheduling for disruption in the dial-a-ride context. Our method is a novel approach since we do not find such G-ALNS in the literature for any extension of VRP-TW, including DARP. We are able to report better results both in terms of computational time and solution quality for our GPU approach when compared to the CPU version of ALNS and a recent GPU-accelerated Tabu Search method.
- According to Sin C Ho et al. (2018) [5] - a survey on DARP, "As every transport operator knows, things never go according to plan. For DAR systems, as for other transport systems, disruption management is essential". New

modeling techniques and frameworks are required for disruption management. In this chapter, we attempt to address this research gap by modeling the DDARP-VB problem and presenting a G-ALNS solution algorithm. Our fleet management strategy that is based on minimizing the fleet size for normal routing operations, and addressing the disruption using a backup fleet in combination with online optimization is novel since we did not find such an approach in the literature. Our results suggest that the proposed strategy attains reduced operational cost and fleet idle time on both normal and disruptive scenarios. We also investigate the influence of the breakdown and effectiveness of the proposed approach on passenger experiences. We have discovered that the passenger experience in terms of user ride improves both before and after the breakdown.

The remainder of this chapter is organized as follows. In Section 4.2, we describe the proposed formulation of DDARP-VB. In Section 4.3, we present the proposed disruption management framework and route optimization algorithm. In Section 4.4, we present various disruption management schemes for DDARP-VB. We report the simulation results and provide comparative analysis in Section 4.5, and offer concluding remarks in Section 4.6.

4.2 Formulation of Dial-A-Ride Problem with Vehicle breakdown

In this section, we introduce the formulation of the disruptive dial-a-ride problem with vehicle breakdown (DDARP-VB).

Dial-a-ride problem addresses the transportation of people between specified pick-up and drop-off locations. DARP is solved as a combinatorial optimization problem to minimize the total travel time while satisfying all the constraints related to passenger convenience. The formulation of DARP in [7] is often regarded as the standard version. In the standard DARP, a homogeneous set of requests are considered with a fixed number of vehicles located at a single depot. All vehicles are considered to have constant speed throughout the network, and euclidean distance formula gives the distance between any two nodes in the network. Since it is one

of the most widely studied DARP in the literature, our formulation of DARP with vehicle breakdown is built upon it with new additional realistic constraints related to the event of a vehicle breakdown. This formulation can be generalized/applicable for other disruptive events (e.g. severe weather, accidents, engine failure, etc.) with multiple vehicles. However, for the sake of brevity, in the following, we describe along the lines of disruptive DARP with single-vehicle breakdown.

In the standard dial-a-ride problem, n customer requests are served by m vehicles. Each request i consists of the time window either for departure or arrival vertex or both. The objective is to minimize the travel cost subject to several constraints. Let $X = \{x_1, x_2, \dots\}$ denote the solution space. All solutions $x_i \in X$ need to satisfy three basic constraints: 1) every route for a vehicle k starts and ends at the depot, 2) departure vertex v_i and arrival vertex v_{i+n} must belong to the same route, and 3) arrival vertex v_{i+n} is always visited after departure vertex v_i . Any solution that violates these basic sets of constraints becomes infeasible. Additionally, four major constraints need to be satisfied: load, duration, time window, and ride time constraints. Load constraint violation $q(x)$ occurs when the number of passengers in a vehicle k exceeds its load limit Q_k ; duration constraint violation $d(x)$ happens when a vehicle k exceeds its duration limit T_k ; time window constraint violation $w(x)$ appears when the time constraint on pick-up and drop-off is violated; ride time constraint violation $t(x)$ occurs when a passenger is transported for a longer time than ride time limit L . B_i represents the service starting time at vertex i ; L_i represents the ride time of request i . For detailed formulation, the readers are referred to as [28]. The above-mentioned four constraints can be formulated mathematically as follows:

$$q(x) = \sum_{\forall k} \max(q_{k,max} - Q_k, 0), \quad (4.1)$$

$$d(x) = \sum_{\forall k} \max(d_k - T_k, 0), \quad (4.2)$$

$$w(x) = \sum_{\forall i} [\max(B_i - l_i, 0) + \max(B_{i+n} - l_{i+n}, 0)], \quad (4.3)$$

$$t(x) = \sum_{\forall i} \max(L_i - L, 0). \quad (4.4)$$

A common DARP application arises in door-to-door transportation services for the elderly and the disabled. In this context, users often populate two requests per day:

an outbound request from home to a destination, and an inbound request for the return trip. Outbound requests, e.g., from home to the hospital, have a tight time window on the destination. Inbound requests, e.g., from the hospital back home, have a tight time window on the origin [9]. As in the work of Cordeau and Laporte (2003) [7], a more realistic way to model the DARP is to let users impose a time window of a pre-specified width on the arrival time of their outbound trip and, similarly, a window on the departure time of their inbound trip. This approach seems to be in line with the practice of several North American transporters [28]. Therefore, we follow a similar assumption in our work along these lines.

According to the previously published works in [9][10][11][103], the version of DARP proposed in [7] is regarded as “standard DARP”, since it is widely treated in the literature. According to [7], “The information regarding time window widths, vehicle capacity, route duration, and maximum ride time was provided by the Montreal Transit Commission (MTC), Canada. The instances are generated according to realistic assumptions”. Inferring from the literature [9][11][77], several recently studied new DARP variants generally extend upon the DARP formulation of [7]. Since our work is the first attempt to investigate vehicle breakdown in the dial-a-ride context, we have considered a similar approach and extend upon the formulation of [7]. The parameters related to vehicle capacity and speed may be defined as vehicle-specific. Similarly, vehicle-specific depots can be defined. Nevertheless, in our DDARP-VB formulation, we have considered such parameters and sets to be homogeneous among all vehicles. This assumption is reasonable in the deployment of autonomous electric buses, as nowadays such vehicles mount the same commercial batteries and do not significantly differ in terms of dimensions, capacity, and weights [24]. Furthermore, the instances of [7], in which we conduct experiments, have considered similar vehicle capacities that reflect the situation of a transporter using mini-buses for the transportation of individuals or groups of individuals.

The most relevant studies for vehicle breakdown in VRP-context can be found in [205] and [200]. We formulate the DDARP-VB by extending the existing VRP-VB model [200] and standard DARP model [7]. Additional assumptions made for this formulation are: (1) dynamic vehicle re-routing happens at the time of disruption, without requiring vehicles to first arrive at their respective destinations; (2) all solutions must be complete, i.e., all requests must be served irrespective of service disruptions; (3) operational cost is computed based on a realistic estimation of

today's transportation scenario. We present the new constraints related to the proposed DDARP-VB as follows:

- All accepted travel requests must be served either feasibly or infeasibly.
- During the event of a breakdown, any vehicle from the fleet must be in one of the followings states:
 - vehicle has not left the depot
 - vehicle is traveling, with no passenger on-board
 - vehicle is traveling, with passenger(s) on-board
 - vehicle is at a service location, with no passenger on-board
 - vehicle is at a service location, with passenger(s) on-board
 - vehicle has returned to the depot
- A passenger i , affected by breakdown event, can be served by another vehicle. For example, when a passenger i is inside a vehicle k_{VB} that involved in an event of a breakdown, then the passenger i can be served by another vehicle k_{VR} while ensuring all the constraints are met. If the passenger i is dropped off by a vehicle that is involved in a breakdown event, then the time window for the subsequent pick-up will be set between breakdown time and the end of the planning horizon. Therefore, if that passenger request is inbound, then it is very likely to be served by a new vehicle without constraint violation. However, if it is an outbound request, the previously tighter schedule produced by the algorithm decreases the chance of feasibly serving this user. In the event of vehicle breakdown, it is important to ensure uninterrupted transportation service for all passengers with minimal disruption to passenger convenience.
- Ride time limit L , which is very important in DARP, needs to be satisfied irrespective of vehicle breakdown to guarantee feasibility. If a passenger i is inside a vehicle k_{VB} , which is broken down, then the passenger i is dropped off at the location of breakdown. Furthermore, a new request will be dynamically generated with the altered constraints from the previous problem, and the new ride time constraint will be set as the difference of original ride time limit L and the travel time L_i of the passenger i during the time of breakdown.

- When a vehicle k involves in a breakdown event, it cannot serve any more passengers for the particular day. Therefore, it rejects all the scheduled requests to be visited in the future, while dropping off the on-board passengers immediately at the location of breakdown.

4.3 Proposed Disruption Management Framework

In this section, we introduce an adaptive algorithm for the disruptive DARP with vehicle breakdown (DDARP-VB). First, we describe the implementation of disruption management framework for dial-a-ride system in Section 4.3.1. Second, we describe the GPU-based optimization methodology (G-ALNS) to quickly generate vehicle routes and schedules for the DDARP-VB in Section 4.3.2.

4.3.1 Overall Framework

In this section, we describe the implementation of disruption management framework for dial-a-ride system.

Figure 4.1 shows the flowchart of our real-time implementation of DDARP-VB. As the service provider dispatches the fleet to serve the passengers based on the solution provided by the algorithm, the *traffic monitoring system* keeps track of the event sequence. In case of unforeseen vehicle breakdown, the information corresponding to the broken-down vehicle will be fed back to the optimizer, which includes vehicle coordinates (x_{VB}, y_{VB}) , vehicle identity k_{VB} , and event occurrence time t_{VB} . In reflection, the algorithm dynamically optimizes routes and returns a new solution that has a minimum value of the objective function. All the requests are served irrespective of feasibility to maintain the service provider's reputation. However, in the real world, the decisions on accepting passenger requests may be often based on several factors. After all the passengers in the current demand-pool have been served, and all vehicles from the fleet have returned to the depot, the system can be shut down. To enhance the search efficiency of the optimization algorithm, we incorporate a tabu list that permanently forbids a set of requests that are already served at the time of disruption, and *halt_Index* to check the feasibility

FIGURE 4.1: Real-time adaptive algorithm for the dial-a-ride system.

of request insertions into the solution after the service disruption. Algorithm 14 shows the implementation details, as described below.

- Allocate GPU memory resources and transfer the DARP instance data from CPU to GPU.
- Generate an initial solution for the given instance using a construction heuristic. Here, a solution is encoded as a set of routes and schedules. Afterward, this initial solution is set as the current solution.
- The route plan for the current solution is optimized using a GPU-based adaptive large neighborhood search algorithm (G-ALNS), then the service provider can start to dispatch vehicles according to the best solution provided by the algorithm. Thereafter, on each iteration, the algorithm attempts to further minimize the objective function, while monitoring any cases of vehicle breakdown based on the information provided in real-time or simulation feedback.
- When the *traffic monitoring system* detects a breakdown:
 - Extract the information of the broken vehicle k_{VB} and the event occurrence time t_{VB} from the system. For simulations, we randomly generate the values of k_{VB} and t_{VB} , provided that the vehicle k_{VB} has started the service, but not ended for the day.
 - The location of all vehicles during the event can be retrieved using the global positioning system in real-time or externally computed in simulation.
 - After the event, all the previously served requests are updated in a list called $T_{tabulist}$. Such requests in the list will not be involved in the post-breakdown online route optimization process. Furthermore, we also record the *halt_Index* that contains previously visited nodes at the time of breakdown to simplify the post-optimization algorithmic process.
 - If there are passengers inside the vehicle k_{VB} , then they will be dropped off immediately at the breakdown location. Meanwhile, the travel times from the breakdown vertex to all other vertices are computed, besides adjusting the constraints of affected users.

- The vehicle k_{VB} is discarded from service once the disruption occurs.
- Once the stopping criterion is satisfied, the best solution that has minimum travel cost is returned and the GPU resources are released.

Algorithm 14 Overall implementation of the DAR system

Require: DARP Instance

Ensure: *BestSolution*

```

1: Allocate GPU resources
2: Copy DARP instance from CPU to GPU
3: Build an initial solution using construction heuristic
4: Set current solution as the generated initial solution
5: repeat
6:   if check(trigger_breakdown) then
7:     Choose a vehicle  $k_{VB}$  to breakdown at time  $t_{VB}$ 
8:     Compute the coordinates for all vehicles  $k \in m$ 
9:     Construct the tabu list  $T_{tabulist}$  and  $halt\_Index$ 
10:    if passengers  $i_b \in n$  inside  $k_{VB} > 0$  then
11:      Update coordinates of the passengers  $i_b \in n$ 
12:      Update travel times between  $i_b \in n$  and  $i \in n$ 
13:      Disable the vehicle  $k_{VB}$  after disruption
14:    end if
15:  else
16:    Copy the current solution from CPU to GPU
17:    while considering  $T_{tabulist}$  and  $halt\_Index$  do
18:      Perform removal operation
19:      Launch Neighborhood_Exploration kernel in GPU
20:    end while
21:    Copy selective data from GPU to CPU
22:    Select the next solution based on G-ALNS
23:    Adaptively adjust the ALNS operator weights
24:  end if
25: until stopping criterion is fulfilled
26: Release GPU resources

```

The general idea of LNS by Shaw (1998) [195] is to repeatedly remove requests from the solution and to reinsert them at more profitable positions. Later, an adaptive weight adjustment scheme to LNS was introduced by Ropke et al. (2006) [164] to develop ALNS for vehicle routing problems with time-windows. In this section, we extend this ALNS approach with GPU-acceleration to solve disruptive DARP-VB. The idea of exploring neighborhoods using GPU for DARP was introduced in our preliminary conference paper in Ramesh et al. (2018a) [6], which was applied for

a Tabu Search algorithm. In this section, we develop a GPU-accelerated Adaptive Large Neighborhood Search (G-ALNS) algorithm to solve the disruptive DARP-VB.

4.3.2 G-ALNS Optimization Framework

In this section, we describe the GPU-based adaptive large neighborhood search methodology (G-ALNS) to quickly optimize vehicle routes and schedules for the DDARP-VB.

Algorithm 15 Algorithm: GPU-based ALNS framework

Require: DARP Instance

Ensure: *Best solution found*

- 1: $CurrentSolution \leftarrow InitialSolution$
 - 2: **while** the termination condition is not satisfied **do**
 - 3: Select ALNS operators according to their past scores
 - 4: Apply *Destroy operator* to the *CurrentSolution*
 - 5: *DataTransfer from CPU to GPU*
 - 6: *// Evaluating neighborhoods in a single batch*
 - 7: Apply *Repair operator* i.e., **Explore neighborhood in GPU** /*launch GPU-accelerated Local Search()*
 - 8: *DataTransfer from GPU to CPU*
 - 9: Select best solution among the neighborhood solutions based on SA-acceptance criterion of ALNS in CPU
 - 10: Set the newly selected solution as *Current solution* in CPU
 - 11: Update $Cost_{best}$ of the *Best solution found* in CPU
 - 12: Trigger the fleet size minimization technique (Section 4.3.2.1)
 - 13: **end while**
 - 14: **return** *BestSolution*
-

Algorithm 3 details the neighborhood exploration in GPU. Algorithm 15 details the proposed G-ALNS framework. The algorithm starts with an initial solution. For each iteration, a pair of destroy and repair ALNS operators are adaptively selected from a pool of operators using a roulette wheel mechanism [164]. First, a destroy operator is applied to remove some requests from the current solution. Then, a repair operator explores the neighborhood using CUDA kernels (Section 3.4.3) in GPU and selects an appropriate solution in CPU based on simulated annealing acceptance criterion. The weights of ALNS operators are adjusted after every segment according to predefined rules. For a detailed description of ALNS heuristic, the readers are referred to the work of Ropke et al. 2006 [164].

We implemented the following neighborhood search operators (destroy and repair operators) for ALNS.

- Destroy operators
 - *Worst removal*: This operator iteratively removes requests that cause the biggest detour in the current solution.
 - *Random removal*: This operator randomly selects the requests to be removed among the planned requests.
 - *Related removal (or Shaw removal)*: This operator aims to remove requests that are alike so that their positions of service can somehow be interchanged.
- Repair operators
 - *Best insertion*: At each iteration, the best insertion cost is computed for each unplanned request and the request with the lowest insertion cost is inserted at its best position. The heuristic stops when all requests are routed or none can be inserted anymore.
 - *Regret insertion*: This heuristic is based on the notion of regret, which is used by Potvin [196] for the Vehicle Routing Problem with Time Windows (VRPTW), and extended by Ropke [164]. It investigates the cost of inserting a request on the k best routes, and finally inserts it on a route whose cost difference between inserting it into the best route and the $k-1$ best routes is largest. In our implementation, we consider regret- k heuristics with values of k between 1 and 5.

In addition to the above-mentioned operators, we perform intra-route optimization whenever a new best solution is found. In intra-route optimization, we remove and re-insert all requests in the best position on their respective routes. Furthermore, we invoke tabu search for every 200 iterations with a stopping criterion of 100 iterations. We summarize the implementation of our GPU-based ALNS method for DARP in Algorithm 12. The procedure can be described as follows: 1) allocate device resources, 2) construct initial solution (as per Section 3.7), 3) copy *Current solution* from CPU to GPU, 4) construct and evaluate neighborhood solutions in GPU, 5) selectively copy data from GPU to CPU, and select a best admissible

solution in CPU, 6) trigger restart mechanism in CPU if required, 7) record the cost of the best solution $Cost_{best}$ in CPU, 8) repeat steps 3-7 until the stopping criterion, 9) print the *Best solution found*, and release the memory resources.

4.3.2.1 Fleet size minimization

The difference between the GPU-based optimization for general DARP and disruptive DARP is the introduction of a new fleet management strategy into the optimization algorithm. In this strategy, we minimize the fleet size alongside optimizing vehicle routes for the DARP before vehicle breakdown. Whenever a vehicle breakdown event occurs, we utilize the available vehicles from the depot (resulted due to the previous fleet size minimization) along with an online optimization technique for disruption management. We consider vehicle minimization as the first priority. Therefore, we devise a simple two-stage algorithm. In the first stage, we start the algorithm with a number of vehicles specified in the instance. In the second stage, when the best solution does not improve anymore for a predefined number of iterations, we reduce the fleet size by one, and greedily reinsert all the requests from the disabled route into all other routes, and continue the optimization process. A similar approach can be found in the literature [164]. However, we have not seen such a method being applied to address disruptions for any variant of VRP, including DARP. The key challenge of fleet size minimization strategy is to keep the running time down. In our case, if the optimization process cannot find a feasible solution with a fleet size F for a limited number of iterations, then we fix the fleet size to be $F + 1$ and do not try to minimize the fleet size anymore, and spend the remaining available time only on the route optimization process.

4.3.2.2 Implementation details

To benchmark the efficiency of our GPU-based ALNS for rescheduling problems that require quick solutions, it is important to re-implement the same ALNS in CPU since Masson et al. (2014) [77] did not report the following: i) solutions at various time instances < 30 sec, ii) algorithmic parameters, iii) CPU type, iv) style of hash table implementation and v) the notion of the critical vertex. In our approach, we use a separate chaining approach with the hash function $f(x) \bmod 199$

to avoid hash collisions, and the notion of the critical vertex is not used since it has the maximum potential to find the best solution during the search operation.

4.4 Disruption management techniques

In this section, we describe the definition of benchmark solutions and various disruption management schemes for solving the disruptive dial-a-ride problem with a vehicle breakdown. Each of the below subsections represents the solution attained for DARP with and without fleet size minimization technique. First, *Benchmark* represents the solution attained for DARP before vehicle breakdown. Thereafter, in simulation or real-world, the vehicle fleet performs pickup and delivery of a sequence of passengers in the given benchmark routes and schedules. Second, *Method* represents the solution attained for DARP to address an event of a disruption, which is a vehicle breakdown. Afterward, the vehicle fleet resumes providing the transportation service to all the unserved passengers along the re-optimized routes and schedules.

4.4.1 Benchmark 1: Solution before breakdown / Benchmark solution

Benchmark 1 refers to the best-known DARP solution for the given problem instances. The best DARP solution refers to the planned routes and schedules with the minimum fleet travel cost.

4.4.2 Method A: Solution after breakdown/ No action taken

Once the vehicles start traveling in the pre-planned route (given by Benchmark-1), the disruptive vehicle break down event happens at a time t_{VB} for a vehicle k_{VB} from the fleet. In Method A, we do not take any action after the time of breakdown and simply let the other vehicles except k_{VB} to complete their already planned routing operations.

4.4.3 Method B: Conventional method (send a rented vehicle in predefined route)

Once the vehicle breaks down at a time t_{VB} for a vehicle k_{VB} from the fleet, some decision needs to be taken. In Method B, we rent an additional vehicle and dispatch in the already planned route, while simply letting the other vehicles except for k_{VB} to complete their already planned routing operations.

4.4.4 Method C: Reoptimized solution using G-ALNS with a rented vehicle

Once the vehicle breaks down at a time t_{VB} for a vehicle k_{VB} from the fleet, some decision needs to be taken. In Method C, we rent an additional vehicle and then re-optimize the routes and schedules of all the vehicles except k_{VB} for a limited run time before resuming the routing operations.

4.4.5 Benchmark 2: Solution before breakdown / Benchmark solution with fleet size minimization

Benchmark 2 refers to the best DARP solution attained by the G-ALNS with fleet size minimization (FSM) (otherwise, termed as G-ALNS-FSM) for the given problem instances. The best DARP solution refers to the planned routes and schedules with the least value of the objective function (prioritizing the vehicle minimization over the fleet travel cost).

4.4.6 Method D: Solution after breakdown/ No action taken

Once the vehicles start traveling in the pre-planned route (given by Benchmark-2), the disruptive vehicle break down event happens at a time t_{VB} for a vehicle k_{VB} from the fleet. In Method D, we do not take any action after the time of breakdown and simply let the other vehicles except k_{VB} to complete their already planned routing operations. We do not involve the saved fleets available at the depot.

4.4.7 Method E: Conventional method (send an available/rented vehicle in predefined route)

Once the vehicle breaks down at a time t_{VB} for a vehicle k_{VB} from the fleet, some decision needs to be taken. In Method E, we employ an additional available vehicle at the depot (resulted from the fleet size minimization scheme) or rent an additional vehicle if there is no available vehicle at the depot. Thereafter, we simply dispatch the additional vehicle in the already planned route, while simply letting the other vehicles except for k_{VB} to complete their already planned routing operations.

4.4.8 Method F: Reoptimized solution using G-ALNS-FSM with a rented vehicle

Once the vehicle breaks down at a time t_{VB} for a vehicle k_{VB} from the fleet, some decision needs to be taken. In Method F, we either employ an additional available vehicle at the depot (resulted from the fleet size minimization scheme) or a rented vehicle if there is no available vehicle at the depot. Thereafter, we re-optimize the routes and schedules of all the vehicles except k_{VB} for a limited run time before resuming the routing operations.

4.5 Simulation Results

In this section, we evaluate the performance of the proposed disruption management framework for the disruptive dial-a-ride problem with a vehicle breakdown. In Section 4.5.1, we describe the simulation settings. In Section 4.5.2, we introduce the model of operational cost function that is based on a realistic estimate of the current transportation operations. In Section 4.5.3, we compare the solution quality of the proposed disruption management scheme with the conventional schemes for DDARP-VB. In Section 4.5.4, we analyze the effectiveness of the proposed approach on passenger experiences for DARP, before and after vehicle breakdown. In Section 4.5.5, we analyze the policy implications of minimizing the fleet size on several key performance indicators for dial-a-ride operations.

4.5.1 Simulation settings

In this section, we describe the program development environment, DARP benchmark instances, and all the algorithmic configurations to perform the simulations.

The proposed G-ALNS algorithm for DARP with vehicle breakdown is implemented in CUDA. The CPU version of the ALNS algorithm is implemented in C++. All simulations are carried out on a computer running 2.1 GHz Intel Xeon E5-2620 v4 processor and Nvidia Tesla P100-PCI-E-16GB GPU, interconnected with a PCI-e bus. Visual Studio IDE 2015 with CUDA 8.0 toolkit is used as the development platform.

We test the proposed methodology on the standard DARP benchmark instances from R1a to R10a [7]. In these instances, the fleet size ranges from 3 to 13 and the corresponding number of requests ranges from 24 to 144. Each vehicle has a capacity of 6 seats with a maximum route duration of 480 min. Maximum user ride time and service time are 90 min and 10 min, respectively. The time window length for either pick-up or drop-off vertex is 15 min, and the other subsequent vertex is 1400 min.

To ensure a fair comparison, each experiment is conducted five times and the average values are reported. Tables 4.2 and 4.3 show the detailed solution for DDARP-VB with different breakdown times such as early, middle, and late of the service. Figures 4.2–4.11 show the comparison of the operational cost (in SGD), total number of customers served n^* , and total number of customer requests n when using various disruption schemes for the DDARP-VB. First, we compute the average over service starting time and ending time of all vehicles for the benchmark solution and then divide them into $m + 1$ slices to find the early, middle, and late stamps of time, respectively.

In Mu et al. (2011) [200] Disruptive VRP, for each problem, a time constraint of 12 sec has been applied. In Li et al. (2009), for disruptive VRP, 300 iterations for the rerouting problem are applied for delivery service.

We show the representation of methodologies in Table 4.1. We report the best solution out of five runs after 30 sec of execution time in Tables 4.2 and 4.3, average out of five runs after 30 sec in Table 4.4, best out of five runs after 120 sec in Tables 4.5, 4.6, and 4.7.

4.5.2 Operational cost function

In this section, we describe and model the proposed operational cost function which is based on a realistic estimate of the current transportation operations in Singapore.

The operational cost is one of the important parameters to assess the fleet operations, which often refers to the overall cost incurred by the service provider that include: vehicle procurement cost, licensing fees, fuel, insurance, driver wages, maintenance, repairs, parking, toll taxes, and all other mileage-dependent depreciation costs. To perform a realistic evaluation of solutions, we have considered the actual vehicle operational cost structure [206] observed in Singapore. The average operational cost for an owned and a rented vehicle in Singapore are 1620\$/month and 2880\$/month, respectively, excluding fuel cost being 0.24\$/km. The cleaning cost of a vehicle includes an average water wash cost of 15\$ and the indoor vacuuming cost of 10\$ per usage when considering a cost-effective cleaning agency based in Singapore. Since the dial-a-ride system involves passenger comfort, we set the cleaning cost to be the maximum value of 25\$ per day [207]. In Singapore, the parking cost of a vehicle is estimated to be 0.04\$ per min for parking space provided by the Housing Board and Development authority. The parking time in dial-a-ride service depends upon the amount of time that the vehicle is parked in a customer location waiting to perform pick-up or drop-off service [208]. For cars operating on Singapore roads, the average wear and tear costs are estimated to be around 300\$ per 10^4 kilometers of travel distance. Considering the upper bound, the average cost of 0.03\$ per kilometer is incurred by the car owners for the wear and tear. The strategy for renting a vehicle, in case of breakdown, involves a full day rental cost. The reported cost values are in Singapore dollars. We formulate the operational cost function in the following.

$$\alpha(x) = (\nu + \Delta)m + (\gamma + \zeta)\Phi(x) + \delta\Gamma(x) + \mu\Upsilon(x), \quad (4.5)$$

where m is the total number of vehicles owned by the fleet operator, ν is the maintenance cost in \$ per vehicle per day, Δ is the cleaning cost in \$ per vehicle per day, $\Phi(x)$ is the total distance traveled by the vehicles in km by both own and rental vehicles, ζ is the wear and tear cost is \$ per vehicle per km, γ is the petrol cost in \$ per vehicle per km, $\Gamma(x)$ is the number of additional vehicles added to

the fleet after the breakdown, δ is the rental cost in \$ per vehicle per day, μ is the parking cost in \$ per vehicle per min, and $\Upsilon(x)$ is the parking time of vehicles in min.

FIGURE 4.2: Case 1: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R1a Instance for DDARP-VB.

FIGURE 4.3: Case 2: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R2a Instance for DDARP-VB.

FIGURE 4.4: Case 3: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R3a Instance for DDARP-VB.

FIGURE 4.5: Case 4: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R4a Instance for DDARP-VB.

FIGURE 4.6: Case 5: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R5a Instance for DDARP-VB.

FIGURE 4.7: Case 6: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R6a Instance for DDARP-VB.

FIGURE 4.8: Case 7: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R7a Instance for DDARP-VB.

FIGURE 4.9: Case 8: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R8a Instance for DDARP-VB.

FIGURE 4.10: Case 9: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R9a Instance for DDARP-VB.

FIGURE 4.11: Case 10: Comparison of operational cost (in \$), number of customers served n^* , and number of customer requests n by various methods on R10a Instance for DDARP-VB.

4.5.3 Comparison of solution quality with the conventional method

In this section, we compare the solution quality of the proposed disruption management scheme with the conventional schemes for DDARP-VB.

The computational speedup is defined as the ratio of computational time in CPU to computational time in GPU. We have already shown (in Chapter 3) the significance of G-ALNS in terms of computational speedup when compared to its CPU counterpart and its superiority in attaining better solutions than other existing methods for various DARP instances. Table 4.1 represents the notation of various methodologies used in this section.

In Tables 4.2 and 4.3, we show the results for R1a instance attained before and after breakdown by methods such as conventional method (CM), G-ALNS and G-ALNS-FSM. Note that NA represents the solution when no recourse action has been taken in the event of a vehicle breakdown. In G-ALNS-FSM, we attain the benchmark solution before breakdown by incorporating a fleet size minimization scheme, whereas we do not include such a scheme in G-ALNS. In CM, we rent a

Notation	Description
G-ALNS	GPU-based adaptive large neighborhood search methodology
G-ALNS-FSM	G-ALNS with fleet size minimization strategy
CM	Conventional Method
Method A	Solution after breakdown/ No action taken
Method B	Conventional method (send a rented vehicle in predefined route)
Method C	Reoptimized solution using G-ALNS with a rented vehicle
Method D	Solution after breakdown/ No action taken
Method E	Conventional method (send an available/rented vehicle in predefined route)
Method F	Reoptimized solution using G-ALNS-FSM with an available/rented vehicle
For the methods A, B, and C, the initial route plan is given by benchmark 1. For the methods D, E and F, the initial route plan is given by benchmark 2. Benchmark 1 and 2 are the best solutions for the DARP with and without fleet size minimization.	

TABLE 4.1: Disruption Management: Representation of the methodologies.

new vehicle and dispatch it to follow the pre-planned route of the broken vehicle. In G-ALNS, we rent an extra vehicle apart from performing online optimization of routes after the disruption. In G-ALNS-FSM, we dispatch a new vehicle from the depot and dynamically optimize the routes after the disruption. For the solved instance, we have only two vehicles for the G-ALNS-FSM benchmark solution, and three vehicles for the G-ALNS benchmark solution. In Table II, the vehicles V1, V2, and V3 serve 4, 26, and 18 nodes, respectively. For the solution details, the

Case	Breakdown Time (min.)	Method	Before Breakdown			After Breakdown		
			n^*	$\Phi(x)$	$\alpha(x)$	n^*	$\Phi(x)$	$\alpha(x)$
Vehicle 1	Early (213.33)	NA	24	190.01	208\$	22	185.12	207\$
		CM	24	190.01	208\$	24	193.46	305\$
		G-ALNS	24	190.01	208\$	24	193.46	305\$
	Middle (286.77)	NA	24	190.01	208\$	24	190.01	208\$
		CM	24	190.01	208\$	24	190.01	208\$
		G-ALNS	24	190.01	208\$	24	190.01	208\$
	Late (360.20)	NA	24	190.01	208\$	24	190.01	208\$
		CM	24	190.01	208\$	24	190.01	208\$
		G-ALNS	24	190.01	208\$	24	190.01	208\$
Vehicle 2	Early (213.33)	NA	24	190.01	208\$	13	119.99	191\$
		CM	24	190.01	208\$	24	194.83	305\$
		G-ALNS	24	190.01	208\$	24	194.83	305\$
	Middle (286.76)	NA	24	190.01	208\$	16	129.17	194\$
		CM	24	190.01	208\$	24	192.50	304\$
		G-ALNS	24	190.01	208\$	24	192.50	304\$
	Late (360.20)	NA	24	190.01	208\$	18	144.86	197\$
		CM	24	190.01	208\$	24	200.35	306\$
		G-ALNS	24	190.01	208\$	24	200.35	306\$
Vehicle 3	Early (213.33)	NA	24	190.01	208\$	15	124.95	192\$
		CM	24	190.01	208\$	24	193.94	305\$
		G-ALNS	24	190.01	208\$	24	193.94	305\$
	Middle (286.76)	NA	24	190.01	208\$	19	143.99	197\$
		CM	24	190.01	208\$	23	194.40	305\$
		G-ALNS	24	190.01	208\$	24	201.03	306\$
	Late (360.20)	NA	24	190.01	208\$	18	151.18	198\$
		CM	24	190.01	208\$	24	195.66	305\$
		G-ALNS	24	190.01	208\$	24	195.66	305\$

$\beta_{avg}(x) = \mathbf{32.97\%}$, $\delta_{avg}(x) = \mathbf{36.32\%}$. Note: n^* indicate the average number of customers served in the best solution attained for the particular method over five independent runs.

TABLE 4.2: R1a Instance: Results attained by the G-ALNS for DDARP-VB. Best result out of five random runs are reported.

Case	Breakdown Time (min.)	Method	Before Breakdown			After Breakdown		
			n^*	$\Phi(x)$	$\alpha(x)$	n^*	$\Phi(x)$	$\alpha(x)$
Vehicle 1	Early (216.22)	NA	24	193.50	155\$	15	123.39	138\$
		CM	24	193.50	155\$	24	197.04	210\$
		G-ALNS-FSM	24	193.50	155\$	24	195.38	209\$
	Middle (315.48)	NA	24	193.50	155\$	19	147.48	198\$
		CM	24	193.50	155\$	23	197.88	210\$
		G-ALNS-FSM	24	193.50	155\$	24	197.69	209\$
	Late (414.75)	NA	24	193.50	155\$	20	168.77	148\$
		CM	24	193.50	155\$	24	197.87	210\$
		G-ALNS-FSM	24	193.50	155\$	24	197.87	210\$
Vehicle 2	Early (216.22)	NA	24	193.50	155\$	13	121.18	138\$
		CM	24	193.50	155\$	24	198.32	210\$
		G-ALNS-FSM	24	193.50	155\$	24	198.32	210\$
	Middle (315.48)	NA	24	193.50	155\$	18	138.40	141\$
		CM	24	193.50	155\$	23	195.99	209\$
		G-ALNS-FSM	24	193.50	155\$	24	200.56	210\$
	Late (414.75)	NA	24	193.50	155\$	19	154.28	145\$
		CM	24	193.50	155\$	23	197.87	209\$
		G-ALNS-FSM	24	193.50	155\$	24	204.89	211\$
$\beta_{avg}(x) = \mathbf{41.76\%}, \quad \delta_{avg}(x) = \mathbf{35.38\%}, \quad \eta_{1(avg)}(x) = \mathbf{25.48\%},$ $\eta_{2(avg)}(x) = \mathbf{25.49\%}.$								

TABLE 4.3: R1a Instance: Results attained by the G-ALNS with fleet size minimization (G-ALNS-FSM) for DDARP-VB. Best result out of five random runs are reported.

readers are referred to as the benchmark solution in [103]. In Table III, the vehicles V1 and V2 serve 22 and 26 nodes, respectively.

In Table 4.4, we perform five executions with randomly selected breakdown vehicle and time at which the vehicle breaks down. From the results, we can see that the numbers for total passengers feasibly served β and percentage cost-hike of rescheduled solution δ , when compared to benchmark solution, are in the favor of G-ALNS-FSM than that of G-ALNS. In this table, η_1 shows the percentage savings in the operational cost of benchmark solution attained by G-ALNS-FSM when compared to G-ALNS, and η_2 shows the percentage savings in operational cost of post-breakdown solution attained by G-ALNS-FSM when compared to G-ALNS. The average reduction in operational cost attained by G-ALNS-FSM, when compared to G-ALNS, corresponds to 17.38% before the breakdown and 14.98%

Case	Before breakdown				After breakdown			
	G-ALNS		G-ALNS-FSM		G-ALNS		G-ALNS-FSM	
	Total	Avg.	Total	Avg.	Total	Avg.	Total	Avg.
R1a	1094.99	45.62	1205.42	50.23	1186.57	49.44	1232.50	51.35
R2a	1929.19	40.19	2297.71	47.87	1937.94	40.37	2339.73	48.74
R3a	3505.72	48.69	3538.91	49.15	3469.18	48.18	3350.99	46.54
R4a	4997.11	52.05	4638.47	48.32	4976.65	51.84	4402.21	45.86
R5a	6002.53	50.02	5810.07	48.42	5971.77	49.76	5589.79	46.58
R6a	7643.21	53.08	6938.14	48.18	7061.58	49.04	6821.72	47.37
R7a	1508.93	41.91	1521.94	42.28	1456.10	40.45	1662.36	46.18
R8a	3692.65	51.29	3648.65	50.68	3802.11	52.81	3716.62	51.62
R9a	5222.37	48.36	5222.37	48.36	5262.71	48.73	5262.71	48.73
R10a	7560.65	52.50	7188.66	49.92	6987.08	48.52	7049.90	48.96
<i>Average</i>	4315.74		4201.03		4211.17		4142.85	
<i>Imp. (%)</i>			2.66				1.62	

TABLE 4.5: Influence of vehicle breakdown and effectiveness of the proposed algorithm on passenger experience. Comparison of user ride times for G-ALNS vs G-ALNS-FSM for DARP. Reduction in user ride time is achieved by our G-ALNS-FSM.

after the breakdown, respectively. Therefore, the usage of fleet size minimization is beneficial in reducing operational costs.

4.5.4 Effectiveness of the proposed approach on passenger experiences

In this section, we analyze the effectiveness of the proposed disruption management framework on passenger experiences for DARP, before and after vehicle breakdown.

In Table 4.5, we discuss the influence of vehicle breakdown and effectiveness of the proposed approach on passenger experiences. In passenger transit services, the user stress levels are significantly influenced by the commute time [209]. Empirical studies show that journey times affect the passenger’s quality of life. According to a psychological study [209], “the passenger stress effects in transit services seemed to be mediated by the time of the trip, i.e., reduced commuting time could account for some of the significant stress impacts of experimental alteration of the one

Case	v	m	m^*	Dist	T_{dur}	T_{idle}	T_{ride}	T_{early}	T_{late}	Δm	Δm (%)	Detour (%)
R1a	48	3	2	193.50	794.09	120.58	1205.42	33.15	1891.02	1	33.33	1.83
R2a	96	5	4	305.50	1674.77	409.27	2297.71	220.12	3205.71	1	20.00	1.38
R3a	144	7	6	542.98	2391.40	408.41	3538.91	137.54	5408.47	1	14.29	2.06
R4a	192	9	6	638.44	2690.66	132.22	4638.47	10.06	7067.35	3	33.33	11.96
R5a	240	11	8	688.52	3360.65	272.13	5810.07	89.83	9145.21	3	27.27	9.83
R6a	288	13	9	862.18	5175.94	290.28	6938.14	60.37	9730.66	4	30.77	9.80
R7a	72	4	3	302.97	1483.75	144.65	1521.94	84.09	2828.67	1	25.00	3.86
R8a	144	6	5	511.43	2323.25	124.03	3648.65	44.80	5253.71	1	16.67	4.84
R9a	216	8	8	658.31	3031.90	213.59	5222.37	33.95	8277.11	0	0.00	0.00
R10a	288	10	9	921.20	4542.06	343.02	7188.66	54.64	10666.18	1	10.00	8.15
Total				5625.04	27468.46	2458.17	42010.35	768.55	63474.09			

Where, v is the number of vertices; m is the number of available vehicles; m^* is the number of used vehicles, Dist is the total travel distance of all vehicles; T_{dur} is the total travel duration of all vehicles; T_{idle} is the total idle time of all vehicles; T_{ride} is the total ride time of all users; T_{early} is the total time for vehicle early arrival; T_{late} is the total time for vehicle late arrival; Δm is the number of vehicles reduced by our FSM strategy; $\Delta m(\%)$ is the vehicle reduction in percentage; Detour (%) is the total extra distance travelled by the fleet in percentage when compared to the benchmark solutions [5] without fleet size minimization.

TABLE 4.6: Details of the solution attained by G-ALNS-FSM on tested instances. G-ALNS-FSM attains significant fleet savings at a reasonable increase in detour distance when compared to the benchmark solutions [5] without fleet size minimization.

Quantifier	Savings (%)	
	G-TS-FSM	G-ALNS-FSM
Reduction in Fleet size	+21.05	+21.05
Reduction in Travel distance	-9.83	-6.22
Reduction in Total vehicle route duration	+10.11	+3.47
Reduction in Total vehicle idle time	+57.77	+58.19
Reduction in Total passenger ride time	+5.02	+2.66
Reduction in Total vehicle early arrival time	+69.34	+72.85
Reduction in Total vehicle late arrival time	-4.21	-1.44
Operational cost	+16.16	+40.06

TABLE 4.7: Policy implications: Average savings with fleet size minimization by G-ALNS and G-TS [6] when compared to the benchmark solutions [5] without fleet size minimization for all the tested DARP instances.

commuting route”. From our simulations, although the objective is not to increase passenger comfort, we have noticed that the passenger experience is improved by the reduction in commute time. When compared to the G-ALNS, the G-ALNS-FSM accounts for the reduction in commute time by 2.66% before the breakdown and 1.62% after the breakdown, respectively.

In Table 4.6, we show the detailed solution attained by the G-ALNS-FSM method on DARP instances.

4.5.5 Policy implications on the key performance indicators

In this section, we analyze the policy implication of minimizing the fleet size on several key performance indicators including the total reduction in fleet size, total travel distance, total vehicle route duration, total vehicle idle time, total user ride time, total vehicle early arrival time, and total vehicle late arrival time.

In Table 4.7, we analyze the policy implication of fleet size minimization if disruption does not occur. We compare the results of our G-ALNS-FSM and an existing GPU-based Tabu Search algorithm [6] with fleet size minimization (G-TS-FSM). Finally, the reductions in quantifiers are computed with respect to the benchmark solutions [5] without fleet size minimization. We are able to reduce the fleet size by

21.05% with a reasonable additional detour distance of 6.22%. Moreover, the fleet idle time goes down significantly while minimizing the number of vehicles. The fleet size minimization strategy is justified by the reduction in fleet idle time, fleet cost savings, reduced user commute time, and reduced operational costs under disruption. It is also worth noting that our G-ALNS-FSM achieves lower operational costs over G-TS-FSM.

We also analyze the influence of breakdown on fleet idle time. The results show that the proposed G-ALNS-FSM, when compared to G-ALNS, reduces the fleet idle time by 58.19% before the breakdown and 59.37% after the breakdown, respectively.

4.6 Discussion

Disruption in centralized fleet operations has been a big issue since the recovery cost is huge for the transportation industry. Despite its importance, this problem has never been addressed in the context of an advanced user-oriented form of mobility service that is becoming increasingly relevant in today's ride-sharing transportation. In this chapter, we focus on the disrupted dial-a-ride problem, in which vehicle breakdown may occur at any point of time during the transportation service. We have formulated the DDARP-VB and proposed a novel GPU-based adaptive large neighborhood search algorithm to solve this problem. The general idea of ALNS is the removal and reinsertion of requests based on their past performances. During the reinsertion of a request, there are many combinations of solution outcomes. Since the sequential evaluation of all these combinations is computationally expensive, we parallelize this operation with the usage of CUDA kernels. The results show that the proposed G-ALNS performs faster than its CPU counterpart while performing better than the GPU-based Tabu Search algorithm in a shorter time. The G-ALNS with fleet size minimization strategy attains solutions with a significant reduction in fleet cost, at the expense of a negligible increase in travel cost. With realistic estimations of operational cost, the G-ALNS-FSM attains lower operational cost compared to G-ALNS, when tested on various standard instances from the literature. Some potential future research directions include (1) extension of G-ALNS-FSM to emergency dial-a-ride systems, such as medical, fire and rescue services, (2) prediction techniques to tackle the disruptive DARP with

traffic congestion and investigate the influence of passenger comfort under congestion, (3) investigate the latest vehicular technologies, such as electric, hybrid and autonomous vehicles, in the context of demand-responsive transportation.

Chapter 5

The Dial-A-Ride Problem with Electric Autonomous Vehicles and Modern Recharging Infrastructures

Summary - The advent of “electric autonomous vehicles” has the potential to revolutionize the intelligent transportation systems, and especially useful in mobility-on-demand services to combat the issues such as global warming, fuel consumption, air pollution, and road safety.

The electric autonomous dial-a-ride problem (e-ADARP) entails designing routes and schedules for user-oriented transit service between specific origins and destinations while satisfying several constraints including time windows, ride-time, battery management, and recharge times. Despite the importance of EVs, the existing distribution of static charging stations (SCS) is either slender or disorganized, which causes range anxiety. The modern charging technology, mobile energy disseminator (MED), is energy-efficient and able to recharge EV on the move without causing any detours.

Therefore, in this chapter, we propose mobile energy dissemination for recharging autonomous electric vehicles (EV) in the dial-a-ride system. First, we formulate the electric autonomous dial-a-ride problem with MEDs. Second, we devise a GPU-based meta-heuristic approach for vehicle route optimization and a scheduling algorithm to efficiently schedule visits to recharging stations.

Simulations are conducted on benchmark and Uber instances, and the results are compared with existing approaches. Finally, the effectiveness of the proposed MED for the e-ADARP system is compared with the conventional SCS.

The preliminary results of this chapter are published as a conference paper in Ramesh et al. 2018b [115], and the journal version is submitted as Ramesh et al. 2020c [132] for publication in the IEEE Transactions on Intelligent Transportation Systems.

5.1 Introduction

Autonomous Mobility-on-Demand (AMoD), a key step toward sustainable urban mobility in the 21st-century [14], refers to a transformative and rapidly developing mode of transportation wherein self-driving vehicles transport passengers in a given environment [13]. These services often involve transportation of human passengers who place transportation requests with an origin and a destination location. The underlying optimization problem is usually modeled in terms of a dial-a-ride problem (DARP), which consists of designing routes to perform pickup and delivery of passengers while respecting the passenger convenience. DARP is an advanced user-oriented form of mobility service that is becoming increasingly relevant in today's public and commercial transit services, emergency transportation, ground, and seaport operations.

According to a survey in 2019 [210], there are more than 278 million fuel-powered cars running on the road in the U.S., with an average age of 11.8 years. These fuel-powered vehicles involved in transportation accounts for about 28.9 percent of total greenhouse gas emissions as reported by the U.S. environmental protection agency [211], which sums up to 1.9 billion tons of CO_2 annually [212]. Climate change is a major concern, and it is essential to reduce greenhouse gas emissions. In recent years, one of the primary focuses of the automobile industry has been to come up with alternative vehicular technologies such as electric vehicles to prevent carbon emissions and the use of fossil fuels [21]. According to a survey in 2019 [22], the estimated number of EVs will increase drastically in the future with the estimated decrease in manufacturing cost [23]. Due to the emergence of autonomous vehicles (AV), the future AMoD systems will depend on both AV and EV to fulfill user

transportation. The utilization of autonomous EVs for everyday activities would rise up, only if the charging infrastructure available is made abundant. However, the existing infrastructure to charge these vehicles is either slender or disorganized, and any changes would be very costly. Due to this problem of EV range anxiety, adopting them to on-demand transit systems is a challenging problem, especially in DARP, where user convenience is of supreme importance. Nevertheless, to promote the usage of autonomous EVs and achieve green transportation, it is necessary to integrate them into the daily transit systems.

FIGURE 5.1: EV-recharging mechanism: Station charging station (SCS)

FIGURE 5.2: EV-recharging mechanism: Battery swapping station (BSS)

FIGURE 5.3: EV-recharging mechanism: Mobile charging (MC)

FIGURE 5.4: EV-recharging mechanism: Mobile energy disseminator (MED)

In general, there are two types of EV charging methodologies. In type-I, an EV should move to the predefined charging location, which is a wired static charging station (SCS) or battery swapping station (BSS). In type-II, the electric charging carrier, which is either mobile charger (MC) or mobile energy disseminator (MED) moves to the requesting EV location. The advantages and drawbacks of each type of charging are summarized in Table 5.1. First, the SCS has been studied in the context of general vehicle routing problem (VRP) by Hiermann et al. 2019 [23], Schneider et al. 2014 [213], Keskin and Catay 2016 [214], Hiermann et al. 2016 [215], and in the context of DARP by Bongiovanni et al. 2019 [24]. The EV should visit the static charging station for wired recharging. Recharging time is generally long. Existing infrastructure is not easily accessible. Although it is possible to optimize the SCS locations, it would be quite costly. Second, BSS has been studied in the context of DARP by Masmoudi et al. 2018 [111]. The EV should visit the charging station to swap the current battery with a new one. Battery swapping time is far shorter compared to recharging time. However, the

Type	Mechanism	Range	Recharging time	Infrastructure changes	Drawbacks	Literature
Existing	SCS Static wired	Low	Long	Required	(a,b,d)	[23][213][214][215]
	BSS Battery swap	Low	Instantaneous	Required	(a,b,c)	[111][24]
	MC Static wired//wireless	Medium	Medium	Not required	(b,d)	[216]
Proposed	MED Dynamic wireless	High	Short	Not required	-	[25]

(a) Slender infrastructure and costly changes may be required,
(b) Routing service is interrupted during recharging process,
(c) Leakage, reliability issues, and problem of ownership,
(d) Large recharging time.

TABLE 5.1: Survey of EV charging methodologies in the context of vehicle routing problem (VRP).

main drawbacks are high capital cost, leakage, reliability issues, and the problem of ownership. Battery model changes depending on the type or manufacturer of the EV. Third, MC has been studied in VRP-context by Shaohua et al. 2018 [216]. EV should place a request to the service provider, upon which the MC will arrive at the requested location. A mobile charging vehicle carries a portable charging port for wired or wireless charging. EV should be at rest when it is being recharged by the mobile charger, which interrupts the vehicle routing service. Recharging time is generally medium. Fourth, MED has been studied by Kosmanos et al. 2018 [25]. A MED vehicle can dynamically recharge a moving EV through an inductive power transfer (IPT) mechanism. Cooperative adaptive cruise control (CACC) technology is used for MED-EV synchronization during the time period in which MED follows EV. For recharging, an EV should place a request to the MED service provider. MED follows EV in a predefined route to recharge on the move through dynamic wireless charging. The vehicle routing service is uninterrupted during the recharging process. A 10-minute charge by MED makes an EV to cover about 9 miles. In comparison, a public charging station provides electric cars with just 3.7 miles of range on a 10-minute charge. With a short recharging time of MED, an EV exhibits a higher driving range. Overall, the MED seems to be superior and suitable for on-demand transit, and it is also safer and more reliable as indicated by Ahmad et al. 2017 [217], a recent survey on EV charging schemes. All these EV-recharging methodologies are illustrated using Figures 5.1–5.4.

The DARP was first studied in 1971 [26], and its applications are reported from several cities such as Bologna [174], Copenhagen [175], and Milan [32]. Several exact methods have been proposed for the single-vehicle case [155][156][176], and multi-vehicle cases [177][28]. Exact methods take a long time to attain optimality. To reduce the running times, heuristics are introduced for different variants of DARP [137][175]. A heuristic for the transportation of disabled people with a homogeneous fleet of vehicles has been proposed in [160]. A DARP with the homogeneous fleet was solved by a tabu search in [7] and a variable neighborhood search in [9], and a DARP with transfer points was solved by adaptive large neighborhood search [77]. A DARP with heterogeneous users was addressed by methods such as evolutionary local search [12] and a hybrid genetic algorithm [103]. A GPU-based solution method for DARP was only found in [6]. For an overview of variants of DARP, we refer to [5], a recent survey in the literature. In the past decade, the focus has been shifted towards green vehicle routing problems. [218]

first studied electric vehicle routing, in which EVs can be recharged at customer locations to 80% of maximum battery level, and a fixed recharging time is considered. Electric vehicle routing problem with SCS and full-recharging policy are first considered by [213]. [214] studied the same problem of [213] while considering a partial-recharging strategy that allows variable recharging time at SCS. Partial recharge with no restrictions on minimal battery level at destination depot was considered by [219]. [215] extended the electric vehicle routing with a heterogeneous fleet of EVs. [111] has studied electric vehicle DARP with BSS, in which a non-linear energy consumption model was proposed.

The electric autonomous dial-a-ride problem (e-ADARP) was first studied by [24], in which both autonomy and electric vehicle aspects are considered. Since autonomous vehicles can operate on a non-stop schedule, it offers a reduction in dead-head miles. Linear recharging rate is considered because of AV's negligible load and cruise speed [220][221]. Static recharging stations, battery management, partial recharging, final minimum battery level, and no maximum route-duration constraints were considered for the problem. A branch-and-cut exact method was used as a solution method. Nevertheless, there are certain research gaps associated with the study of [24]. First, conventional SCS is considered, which has a slender infrastructure that may require costly changes and cause large detours to recharge EVs. Meanwhile, the modern EV charging technology such as MED is superior in terms of efficiency and suitability for DARP, when compared to SCS. Recently, [25] has studied MED in the context of the shortest path routing problem (SPRP) using the Dijkstra method. However, e-ADARP is very challenging when compared to SPRP due to the range of constraints associated with it. Second, the exact methods are not suitable in practice since not even a feasible solution is found under 120 min for some instances [24]. A recent survey on DARP [5] states that it is important to adapt the existing heuristics for the state-of-the-art DARP variants. Moreover, to achieve the computational speeds necessary, parallel computing by GPU is a promising direction that is currently underexplored in the literature [6].

The location of the static recharging stations can be optimized to reduce EV-range anxiety. To explore this idea, we developed a bi-level optimization framework to solve DARP with depot optimization. The findings in our conference paper in Ramesh et al. (2018b) [115] have shown the possible benefits of optimizing the location of static recharging stations for DARP. The bi-level optimization framework

method for DARP with depot optimization and the simulation results are discussed in Appendix D. However, any changes to the existing infrastructure might be very costly. Therefore, in this chapter, we propose mobile energy disseminators for the DARP with electric autonomous vehicles.

5.1.1 Our Contributions

In this section, we describe the main contributions of this chapter.

- We extend the general DARP formulation of Cordeau and Laporte (2003) [7] and e-ADARP formulation of Bongiovanni et al. [24] to model the operations of electric autonomous dial-a-ride problem with mobile energy disseminators (e-ADARP-MED) that includes new constraints related to the routing and scheduling of the mobile energy disseminators.
- In Chapter 3, we proposed a GPU-accelerated solution methodology (integrated with TS, VNS, and ALNS) for the general DARP. Since the ALNS seems to exhibit the highest solution quality for general DARPs, in this section, we extend the GPU-based ALNS method with new problem-specific mechanisms to solve the e-ADARP-MED. We introduce a new scheduling algorithm to evaluate the feasibility of a solution that reduces the excess user ride time, and this method can be used for scheduling visits to either static charging stations or mobile energy disseminators.
- We have conducted extensive simulations on adapted instances of general DARP and instances based on Uber technologies in San Francisco City [24]. We show the effectiveness of the proposed solution algorithm for e-ADARP in comparison to existing methods, while evaluating the effect of MED fleet size on solution quality. Furthermore, we analyze the effectiveness of the proposed mobile energy disseminators for DARP in terms of the operational quality and user experiences when compared to the static recharging infrastructure.

The remainder of the chapter is organized as follows. In Section 5.2, we introduce the formulation of electric autonomous dial-a-ride problem with mobile energy disseminators. In Section 5.3, we introduce the solution methodology and scheduling algorithm for the electric autonomous dial-a-ride problem. We present the simulation results in Section 5.4 and offer concluding remarks in Section 5.5.

5.2 Formulation of Electric Autonomous Dial-A-Ride Problem

In this section, we introduce the formulation of the electric autonomous dial-a-ride problem with mobile energy disseminators (e-ADARP-MED). Table 5.2 describes the terminology used in this chapter.

Notation	Description
$P = \{1, \dots, n\}$	Set of all pick up locations.
$D = \{n+1, \dots, 2n\}$	Set of all drop off locations.
$N = P \cup D$	Set of all pick up and drop off locations.
F	Location of depot for electric vehicle.
K	Set of available electric vehicles.
S	Set of all charging stations.
$V = F \cup P \cup D \cup S$	Set of all possible locations.
T_k	Maximum route duration of k^{th} vehicle.
A_i	Time at which a vehicle arrives at node i .
B_i	Time at which a vehicle begins its service at node i .
C_i	Time at which a vehicle ends its service at node i .
D_i	Time at which a vehicle departs from node i .
d_i	Service time of any vehicle at node i^{th} .
W_i^k	Duration for which vehicle k waits at node i before starting the service.
L	Total time spent by each of the passengers in a vehicle.
L_i^k	Time spent by the i^{th} customer in the k^{th} vehicle.
$c(x)$	Total distance traveled by all the vehicles to serve the customers.
Q_i^k	The number of passengers on the vehicle k after visiting node i .
q_i	Load available at the i^{th} node.
C_v^k	Capacity of the k^{th} vehicle.
e_i	Earliest time when service can begin at the i^{th} node.
l_i	Latest time beyond the i^{th} node cannot be serviced.
t^k	Current time of vehicle k .
x	A DARP solution or routes and schedules designed for the fleet.
$f(x)$	Objective function value of a DARP solution x .
x_{ij}^k	1 if arc (i, j) is traversed by vehicle k and 0 otherwise.
E_s^k	Recharge time of vehicle k at stations $s \in S$.
r	Final minimum battery level ratio of the EV k .
α_s	Recharge rate at the station s .
r_u	Final minimum battery level ratio of the MED vehicle k_u .
b_0^k	Initial battery level and effective battery capacity of the EV k .
$b_{u0}^{k_u}$	Initial battery level and effective battery capacity of the MED vehicle k_u .
u	A MED solution or routes and schedules designed for the fleet.
Z	Fleet size or total number of MED vehicles.
Y	Total number of customer nodes that can be visited by MED vehicle.
T_u	Time taken to connect the MED and EV before recharging process.
k_u	A MED vehicle from the fleet size Z .
$A_{ui}^{k_u}$	Time at which a MED vehicle k_u arrives at node i .
$B_{ui}^{k_u}$	Time at which a MED vehicle k_u begins its service at node i .
$C_{ui}^{k_u}$	Time at which a MED vehicle k_u ends its service at node i .
$D_{ui}^{k_u}$	Time at which a MED vehicle k_u departs from node i .

TABLE 5.2: Terminology.

In the e-ADARP [24], a fleet of m autonomous electric vehicles must serve n customer requests. The customer requests lead to n pickup and n drop off nodes. The nodes form a directed graph (N, A) graph where $N = P \cup D$, $P = \{1, \dots, n\}$

and $D = \{n + 1, \dots, 2n\}$. N denotes the number of vertices, and A denotes the number of edges. Nodes 0 and $2n + 1$ represent the origin and destination depots. Any arc (i, j) denotes a path from the i^{th} to the j^{th} node. The time to travel the arc (i, j) is t_{ij} and the cost of routing is c_{ij} . Each location $i \in V$ is characterized by the change in load p_i , which is positive at pickup locations, negative at dropoff locations, and is zero at all other locations. In addition, non-null service time d_i are defined at pickup and dropoff locations, which represent the time that the vehicles need in order to board users. The amount of energy E_s recharged at a charging station S is proportional to the time spent at the facilities. Each charging facility can only be accessed by empty vehicles and is characterized by a recharge rate α that indicates the amount of energy transferred per unit time. It is assumed that vehicles can partially recharge at the visited charging stations. Finally, the vehicle battery level cannot be lower than a certain minimum SOC upon arrival at the destination depot, which is controlled by the minimum battery level ratio r . The battery consumptions $\beta_{(i,j)}$ can be inferred from the travel times t_i , $j \in A$ combined with other features by means of an energy consumption model [220]. It is assumed that triangular inequality holds both for travel times and battery consumptions. A binary decision variable $x_{i,j}^k$ denotes whether vehicle k sequentially visits locations i and $j \in V$. The optimization problem (5.1) minimizes routing cost $f_1(x) = \sum_{(i,j) \in A, k \in m} x_{ij}^k t_{ij}$, and excess ride time $f_2(x) = \sum_{i \in n} (B_{i-} - D_{i+} - t_{i+i-})$, where w_1 and w_2 are weight factors. Note that the parameters related to charging stations, battery consumption, recharge rate, and final minimum battery level may be defined as vehicle-specific. Nevertheless, such parameters are assumed to be homogeneous since it is reasonable in the employment of autonomous electric mini-buses, as nowadays such vehicles mount the same commercial batteries and do not significantly differ in terms of dimensions and weights [24]. We formulate the proposed electric autonomous dial-a-ride system as follows.

$$\begin{aligned} \text{Min } f(x) &= w_1 \cdot f_1(x) + w_2 \cdot f_2(x), \\ \text{subject to,} \end{aligned} \tag{5.1}$$

$$\sum_{j \in N} x_{0j}^k = 1 \quad \forall k \in K, \quad (5.2)$$

$$\sum_{i \in D} x_{i,2n+1}^k = 1 \quad \forall k \in K, \quad (5.3)$$

$$\sum_{j \in N} x_{ji}^k - \sum_{j \in N} x_{ij}^k = 0 \quad \forall i \in P \cup D, k \in K, \quad (5.4)$$

$$\sum_{k \in K} \sum_{j \in N} x_{ij}^k = 1 \quad \forall i \in P \cup D, \quad (5.5)$$

$$\sum_{j \in N} x_{ij}^k - \sum_{j \in N} x_{n+i,j}^k = 0 \quad \forall i \in P, k \in K, \quad (5.6)$$

$$Q_j^k \geq (Q_i^k + q_j)x_{ij}^k \quad \forall i \in N, \forall j \in N, \forall k \in K, \quad (5.7)$$

$$\max(0, q_i) \leq Q_i^k \leq \min(C_{vk}, C_{vk} + q_i) \quad \forall i \in N, \forall k \in K \quad (5.8)$$

$$B_j^k \geq (B_i^k + d_i + t_{ij})x_{ij}^k \quad \forall i \in N, \forall j \in N, \forall k \in K, \quad (5.9)$$

$$e_i \leq B_i^k \leq l_i \quad \forall i \in N, \forall k \in K, \quad (5.10)$$

$$L_i^k = (B_{n+i}^k - (B_i^k + d_i))x_{ij}^k \quad \forall i \in P, \forall k \in K, \quad (5.11)$$

$$t_{i,n+i} \leq L_i^k \leq L \quad \forall i \in P, \forall k \in K, \quad (5.12)$$

$$B_{2n+1}^k - B_0^k \leq T_k \quad \forall k \in K, \quad (5.13)$$

$$L_i^k = 0 \quad \forall k \in K, i \in o^k F \cup S, \quad (5.14)$$

$$b_i^k = b_0^k \quad \forall k \in K, i \in o^k, \quad (5.15)$$

$$b_j^k = \min(0, b_i^k - \beta_{i,j}) \quad \forall k \in K, i, j \in V, \quad (5.16)$$

$$b_j^k = \max(b_0^k, b_i^k + \alpha_s E_s^k - \beta_{i,j}) \quad \forall k \in K, s \in S, \quad (5.17)$$

$$b_i^k \geq r b_0^k \quad \forall k \in K, i \in F, \quad (5.18)$$

$$\sum_{j \in N} u_{0j}^{k_u} = 1 \quad \forall k_u \in Z, \quad (5.19)$$

$$\sum_{i \in D} u_{i,2n+k_u}^{k_u} = 1 \quad \forall k_u \in Z, \quad (5.20)$$

$$\sum_{j \in N} u_{ji}^{k_u} - \sum_{j \in N} u_{ij}^{k_u} = 0 \quad \forall i \in P \cup D, k_u \in Z, \quad (5.21)$$

$$\sum_{k_u \in Z} \sum_{j \in N} u_{ij}^{k_u} = 1 \quad \forall i \in P \cup D, \quad (5.22)$$

$$A_i^k \leq B_{u_i}^{k_u} \leq D_i^k \quad \forall i \in N, k \in K, k_u \in Z, \quad (5.23)$$

$$b_{u_i}^{k_u} \geq r_u b_{u_0}^{k_u} \quad \forall i \in N, k_u \in Z. \quad (5.24)$$

In the objective function $f(x)$ (in equation 5.1), the first term $f_1(x)$ is the total routing cost (i.e., fleet travel 'time') and the second term $f_2(x)$ is the total excess user ride 'time'. Both terms are in the units of time. According to [24], the values of weights, w_1 and w_2 , are set to 0.75 and 0.25, respectively, to maintain a balance between the two objective terms during the optimization process. Specifically, the minimization of the objective term $f_1(x)$ is given a higher priority over the objective term $f_2(x)$, in order to reduce the travel cost/time (reflected by lower values of $f_1(x)$) while increasing the customer satisfaction (reflected by lower values of $f_2(x)$).

The constraints specified in Eqs. (5.2)–(5.3) ensure that all the vehicle routes start and end at the depot. Eqs. (5.4)–(5.5) makes it mandatory for every node to be visited by only one vehicle such that any vehicle reaching a particular node must exit it. The customers that are picked up are always delivered to their corresponding drop-off locations as guaranteed by constraint in Eq. (5.6). Additionally, every vehicle must not exceed its maximum capacity and maximum ride time constraints must hold for the vehicles and the customers. Eq. (5.7) updates the load of the k^{th} vehicle once it reaches node j from node i . The constraint in Eq. (5.8) ensures that the capacity of vehicle k is not exceeded at any of the nodes. The time consistency is maintained by Eq. (5.9) which ensures that node j can be serviced only after it is reached. Eq. (5.10) specifies the time window of the service at each location and makes sure that service begins only within the windows. Vehicle k departs from node i as soon as the service ends at that node. Constraints (5.11)–(5.12) guarantee that the maximum ride time of a customer is not exceeded. Finally, the constraint on the route duration of any vehicle k is ensured by Eq. (5.13). Constraint (5.14) ensures that vehicles are empty at depots and charging stations. Constraint (5.15) sets initial battery levels for vehicles at origin depots. Constraint (5.16) sets the battery level state from any location i to j . Constraint (5.17) sets upper bounds on the battery level states at nodes. Constraint (5.18) imposes minimum battery levels for all vehicles returning to the depots. The above constraints are in line with [24].

Furthermore, we propose the following new constraints associated with MED. Constraints (5.19)–(5.20) ensure that all MED vehicles start and end at their respective stations. Constraints (5.21)–(5.22) make it mandatory for every node to be visited by only one MED vehicle such that any MED reaching a particular node must

exit it. Constraint (5.23) ensures that the recharging process is initiated by MED $k_u \in Z$ at the requested location $i \in N$ only after the EV arrival time A_i and before the EV departure time D_i . Constraint (5.24) imposes minimum battery levels for all MEDs returning to the station.

5.3 Solution Methodology and Scheduling heuristic

We address a different variant of DARP in each chapter, and subsequently, devise modifications & improvements to the GPU-based solution method. In terms of solution method alone, in Chapter 3, we introduce a generic GPU-based solution method to solve general DARP, while in Chapter 4, we introduce a new fleet management strategy and an online optimization framework into the GPU-based solution method to solve disruptive DARP, and in this Chapter 5, we introduce a new scheduling algorithm and a charging-station insertion strategy into the GPU-based solution method to solve electric autonomous DARP.

In this section, we introduce the solution methodology and scheduling algorithm for the electric autonomous dial-a-ride problem with mobile energy disseminators. In Section 5.3.1, we present the overall framework and route optimization algorithm for e-ADARP. In Section 5.3.2, we present the scheduling algorithm to evaluate the vehicle routes (designed during the optimization process) for e-ADARP.

To solve e-ADARP, we iteratively explore the neighborhood solutions of an initial solution until a stopping criterion is met. *e-ADARP instance* will be the input. *Best solution found* x_{best} is the output, which must be both feasible and should have the least value of objective function $f(x_{best})$ value among all the explored solutions. Firstly, an *Initial solution* $x_{initial}$ for the given problem instance will be constructed and it will be set as the *Current solution* $x_{current}$, which is the incumbent solution at any given iteration. Then, the process of iteratively exploring the neighborhood of a $x_{current}$ is performed by GPU. The ways of generating *neighborhood solutions* from the $x_{current}$, and acceptance criterion to select the best solution iteratively are based on ALNS heuristic and will be performed in CPU. For search diversification, the acceptance criteria may allow infeasible solutions to be selected as $x_{current}$. However, x_{best} must be always feasible and updated after every iteration.

In the following, we describe the neighborhood exploration, implementation details of the optimization algorithm, and scheduling heuristic.

The process of moving a customer from one vehicle into another is referred to as *neighborhood move*. All possible solutions resulting from this moving of a customer into each available vehicle is called as *neighborhood solutions*. For computational efficiency, the neighborhood solutions are divided into batches that are executed concurrently in GPU. Each batch evaluates the process of moving a unique request into all vehicles. Inside each batch, the GPU threads simultaneously evaluate the process of moving a unique request into a unique vehicle. The choice of which customer requests need to be relocated is chosen by ALNS heuristic. Algorithm 1 describes the implementation of a GPU-based neighborhood exploration function, which is based on our GPU framework discussed in Chapter 3.

5.3.1 Proposed Solution Framework

In this section, we introduce the route optimization method that is based on GPU-accelerated adaptive large neighborhood search algorithm for solving e-ADARP.

The general idea of LNS by Shaw (1998) [195] is to repeatedly remove requests from the solution and to reinsert them at more profitable positions. Later, an adaptive weight adjustment scheme to LNS was introduced by Ropke et al. (2006) [164] to develop ALNS for vehicle routing problems with time-windows. In this section, we extend this ALNS approach with GPU-acceleration for the route optimization of the electric autonomous dial-a-ride problem. The idea of exploring neighborhoods using GPU for DARP was introduced in Chapter 3. In this section, we implement the GPU-based Adaptive Large Neighborhood Search (G-ALNS) algorithm to solve the e-ADARP.

Algorithm 16 details the proposed G-ALNS framework, which is based on the design presented in Chapter 3. The algorithm starts with an initial solution. For each iteration, a pair of destroy and repair ALNS operators are adaptively selected from a pool of operators using a roulette wheel mechanism [164]. First, a destroy operator is applied to remove some requests from the current solution. Then, a repair operator explores the neighborhood using the CUDA kernels (Chapter 3) in GPU and selects an appropriate solution in CPU based on simulated annealing

Algorithm 16 G-ALNS optimization framework for e-ADARP

Require: e-ADARP Instance

- 1: Allocation of GPU resources and copy e-ADARP instance from CPU to GPU.
 - 2: Construct an *Initial solution* in CPU and set it as the *Current solution* in CPU
 - 3: **repeat**
 - 4: Update *req_list* based on ALNS in CPU
 - 5: Copy $x_{current}$ from CPU to GPU
 - 6: **Explore neighborhood in GPU** i.e. launch *Neighborhood.Exploration()*
 - 7: Update the overall cost of each neighborhood solution in CPU
 - 8: Select best solution among the neighborhood solutions based on ALNS in CPU
 - 9: Select the optional origin and destination depot (if applicable)
 - 10: Perform SCS or MED insertions into the the best solution
 - 11: Set the newly selected solution as *Current solution* in CPU
 - 12: Update $f(x_{best})$ of the *Best solution found* in CPU
 - 13: **if** No improvement in $f(x_{current})$ for a predefined number of iterations **then**
 - 14: Set the x_{best} as the $x_{current}$
 - 15: **end if**
 - 16: **until** stopping criterion
 - 17: Release GPU resources and **return** *Best solution found*
-

acceptance criterion. The weights of ALNS operators are adjusted after every segment according to predefined rules. For a detailed description of ALNS heuristic, the readers are referred to the work of Ropke et al. 2006 [164]. We implemented the following neighborhood search operators (destroy and repair operators) for ALNS.

- Destroy operators

- *Worst removal*: This operator iteratively removes requests that cause the biggest detour in the current solution.
- *Random removal*: This operator randomly selects the requests to be removed among the planned requests.
- *Related removal (or Shaw removal)*: This operator aims to remove requests that are alike so that their positions of service can somehow be interchanged.

- Repair operators

- *Best insertion*: At each iteration, the best insertion cost is computed for each unplanned request and the request with the lowest insertion cost is

inserted at its best position. The heuristic stops when all requests are routed or none can be inserted anymore.

- *Regret insertion:* This heuristic is based on the notion of regret, which is used by Potvin [196] for the Vehicle Routing Problem with Time Windows (VRPTW) and extended by Ropke [164]. It investigates the cost of inserting a request on the k best routes, and finally inserts it on a route whose cost difference between inserting it into the best route and the $k-1$ best routes is largest. In our implementation, we consider regret- k heuristics with values of k between 1 and 5.

During the search process, if the new best solution is not battery-feasible, we schedule visits with either SCS or MED using the layer-3 of the proposed scheduling heuristic (Section 17) by reinserting all charging stations into the best possible positions in the solution. If multiple solutions have the same value of $f(x)$, then the solution that has the least MED fleet travel time is selected. To schedule the MED routes, layer-1 of the scheduling heuristic (Section 17) is used. We employ a restart mechanism, in which, if the $x_{current}$ does not improve for more than 200 iterations, we set x_{best} as $x_{current}$ and restart the search for intensification. We use the best-improvement strategy since complete neighborhood exploration has the maximum potential to find the best solution during the search [194], and GPU-computation accelerates the compute-intensive neighborhood exploration process [6].

After the local search process, the optional origin and destination depot are selected heuristically. The optional origin depots are assigned in the order of the vehicle. For a random order of the fleet, the best possible optional destination depot with the shortest route deviation will be assigned one-by-one for each vehicle. This process is repeated five times for several random orders of the fleet, and the best set of optional destination depots are chosen.

Algorithm 16 describes the implementation of our GPU-based ALNS method for DARP. The procedure can be described as follows: 1) allocate device resources, 2) construct initial solution (as per Section 3.7) and set it as $x_{current}$, 3) copy $x_{current}$ from CPU to GPU, 4) construct and evaluate *neighborhood solutions* in GPU, 5) selectively copy data from GPU to CPU, 6) trigger restart mechanism in CPU if required, 7) record x_{best} after inserting SCS or MED in the best possible positions,

8) repeat steps 3-7 until the stopping criterion is satisfied, 9) print x_{best} , and release memory resources.

To solve generic DARP with a single objective function, the scheduling heuristic of Cordeau and Laporte (2003) [7], which is based on the concept of the forward slack time [193], has been exploited extensively in the literature. This procedure aims at generating a feasible schedule by minimizing violations of time windows, maximum route duration, and maximum user ride time. In contrast, for e-ADARP, user excess ride time $f_2(x)$ needs to be minimized as well in the weighted objective function in Eq. 5.1. Bongiovanni and Geroliminis (2019) [24] used an exact procedure to solve this problem. In the following, we propose a heuristic procedure to evaluate the solution of e-ADARP.

Let $q(x)$, $d(x)$, $w(x)$, $t(x)$ and $b(x)$ denote the total violations of load, route duration, time window, ride time constraints, and battery, respectively.

$$q(x) = \sum_{\forall k} \max(q_{k,max} - Q_k, 0), \quad (5.25)$$

$$d(x) = \sum_{\forall k} \max(d_k - T_k, 0), \quad (5.26)$$

$$w(x) = \sum_{\forall i} [\max(B_i - l_i, 0) + \max(B_{i+n} - l_{i+n}, 0)], \quad (5.27)$$

$$t(x) = \sum_{\forall i} \max(L_i - L, 0), \quad (5.28)$$

$$b(x) = \sum_{\forall k} \min((-1) * \sum_{\forall i} b_i^k, 0). \quad (5.29)$$

Infeasible solutions are allowed during the optimization through a relaxation mechanism [7]. Solutions are evaluated by a modified objective function $f'(x) = f(x) + \alpha * q(x) + \beta * d(x) + \gamma * w(x) + \phi * t(x) + \Gamma * b(x)$, where $\alpha, \beta, \gamma, \phi, \Gamma$ are self-adjusting positive parameters. During every iteration, these parameters are dynamically increased and decreased by a factor of $(1 + \delta)$, if the current incumbent solution is feasible and infeasible, respectively. A e-ADARP solution is feasible if it does not violate any given constraints in the formulation.

5.3.2 Proposed Scheduling Algorithm

In this section, we introduce the scheduling algorithm to evaluate the schedules for the vehicle routes (designed during the optimization process in Section 5.3.1) for e-ADARP. Algorithm 17 describes the implementation of the proposed scheduling algorithm, which is based on the following three layers.

Algorithm 17 Scheduling heuristic

- 1: Set $D_0 \leftarrow e_0$
 - 2: Compute A_i, w_i, B_i, D_i, y_i for each node i in route k
If some $B_i > l_i$ or $y_i > Q$, goto step 8
 - 3: Compute F_0
 - 4: Set $D_0 = e_0 + \min(F_0, \sum_{0 < p < q} w_p)$
 - 5: Update A_i, w_i, B_i and D_i
 - 6: Compute all T_i^{ride}
If all $T_i^{ride} < T^{ride}$ goto step 8
 - 7: For every node j that comes after the zero split node $i_z \in D$
 - (a) Compute the feasible shift in waiting time W_{min}^{Slack} between i_z and i_{z+1} .
 - (b) Set $w_j := w_j + W_{min}^{Slack}$; Update A_i, w_i, B_i, D_i and T_i^{ride} for each node i that comes after j in the route.
 - (c) If all $T_i^{ride} \leq T^{ride}$ for all i whose destinations lie after j , return true
 - 8: Return false
-

5.3.2.1 Layer 1

The service at each node $i \in N$ is started as early as possible, i.e., $\min(A_i^k, e_i)$, in the order in which the vehicle $k \in K$ visits the node. After visiting any node $i \in N$, the vehicle departs as early as possible, i.e., $D_i = B_i + d_i$, upon which the waiting time is computed as $W_i = B_i - A_i$. This layer minimizes the time window violations $w(x_k)$. Finally, the departure time from the depot is delayed as described in [7], to minimize the route duration violations $d(x_k)$.

5.3.2.2 Layer 2

Zero split nodes $i_z \in D$ are the nodes $i \in N$ at which the vehicle $k \in K$ is empty. In the existing procedure of [7], the idea is to delay the beginning of service at all pickup nodes $i \in P$ based on [193] in the hope of reducing the ride time associated with the requests. Therefore, the waiting times w_i associated with all nodes $j \in N$

between any pickup node $i \in P$ and the last visited node $l \in N$ are feasibly shifted into the node $i \in P$ without increasing the violation of $w(x_k)$. The above process is sequentially performed for all the nodes $i \in P$. In contrast, in our approach at layer-2, the waiting times w_i associated with nodes $j \in N$ between a zero-split node $i_z \in D$ and the immediate next zero split node $i_{z+1} \in D$ are feasibly shifted into the node $i_z + 1 \in P$ without increasing the violation of $w(x_k)$. W_{min}^{Slack} is calculated according to [193], followed by the update of A_i, B_i , and w_i of all nodes $i \in N$ that comes after node $i_z \in D$. It is followed by another shifting of waiting times associated between node $i_{z+1} \in D$ and $i_{z+2} \in D$ into the node $i_{z+1} + 1 \in P$. This process is repetitively performed in the order of zero split nodes visited by a vehicle $k \in K$, which minimizes the customer excess ride time besides reducing the user ride time. The layers 1&2 are used to evaluate any solution in the Algorithm 12.

5.3.2.3 Layer 3

Since e-ADARP allows recharging only when the vehicle is empty, the main objective of this layer is to maximize the waiting time of a vehicle at the node that comes after the zero split nodes $i_z \in D$ after which an SCS or MED is scheduled for a visit. Following is the procedure. Before the insertion SCS or MED after a node $i_z \in D$ in a vehicle $k \in K$, the waiting times associated with all the subsequent nodes $j \in N$ are feasibly shifted into the node $i_z + 1 \in P$ such that time window constraints are not violated and user excess ride time remains unchanged. In other words, W_{min}^{Slack} is calculated between node $i_z + 1 \in P$ and the destination depot. W_{min}^{Slack} is later added to the w_{i_z+1} , followed by the update of A_i, B_i , and w_i of all nodes $i \in N$ that comes after the node $i_z + 1 \in P$. This directly maximizes the possible amount of recharging time, thereby enabling the EV to recharge as much as possible. The same procedure is followed for any number of SCS or MED insertions into a route. To reduce computational complexity, this layer is used only if the best solution x_{best} is not battery feasible during the optimization in Section 5.3.1.

5.3.3 Recharging Infrastructures

In this section, we describe the procedure for electric autonomous vehicles to visit various recharging infrastructures such as static charging stations (Section 5.3.3.1) or mobile energy disseminators (Section 5.3.3.2) during the dial-a-ride operations.

5.3.3.1 Static Charging Station

The static charging station (SCS) denotes a fixed recharging infrastructure located at a static location. There may be multiple recharging stations located at various parts of the transportation network. To perform the recharging process, the battery-operated electric vehicle (EV) should go to the location of SCS and then wait at the SCS location while buying the electric power from the recharging station.

5.3.3.2 Mobile Energy Disseminators

The mobile energy dissemination (MED) denotes a movable recharging infrastructure. A MED vehicle can dynamically recharge a moving EV through an inductive power transfer (IPT) mechanism. Cooperative adaptive cruise control (CACC) technology is used for MED-EV synchronization during the time period in which MED follows EV. For the recharging process, the electric vehicle should place a request to the MED service provider. Upon the availability of MED, the MED vehicle reaches an EV at a requested location. Then, the MED follows EV in a predefined route to perform recharging on the move through a dynamic wireless charging mechanism. Therefore, the vehicle routing service is uninterrupted during the recharging process.

5.4 Simulation Results

In this section, we evaluate the performance of the proposed solution framework for e-ADARP and analyze computational performance, solution quality, and passenger experiences when using various recharging technologies. In Section 5.4.1, we describe the simulation settings and e-ADARP instances. In Section 5.4.2, we

analyze the computational performance and solution quality of the proposed approach when compared to the existing methods for e-ADARP. In Section 5.4.3, we analyze the effect of changing the MED fleet size on solution quality for e-ADARP. In Section 5.4.4, we analyze the effectiveness of the proposed solution framework and recharging infrastructures on passenger experiences for e-ADARP.

5.4.1 Simulation settings

In this section, we describe the program development environment, e-ADARP benchmark instances, and the real-world Uber taxi trips in San Francisco City, USA.

All the algorithms are implemented in CUDA C and C++ language. Simulations are carried out on a computer running 2.1 GHz Intel Xeon E5-2620 v4 processor and NVidia Tesla P100-PCI-E-16GB GPU, interconnected with a PCI-e controller. Visual studio 2015 with CUDA 8.0 toolkit in Windows 8.1 is the integrated development environment.

In the **benchmark instances** of e-ADARP [24], the number of customer nodes ranges from 32 to 80, and the number of autonomous electric vehicles ranges from 2 to 5. Each request is randomly generated in the Euclidean space of $[-10, 10]^2$. The time window length is 15 min. Each instance has a different geographical distribution of pick-up and drop-off locations. The service time for each vehicle at any location is set as 3 min. The seating capacity of each vehicle is 3, and the maximum passenger ride time limit is 30 min. All vehicles are battery-powered and travel at a constant speed in the road network. For all the cases, the total number of SCS is 3, which are located at $(0, 0)$, $(4, 4)$, and $(-4, -4)$ in the road network. Recharge and discharge rates are both set to be 0.055 kWh per minute, while the effective battery capacity and initial battery charge are both set to 14.85 kWh, as given by the autonomous-electric shuttle's manufacturer Navya [24]. All vehicles need to return to the destination depot with at least 10%, 40% and 70% battery ratio levels (i.e. $r=0.1$, $r=0.4$, $r=0.7$). Each vehicle can visit a maximum of one SCS. The weight factors (w_1, w_2) in the objective function are set as $\{0.75, 0.25\}$. All vehicles are assumed to start with full battery capacity and that the amount of battery discharged is proportional to the vehicle travel time.

The **Uber instances** are produced by extrapolating customer locations and times available in the ride-sharing GPS logs of the San Francisco City, USA, provided by Uber Technologies Inc. in 2011. This dataset contains 1.2 million GPS logs, which are updated by the active Uber cars every 4 seconds for one week. The pickup and dropoff locations were extracted from the data after removing the invalid records. The raw dataset is obtained from the GitHub repository at <https://github.com/dima42/uber-gps-analysis/tree/master/gpsdata>. The Uber instances used in this study focus on ride-sharing requests in the Downtown/Civic Center districts. Neighborhood information was obtained through the San Francisco Enterprise GIS Program (SFGIS) on the SF OpenData website. The weekend requests were aggregated into a single day, and the instances were generated by randomly sampling customer requests at the rate of one request per 3 minutes. OpenStreetMap (OSM) server was used to extract the San Francisco Transportation Network. The information about the electric vehicle static charging stations is obtained from the Alternative Fuels Data Center (AFDC) of the U.S. Department of Energy. The road travel times between all the nodes in the transportation network was calculated using Dijkstra's shortest path algorithm. The vehicles are assumed to travel at a constant speed of 35km/h. In the envisioned application AVs can provide non-stop ride-sharing services. In these instances, each vehicle may visit a maximum of five charging stations (information provided by the AFDC) located at different places and may return to any of the 5 optional destination depots. The difference between the benchmark and Uber instances consists of the following: (1) the pickups and drop-offs can appear anywhere in benchmark instances, whereas they can only appear on the transportation network in Uber instances; (2) the time windows in benchmark instances are tighter than the time windows in Uber instances; (3) there is more overlap between the time windows of the same customer request in benchmark instances than in Uber instances; and (4) the overlap between the time windows of multiple customers tend to be higher in benchmark instances than in Uber instances, i.e., the customer arrival rate is relatively higher in the Uber instances. To scale up the region of study, the road travel times are multiplied by a factor of 2, while increasing the battery consumption by 30%, and reducing the effective battery capacity to 3.5 kWh. Due to this transformation, each vehicle can visit at most approximately 15 nodes, with full effective battery capacity. The vehicle departs from the origin depot with full battery capacity. The customer maximum ride time limit is set to eight minutes and the user drop-off time window is set to

15 min. Both the benchmark and Uber instances are provided by Bongiovanni et al. 2019 [24].

As discussed above, these instances are produced and calibrated according to the information available in the ride-sharing GPS logs of the San Francisco City, USA, provided by Uber Technologies Inc. In terms of the choices of service time and passenger ride time limit, these authors follow the practices of [7][28], which are set in accordance with the practices of several North American transporters. In terms of the choice of the number of EVs, we observe that these authors considered small-scale instances since they introduce an exact method to solve e-ADARP. It can be noted that exact methods guarantee optimality, but consume large execution time, therefore applicable only in small-scale instances. Since our objective is to directly compare the effect of our optimization algorithm and charging infrastructures against existing algorithms on publicly available instances, we did not modify the instance characteristics to ensure a fair comparison with existing results.

For consistency with the instances of [24] while solving e-ADARP with MED, we consider 3 MEDs with the same recharging/dischARGE rate, each at the respective SCS locations. Each MED is restricted to recharge one EV, although it can recharge up to 4 EVs without returning to its starting location or depot to recharge itself [25]. A MED can follow an EV between any two prescheduled nodes $i \in N$ and return to the starting location S upon completion of the service. Therefore, the MED-related variables are set as $Z=3$, $Y=2$, $T_u=0$, and $r_u=0$. To test the computational efficiency of the proposed approach on a bigger scale, a set of four instances such as LSI1, LSI2, LSI3, LSI4 is generated. The number of nodes is 336, 480, 720, 1008, and the number of vehicles is 16, 22, 33, 45, respectively. Other operational constraints are set like standard test instances. To ensure a fair comparison, the initial solution of both CPU and GPU-based method is generated by the random insertion of unserved customers into the available vehicles.

5.4.2 Comparison of solution quality with the existing results

In this section, we analyze the computational performance of the proposed solution method for e-ADARP. Furthermore, we analyze the solution quality attained by

the proposed GPU-based metaheuristic approach for e-ADARP (with SCS/MED) when compared to the state-of-the-art results.

In Tables 5.4 and 5.5, we compare the solution quality achieved by our approach with the existing approaches for adapted DARP instances and problems based on taxi trips of San Francisco City, USA. LB represents the lower bound of each instance provided by [24], however, the execution times are not reported. gap is defined as the percentage difference between LB and $f(x_{best})$ attained by the proposed solution method. gap^* represents the best gap, and $\bar{gap}$ denotes the average value of the gap over five executions. t denotes the computational time in min with the stopping criterion set as 2 min. The state-of-the-art results for the e-ADARP are reported by the branch-and-cut method of [24]. We show the computational time (in Chapter 3) attained by the solution method in CPU and GPU. The results show that the proposed solution method attains new LB for several e-ADARP benchmark instances when using MED in comparison with SCS. Furthermore, because of the proposed scheduling heuristic, our solution with SCS seems to attain better solutions than LB for a few instances under 2 min, while the existing approaches could not even find feasible solutions under 120 min. Figures 5.5–5.8 depicts the best route plan attained for a2-16-0.7 e-ADARP instance for illustration when using SCS or MED or no charging station. Clearly, we attain a better feasible solution quality when MEDs are employed for e-ADARP.

FIGURE 5.5: Best route plan for a2-16-0.7 e-ADARP instance.

Case	<i>nodes</i>	<i>m</i>	CPU (ms)	GPU (ms)	Speedup	Difficulty
a2-16- <i>r</i>	32	2	1038	156	6.64	1038
a2-20- <i>r</i>	40	2	2274	235	9.69	2274
a2-24- <i>r</i>	48	2	6057	292	20.77	6057
a3-18- <i>r</i>	36	3	696	149	4.66	696
a3-24- <i>r</i>	48	3	2435	232	10.50	2435
a3-30- <i>r</i>	60	3	5397	267	20.23	5397
a3-36- <i>r</i>	72	3	7355	332	22.17	7355
a4-16- <i>r</i>	32	4	417	125	3.33	417
a4-24- <i>r</i>	48	4	1227	168	7.29	1227
a4-32- <i>r</i>	64	4	4421	204	21.71	4421
a4-40- <i>r</i>	80	4	5882	268	21.93	5882
a5-40- <i>r</i>	80	5	6438	278	23.16	6438
LSI1	336	16	237309	8714	27.23	237309
LSI2	480	22	513598	16408	31.30	513598
LSI3	720	33	1861300	45195	41.18	1861300
LSI4	1008	45	2171500	83298	26.07	2171500

TABLE 5.3: Computational speedup attained by ALNS on GPU vs CPU for e-ADARP. Stopping criterion is 10^3 iterations. The value of r refers to the final minimum battery level ratio.

FIGURE 5.6: Best route plan with SCS visits for a2-16-0.7 e-ADARP instance.

FIGURE 5.7: Best route plan with MED visits for a2-16-0.7 e-ADARP instance.

FIGURE 5.8: Legends.

Case	Size	m	LB	B&C		P (with SCS)			P (with MED)		
				<i>gap*</i>	<i>t</i>	<i>gāp</i>	<i>gap*</i>	<i>t</i>	<i>gāp</i>	<i>gap*</i>	<i>t</i>
a2-16-0.1	32	2	237.38	0.00	0.02	0.00	0.00	0.01	0.00	0.00	0.01
a2-20-0.1	40	2	279.08	0.00	0.07	0.00	0.00	0.01	0.00	0.00	0.01
a2-24-0.1	48	2	346.21	0.00	0.15	0.00	0.00	0.02	0.00	0.00	0.02
a3-18-0.1	36	3	236.81	0.00	0.08	0.00	0.00	0.00	0.00	0.00	0.00
a3-24-0.1	48	3	274.80	0.00	0.23	0.00	0.00	0.02	0.00	0.00	0.02
a3-30-0.1	60	3	413.27	0.00	1.70	0.00	0.00	0.11	0.00	0.00	0.11
a3-36-0.1	72	3	481.72	0.00	1.78	0.00	0.00	0.05	0.00	0.00	0.05
a4-16-0.1	32	4	222.49	0.00	0.06	0.00	0.00	0.02	0.00	0.00	0.02
a4-24-0.1	48	4	310.84	0.00	0.52	0.00	0.00	0.09	0.00	0.00	0.09
a4-32-0.1	64	4	393.96	0.00	10.20	0.00	0.00	0.17	0.00	0.00	0.17
a4-40-0.1	80	4	453.84	0.00	8.62	0.00	0.00	0.30	0.00	0.00	0.30
a5-40-0.1	80	5	414.51	0.00	19.03	0.00	0.00	0.74	0.00	0.00	0.74
Avg. <i>gap</i>	53	3	67.55	0.00	3.54	0.00	0.00	0.13	0.00	0.00	0.13
a2-16-0.4	32	2	237.38	0.00	0.03	0.00	0.00	0.01	0.00	0.00	0.01
a2-20-0.4	40	2	280.70	0.00	0.83	0.00	0.00	0.01	- 0.58	- 0.58	0.03
a2-24-0.4	48	2	348.04	0.00	0.42	- 0.29	- 0.29	0.05	- 0.53	- 0.53	0.02
a3-18-0.4	36	3	236.82	0.00	0.07	0.00	0.00	0.00	0.00	0.00	0.00
a3-24-0.4	48	3	274.80	0.00	0.28	0.00	0.00	0.01	0.00	0.00	0.02
a3-30-0.4	60	3	413.37	0.00	1.65	0.00	0.00	0.18	- 0.02	- 0.02	0.15
a3-36-0.4	72	3	484.14	0.00	5.11	0.00	0.00	1.06	- 0.46	- 0.50	1.27
a4-16-0.4	32	4	222.49	0.00	0.09	0.00	0.00	0.01	0.00	0.00	0.01
a4-24-0.4	48	4	311.03	0.00	0.66	0.00	0.00	0.11	- 0.06	- 0.06	0.13
a4-32-0.4	64	4	394.26	0.00	11.36	0.00	0.00	0.29	- 0.08	- 0.08	0.25
a4-40-0.4	80	4	453.84	0.00	6.96	0.00	0.00	0.91	0.00	0.00	0.87
a5-40-0.4	80	5	414.51	0.00	20.35	0.00	0.00	0.81	0.00	0.00	0.79
Avg. <i>gap</i>	53	3	339.28	0.00	3.98	- 0.02	- 0.02	0.29	- 0.14	- 0.15	0.29
a2-16-0.7	32	2	240.66	0.00	0.09	0.00	0.00	0.03	- 1.36	- 1.36	0.01
a2-20-0.7	40	2	287.17	NA	120.00	2.18	2.18	0.52	- 2.25	- 2.25	0.05
a2-24-0.7	48	2	358.21	0.00	16.02	- 1.40	- 1.40	0.06	- 3.35	- 3.35	0.03
a3-18-0.7	36	3	240.58	0.00	0.80	0.00	0.00	0.01	- 1.57	- 1.57	0.00
a3-24-0.7	48	3	277.72	0.00	2.54	- 0.63	- 0.63	0.07	- 1.05	- 1.05	0.02
a3-30-0.7	60	3	417.06	NA	120.00	2.29	2.14	0.59	- 0.34	- 0.34	0.73
a3-36-0.7	72	3	485.91	1.67	120.00	3.63	2.32	1.17	0.92	0.70	0.56
a4-16-0.7	32	4	223.13	0.00	1.12	0.00	0.00	0.04	- 0.29	- 0.29	0.00
a4-24-0.7	48	4	318.19	0.01	30.58	- 0.55	- 0.55	0.95	- 2.14	- 2.14	0.30
a4-32-0.7	64	4	387.99	10.85	120.00	3.74	3.67	1.39	1.76	1.74	0.34
a4-40-0.7	80	4	443.62	NA	120.00	12.30	12.30	0.90	4.01	3.89	0.53
a5-40-0.7	80	5	405.99	10.26	120.00	3.20	3.20	1.89	2.39	2.39	0.56
Avg. <i>gap</i>	53	3	340.52	NA	64.26	2.06	1.93	0.63	- 0.27	- 0.30	0.26

B&C: Existing results of branch-and-cut method for a 2-index model of e-ADARP with SCS with maximum running time of 120 min [24]; *P*: Results of the proposed method for e-ADARP with maximum running time of 2 min.

TABLE 5.4: Benchmark instances: Comparison of solution quality attained by the proposed method for e-ADARP with SCS and MED vs. existing method for e-ADARP.

5.4.3 Effectiveness of the proposed e-ADARP with MED

In this section, we analyze the effect of changing the MED fleet size on solution quality for the electric autonomous dial-a-ride problem.

In Tables 5.6, 5.7, 5.8, 5.9, we analyze the policy implication of MED fleet size on solution quality for e-ADARP. The effect of including and excluding a MED vehicle into the fleet are analyzed. By default, we set MED fleet size as 3, since the total SCS is 3 in benchmark instances. Subsequently, we modify the MED fleet size by adding or deleting a MED vehicle at all the default locations such as (0, 0),

Case	Size	m	LB	B&C		P (with SCS)			P (with MED)		
				gap^*	t	$g\bar{a}p$	gap^*	t	$g\bar{a}p$	gap^*	t
u2-16-0.1	32	2	57.61	0.00	0.35	0.00	0.00	0.00	0.00	0.00	0.01
u2-20-0.1	40	2	55.59	0.00	0.16	0.00	0.00	0.02	0.00	0.00	0.03
u2-24-0.1	48	2	91.27	0.00	7.20	-0.40	-0.40	0.76	-1.76	-1.76	0.08
u3-18-0.1	36	3	50.74	0.00	0.18	0.00	0.00	0.03	0.00	0.00	0.01
u3-24-0.1	48	3	67.56	0.00	2.17	0.00	0.00	1.08	0.00	0.00	0.13
u3-30-0.1	60	3	76.75	0.00	7.30	0.00	0.00	0.30	0.00	0.00	0.35
u3-36-0.1	72	3	104.04	0.00	18.08	0.00	0.00	0.71	-0.52	-0.52	0.11
u4-16-0.1	32	4	53.58	0.00	0.80	-1.18	-1.18	0.06	-1.18	-1.18	0.08
u4-24-0.1	48	4	89.83	0.00	0.22	0.01	0.01	0.20	-0.19	-0.19	0.07
u4-32-0.1	64	4	99.29	0.00	19.31	0.06	0.06	0.61	0.14	0.14	0.23
u4-40-0.1	80	4	133.11	0.00	3.09	0.44	0.44	0.60	0.06	0.02	0.73
u5-40-0.1	80	5	114.12	6.78	120.00	8.00	7.98	0.43	7.15	7.15	0.83
Avg. gap	53	3	82.79	0.57	14.61	0.58	0.58	0.40	0.31	0.31	0.22
u2-16-0.4	32	2	57.65	0.00	0.43	0.00	0.00	0.04	-0.07	-0.07	0.02
u2-20-0.4	40	2	56.34	0.00	0.20	0.00	0.00	0.02	-1.33	-1.33	0.05
u2-24-0.4	48	2	91.63	0.00	12.62	0.00	0.00	0.91	-2.15	-2.15	0.15
u3-18-0.4	36	3	50.74	0.00	0.23	0.00	0.00	0.02	0.00	0.00	0.01
u3-24-0.4	48	3	67.56	0.00	3.68	0.00	0.00	1.08	0.00	0.00	0.12
u3-30-0.4	60	3	76.75	0.00	5.61	0.00	0.00	0.30	0.00	0.00	0.35
u3-36-0.4	72	3	104.06	0.00	33.50	0.83	0.83	1.33	-0.54	-0.54	0.10
u4-16-0.4	32	4	53.58	0.00	0.74	-1.18	-1.18	0.06	-1.18	-1.18	0.08
u4-24-0.4	48	4	89.83	0.00	0.47	-0.04	-0.04	0.20	-0.19	-0.19	0.75
u4-32-0.4	64	4	99.29	0.00	44.46	0.66	0.66	0.43	0.14	0.14	0.20
u4-40-0.4	80	4	133.91	0.00	44.22	0.73	0.73	0.60	-0.49	-0.58	1.08
u5-40-0.4	80	5	112.58	8.57	120.00	10.06	9.98	0.43	8.90	8.62	0.77
Avg. gap	53	3	82.83	0.71	22.18	0.92	0.91	0.45	0.26	0.23	0.31
u2-16-0.7	32	2	59.19	0.00	5.64	0.00	0.00	0.00	-2.67	-2.67	0.02
u2-20-0.7	40	2	56.83	0.00	1.20	0.00	0.00	0.02	-2.23	-2.23	0.02
u2-24-0.7	48	2	90.83	NA	120.00	2.14	2.12	1.08	-1.29	-1.29	0.15
u3-18-0.7	36	3	50.99	0.00	0.40	0.00	0.00	0.04	-0.49	-0.49	0.01
u3-24-0.7	48	3	68.39	0.00	6.67	0.88	0.88	1.78	-1.21	-1.21	0.23
u3-30-0.7	60	3	78.14	0.00	56.69	-0.08	-0.08	0.57	-1.78	-1.78	0.35
u3-36-0.7	72	3	104.37	1.36	120.00	2.28	2.01	0.82	-0.83	-0.83	0.11
u4-16-0.7	32	4	53.87	0.00	1.48	0.19	0.19	0.06	-1.71	-1.71	0.03
u4-24-0.7	48	4	89.96	0.00	0.38	0.01	0.01	0.05	-0.33	-0.33	0.72
u4-32-0.7	64	4	99.50	0.00	47.12	2.64	2.45	1.55	-0.07	-0.07	0.26
u4-40-0.7	80	4	133.01	NA	120.00	3.58	3.51	0.55	0.21	0.10	0.79
u5-40-0.7	80	5	109.28	NA	120.00	16.47	16.04	1.50	12.61	12.55	0.98
Avg. gap	53	3	82.87	NA	49.97	2.34	2.26	0.67	0.02	0.00	0.31

E: Existing results of branch-and-cut method for a 2-index model of e-ADARP with SCS with maximum running time of 120 min
[24]; *P*: Results of the proposed method for e-ADARP with maximum running time of 2 min.

TABLE 5.5: Uber instances: Comparison of solution quality attained by the proposed approach vs. existing method for e-ADARP.

Case	Size	m	LB	MED= 3	MED= 2			MED= 4		
					$no(0, 0)$	$no(-4, -4)$	$no(4, 4)$	$add(0, 0)$	$add(-4, -4)$	$add(4, 4)$
a2-16-0.1	32	2	237.38	0.00	0.00	0.00	0.00	0.00	0.00	0.00
a2-20-0.1	40	2	279.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00
a2-24-0.1	48	2	346.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00
a3-18-0.1	36	3	236.81	0.00	0.00	0.00	0.00	0.00	0.00	0.00
a3-24-0.1	48	3	274.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00
a3-30-0.1	60	3	413.27	0.00	0.00	0.00	0.00	0.00	0.00	0.00
a3-36-0.1	72	3	481.72	0.00	0.00	0.00	0.00	0.00	0.00	0.00
a4-16-0.1	32	4	222.49	0.00	0.00	0.00	0.00	0.00	0.00	0.00
a4-24-0.1	48	4	310.84	0.00	0.00	0.00	0.00	0.00	0.00	0.00
a4-32-0.1	64	4	393.96	0.00	0.00	0.00	0.00	0.00	0.00	0.00
a4-40-0.1	80	4	453.84	0.00	0.00	0.00	0.00	0.00	0.00	0.00
a5-40-0.1	80	5	414.51	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Avg.	53	3	338.74	0.00	0.00	0.00	0.00	0.00	0.00	0.00
a2-16-0.4	32	2	237.38	0.00	0.00	0.30	0.00	0.00	0.00	0.00
a2-20-0.4	40	2	280.70	− 0.58	− 0.58	− 0.58	− 0.58	− 0.58	− 0.58	− 0.58
a2-24-0.4	48	2	348.04	− 0.53	− 0.53	− 0.53	− 0.53	− 0.53	− 0.53	− 0.53
a3-18-0.4	36	3	236.82	0.00	0.00	0.00	0.00	0.00	0.00	0.00
a3-24-0.4	48	3	274.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00
a3-30-0.4	60	3	413.37	− 0.02	− 0.02	− 0.02	− 0.02	− 0.02	− 0.02	− 0.02
a3-36-0.4	72	3	484.14	− 0.50	0.27	0.28	0.02	− 0.50	− 0.50	− 0.50
a4-16-0.4	32	4	222.49	0.00	0.00	0.00	0.00	0.00	0.00	0.00
a4-24-0.4	48	4	311.03	− 0.06	− 0.06	− 0.06	− 0.06	− 0.06	− 0.06	− 0.06
a4-32-0.4	64	4	394.26	− 0.08	− 0.08	− 0.08	− 0.08	− 0.08	− 0.08	− 0.08
a4-40-0.4	80	4	453.84	0.00	0.00	0.35	0.00	0.00	0.00	0.00
a5-40-0.4	80	5	414.51	0.00	0.00	0.31	0.00	0.00	0.00	0.00
Avg.	53	3	339.28	− 0.15	− 0.08	0.00	− 0.10	− 0.15	− 0.15	− 0.15
a2-16-0.7	32	2	240.66	− 1.36	− 1.07	− 1.07	− 1.07	− 1.36	− 1.36	− 1.36
a2-20-0.7	40	2	287.17	− 2.25	3.84	NA	3.84	− 2.25	− 2.25	− 2.25
a2-24-0.7	48	2	358.21	− 3.35	NA	NA	NA	− 3.35	− 3.35	− 3.35
a3-18-0.7	36	3	240.58	− 1.57	− 1.57	− 1.57	− 1.57	− 1.57	− 1.57	− 1.57
a3-24-0.7	48	3	277.72	− 1.05	− 0.74	− 1.05	− 0.74	− 1.05	− 1.05	− 1.05
a3-30-0.7	60	3	417.06	− 0.34	NA	NA	NA	− 0.34	− 0.34	− 0.34
a3-36-0.7	72	3	485.91	0.70	NA	NA	NA	0.70	0.70	0.70
a4-16-0.7	32	4	223.13	− 0.29	− 0.29	− 0.29	− 0.29	− 0.29	− 0.29	− 0.29
a4-24-0.7	48	4	318.19	− 2.14	− 1.36	− 1.36	− 1.36	− 2.14	− 2.14	− 2.14
a4-32-0.7	64	4	387.99	1.74	6.43	4.59	NA	1.74	1.74	1.74
a4-40-0.7	80	4	443.62	3.89	NA	NA	NA	3.89	3.89	3.89
a5-40-0.7	80	5	405.99	2.39	NA	8.24	4.90	2.39	2.39	2.39
Avg.	53	3	340.52	− 0.30	NA	NA	NA	− 0.30	− 0.30	− 0.30

$no(0, 0)$: a MED is removed at this location; $no(4, 4)$: a MED is removed at this location; $no(-4, -4)$: a MED is removed at this location; $add(0, 0)$: a MED is added at this location; $add(4, 4)$: a MED is added at this location; $add(-4, -4)$: a MED is added at this location.

TABLE 5.6: Benchmark instances: Policy implications: Effect of MED fleet size on solution quality for e-ADARP with MED.

$(4, 4)$, and $(-4, -4)$. The gap^* value is recorded over five runs with the stopping criterion as 2 min. From the results, the average solution quality degrades when reducing the MED fleet size, and the average solution quality does not improve when increasing the MED fleet size.

Case	Size	m	LB	MED=3	MED=2					MED=4				
					$-CS_1$	$-CS_2$	$-CS_3$	$-CS_4$	$-CS_5$	$+CS_1$	$+CS_2$	$+CS_3$	$+CS_4$	$+CS_5$
u2-16-0.1	32	2	57.61	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
u2-20-0.1	40	2	55.59	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
u2-24-0.1	48	2	91.27	-1.76	-1.76	-1.76	-1.76	-1.76	-1.76	-1.76	-1.76	-1.76	-1.76	-1.76
u3-18-0.1	36	3	50.74	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
u3-24-0.1	48	3	67.56	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
u3-30-0.1	60	3	76.75	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
u3-36-0.1	72	3	104.04	-0.52	-0.52	-0.52	-0.52	-0.52	-0.52	-0.52	-0.52	-0.52	-0.52	-0.52
u4-16-0.1	32	4	53.58	-1.18	-1.18	-1.18	-1.18	-1.18	-1.18	-1.18	-1.18	-1.18	-1.18	-1.18
u4-24-0.1	48	4	89.83	-0.19	-0.19	-0.19	-0.19	-0.19	-0.19	-0.19	-0.19	-0.19	-0.19	-0.19
u4-32-0.1	64	4	99.29	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
u4-40-0.1	80	4	133.11	0.02	0.04	0.02	0.23	0.53	NA	0.02	0.02	0.02	0.02	0.02
u5-40-0.1	80	5	114.12	7.15	7.15	7.72	7.82	8.09	NA	7.15	7.15	7.15	7.15	7.15
Avg.	53	3	82.79	0.31	0.31	0.37	0.41	0.43	NA	0.31	0.31	0.31	0.31	0.31

$-CS_l$: a MED is removed at charging depot location l ; $+CS_l$: a MED is added at charging depot location l .

TABLE 5.7: Uber instances: Policy implications: Effect of MED fleet size on solution quality for e-ADARP-MED with $r = 0.1$.

Case	Size	m	LB	MED=3					MED=2					MED=4				
				MED=3					MED=2					MED=4				
				$-CS_1$	$-CS_2$	$-CS_3$	$-CS_4$	$-CS_5$	$-CS_1$	$-CS_2$	$-CS_3$	$-CS_4$	$-CS_5$	$+CS_1$	$+CS_2$	$+CS_3$	$+CS_4$	$+CS_5$
u2-16-0.4	32	2	57.65	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07
u2-20-0.4	40	2	56.34	-1.33	-1.33	-1.33	-1.33	-1.33	-1.33	-1.33	-1.33	-1.33	-1.33	-1.33	-1.33	-1.33	-1.33	-1.33
u2-24-0.4	48	2	91.63	-2.15	-2.15	-2.15	-2.15	-2.15	-2.15	-2.15	-2.15	-2.15	-2.15	-2.15	-2.15	-2.15	-2.15	-2.15
u3-18-0.4	36	3	50.74	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
u3-24-0.4	48	3	67.56	0.00	0.15	0.00	0.00	0.00	0.00	0.15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
u3-30-0.4	60	3	76.75	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
u3-36-0.4	72	3	104.06	-0.54	-0.54	-0.12	-0.54	-0.54	-0.54	-0.54	-0.12	-0.54	-0.54	-0.54	-0.54	-0.54	-0.54	-0.54
u4-16-0.4	32	4	53.58	-1.18	-1.18	-1.18	-1.18	-1.18	-1.18	-1.18	-1.18	-1.18	-1.18	-1.18	-1.18	-1.18	-1.18	-1.18
u4-24-0.4	48	4	89.83	-0.19	-0.19	-0.19	-0.19	-0.19	-0.19	-0.19	-0.19	-0.19	-0.19	-0.19	-0.19	-0.19	-0.19	-0.19
u4-32-0.4	64	4	99.29	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
u4-40-0.4	80	4	133.91	-0.01	-0.58	-0.37	-0.07	NA	-0.01	-0.58	-0.37	-0.07	NA	-0.58	-0.58	-0.58	-0.58	-0.58
u5-40-0.4	80	5	112.58	8.62	9.63	9.29	9.57	NA	8.62	9.63	9.29	9.57	NA	8.62	8.62	8.62	8.62	8.62
Avg.	53	3	82.83	0.27	0.32	0.34	0.35	NA	0.27	0.32	0.34	0.35	NA	0.23	0.23	0.23	0.23	0.23

$-CS_l$: a MED is removed at charging depot location l ; $+CS_l$: a MED is added at charging depot location l .

TABLE 5.8: Uber instances: Policy implications: Effect of MED fleet size on solution quality for e-ADARP-MED with $r=0.4$.

Case	Size	m	LB	MED=3					MED=2					MED=4				
				MED=3					MED=2					MED=4				
				$-CS_1$	$-CS_2$	$-CS_3$	$-CS_4$	$-CS_5$	$-CS_1$	$-CS_2$	$-CS_3$	$-CS_4$	$-CS_5$	$+CS_1$	$+CS_2$	$+CS_3$	$+CS_4$	$+CS_5$
u2-16-0.7	32	2	59.19	-2.67	-2.67	-2.67	-2.67	-2.67	-2.67	-2.67	-2.67	-2.67	-2.67	-2.67	-2.67	-2.67	-2.67	-2.67
u2-20-0.7	40	2	56.86	-2.23	-2.23	-2.23	-2.23	-2.23	-2.23	-2.23	-2.23	-2.23	-2.23	-2.23	-2.23	-2.23	-2.23	-2.23
u2-24-0.7	48	2	90.83	-1.29	-1.29	-1.29	-1.29	-1.29	-1.29	-1.29	-1.29	-1.29	-1.29	-1.29	-1.29	-1.29	-1.29	-1.29
u3-18-0.7	36	3	50.99	-0.49	-0.49	-0.49	-0.49	-0.49	-0.49	-0.49	-0.49	-0.49	-0.49	-0.49	-0.49	-0.49	-0.49	-0.49
u3-24-0.7	48	3	68.39	-1.21	-1.07	-1.21	-1.21	-1.21	-1.21	-1.07	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21
u3-30-0.7	60	3	78.14	-1.78	-1.78	-1.78	-1.78	-1.78	-1.78	-1.78	-1.78	-1.78	-1.78	-1.78	-1.78	-1.78	-1.78	-1.78
u3-36-0.7	72	3	104.37	-0.83	-0.83	-0.41	-0.83	-0.41	-0.83	-0.83	-0.41	-0.83	-0.41	-0.83	-0.83	-0.83	-0.83	-0.83
u4-16-0.7	32	4	53.87	-1.71	-1.71	-1.71	-1.71	-1.71	-1.71	-1.71	-1.71	-1.71	-1.71	-1.71	-1.71	-1.71	-1.71	-1.71
u4-24-0.7	48	4	89.96	-0.22	-0.33	-0.33	-0.33	-0.33	-0.33	-0.33	-0.33	-0.33	-0.33	-0.33	-0.33	-0.33	-0.33	-0.33
u4-32-0.7	64	4	99.50	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07	-0.07
u4-40-0.7	80	4	133.01	0.10	0.10	0.53	0.61	NA	0.87	0.10	0.53	0.61	NA	0.10	0.10	0.10	0.10	0.10
u5-40-0.7	80	5	109.28	12.55	13.26	12.59	12.88	NA	12.75	13.26	12.59	12.88	NA	12.55	12.55	12.55	12.55	12.55
Avg.	53	3	82.87	0.00	0.07	0.08	0.07	NA	0.09	0.07	0.08	0.07	NA	0.00	0.00	0.00	0.00	0.00

$-CS_l$: a MED is removed at charging depot location l ; $+CS_l$: a MED is added at charging depot location l .

TABLE 5.9: Uber instances: Policy implications: Effect of MED fleet size on solution quality for e-ADARP-MED with $r=0.7$.

	Obj.	TC	ERT	RD	VWT	PRT
Cases with $r=0.1$	MED	338.74	425.63	78.06	1224.46	638.83
	SCS	338.74	425.63	78.06	1224.46	638.83
	<i>Imp. (%)</i>	0.00	0.00	0.00	0.00	0.00
Cases with $r=0.4$	MED	338.74	425.63	78.06	1248.63	645.02
	SCS	339.20	427.04	75.49	1257.79	629.80
	<i>Imp. (%)</i>	-0.13	-0.33	3.40	-0.73	2.42
Cases with $r=0.7$	MED	340.50	428.59	76.24	1270.74	640.91
	SCS	348.83	436.00	80.51	1357.55	521.33
	<i>Imp. (%)</i>	-2.39	-1.70	-5.30	-6.39	22.94
<i>Imp. (%)</i> : Percentage improvement in quantifier value when MEDs are used instead of SCS; Negative value of this quantifier is desirable. Obj.: Objective function of the best solution; TC: Total travel time of EV fleet; ERT: Total excess user ride time, RD: Total route duration of EV fleet; VWT: Total waiting time of EV fleet; PRT: Total passenger ride time.						

TABLE 5.10: Benchmark instances: Effectiveness of the proposed e-ADARP-MED system on passenger experiences, fleet idle time, etc.

5.4.4 Effectiveness of various recharging infrastructures on passenger experiences for e-ADARP

In this section, we analyze the effectiveness of using various recharging infrastructures on passenger experiences for the electric autonomous dial-a-ride problem.

We show the details of the best solution x_{best} attained by the proposed solution method for e-ADARP (with SCS/MED) on benchmark and UBER instances in Appendix C. In Tables 5.10 and 5.11, we show the effectiveness of the proposed

	Obj.	TC	ERT	RD	VWT	PRT
Uber cases with r=0.1	MED	83.24	107.45	10.62	488.16	439.94
	SCS	83.52	108.23	9.40	554.05	399.93
	<i>Imp.(%)</i>	-0.34	-0.72	12.93	-11.89	10.00
Uber cases with r=0.4	MED	83.24	107.45	10.62	614.89	435.39
	SCS	83.92	108.57	9.96	645.88	428.71
	<i>Imp.(%)</i>	-0.81	-1.03	6.67	-4.80	1.56
Uber cases with r=0.7	MED	83.30	107.52	10.62	620.61	452.72
	SCS	85.31	110.49	9.77	654.91	377.60
	<i>Imp.(%)</i>	-2.36	-2.69	8.68	-5.24	19.89

Imp.(%): Percentage improvement in quantifier value when MEDs are used instead of SCS; Negative value of this quantifier is desirable. Obj.: Objective function of the best solution; TC: Total travel time of EV fleet; ERT: Total excess user ride time, RD: Total route duration of EV fleet; VWT: Total waiting time of EV fleet; PRT: Total passenger ride time.

TABLE 5.11: Uber instances: Effectiveness of the proposed e-ADARP-MED system on passenger experiences, fleet idle time, etc.

e-ADARP-MED system. When compared to using SCS, e-ADARP with MED shows percentage improvement in several factors. Particularly, the improvements in fleet travel time, fleet route duration, fleet waiting time constitute lesser battery consumption from the fleet operator's point of view, and the improvements in user excess ride time and user ride time facilitate higher convenience from the customer's point of view.

5.5 Discussion

On-demand mobility has become a topic of interest due to the range of applications (medical emergency transit, public transportation, airport and seaport operations, etc.,) associated with it. In today's world, DARP often occurs in real-time rather than in advanced reservation mode, and hence attaining quicker solutions can be critical. On the other hand, the emergence of electric autonomous vehicles directly targets the problems of oil dependency, pollution, and global warming. The feasibility of modern vehicular technologies needs to be assessed in the context of dial-a-ride systems. In this chapter, we have proposed mobile energy disseminator, an EV charging mechanism, for the electric autonomous dial-a-ride system. Firstly, we have formulated the e-ADARP with MEDs. Secondly, we have developed a GPU-based solution methodology for e-ADARP. Simulations are conducted on standard test instances from the literature. The proposed ALNS on GPU is faster than its CPU counterpart and attains better average solution quality when compared to the existing approaches for various instances. The effect of including and excluding a MED vehicle on overall solution quality is analyzed. Furthermore, when compared to using SCS for e-ADARP, the usage of MED attains improvements in fleet travel time, fleet idle time, fleet waiting time, user ride time, and user excess ride time. We conclude that the modern EV charging method such as MED that acts as mobile charging stations can improve the overall quality of electric autonomous dial-a-ride systems when compared to conventional charging methodologies. The following are some potential future research directions: 1) solve richer variants of the proposed problem with heterogeneous demand, fleet, and dynamic traffic conditions, 2) apply the proposed framework for emergency transportation services (e.g. medical ambulance, fire and rescue operations), where attaining quick solutions is critical, 3) solve the dial-a-ride problem with a mix of conventional and electric vehicles while allowing multiple recharging infrastructures to support the operations of the green transportation systems.

Chapter 6

Conclusion and Future Directions

6.1 Conclusion and Summary of the Contributions

On-demand mobility has become a topic of interest due to the range of applications (medical emergency transit, public transportation, airport, and seaport operations, etc.) associated with it. In the last few years, the general public has developed a significant interest in MOD transportation. A dial-a-ride is a form of MOD transit service, which is NP-hard in nature. In today's world, DARP often occurs in real-time rather than in advanced reservation mode, and hence attaining quicker solutions can be critical. GPU computing has turned out to be a fundamental pillar in modern science. On the other hand, disruption in centralized fleet operations has been a big issue since the recovery cost is huge for the transportation industry. Despite its importance, this problem has never been addressed in the context of an advanced user-oriented form of mobility service that is becoming increasingly relevant in today's ride-sharing transportation. The emergence of electric autonomous vehicles directly targets the problems of oil dependency, pollution, and global warming. To promote green transportation, the feasibility and effectiveness of the modern vehicular technologies and recharging infrastructures need to be assessed in the context of dial-a-ride operations.

Therefore, in this dissertation, we have developed efficient modeling frameworks and GPU-accelerated solution algorithms for the static and dynamic dial-a-ride

problem with electric autonomous vehicles and modern recharging infrastructures. We have tested the proposed frameworks on the DARP instances based on the Montreal Transit Commission and Uber technologies in San Francisco City, USA. We have analyzed the effect of GPU on accelerating solution algorithms for solving DARP under dynamic real-life settings, the effect of minimizing fleet size on the service provider and customer experience both before and after disruptive events such as vehicle breakdown, the effect of optimizing depot locations on transportation costs, and the effect of the latest recharging infrastructures on reducing the EV range anxiety in dial-a-ride operations. We provide a summary of each chapter of the thesis in the following.

In Chapter 2, we provide an in-depth review of the dial-a-ride problem to support the growing interest in this research area. We first described the relevance of this problem in the context of a real-life demand-responsive transportation system. As a combination of the routing and scheduling problem, the dial-a-ride routing problem has an interesting, but complex structure. Therefore, most of the studies proposed heuristics and metaheuristics to handle the problem, while a few exact algorithms were developed. We discussed the models and solution techniques that have been developed for several variants of the DARP and summarized the results.

In Chapter 3, we have introduced a GPU-based solution methodology to solve DARP under dynamic real-life settings. First, we have introduced the building blocks of the proposed GPU framework including CUDA kernels and local search operation. Secondly, we have proposed five device optimization strategies tailored for the described problem to improve computational efficiency using GPU. Thirdly, we have integrated three meta-heuristics, tabu search (TS), variable neighborhood search (VNS), and adaptive large neighborhood search (ALNS), with the proposed GPU framework. Simulations are conducted on DARP test instances from the literature. The proposed GPU approach attains a higher speed up when compared to the conventional GPU approach due to the proposed device optimization strategies. The effects of inclusion and exclusion of each device optimization strategy on computational speed-up have been reported. Finally, the proposed GPU approach attains better average solution quality in a short time than the existing best-performing sequential approaches for DARP. We conclude that accelerating time-critical neighborhood explorations in GPU along with a guided search of meta-heuristic can be significant in efficiently solving DARP.

In Chapter 4, we have introduced the disruptive dial-a-ride problem, in which vehicle breakdown may occur at any point of time during the transportation service. We have formulated the disruptive dial-a-ride problem with vehicle breakdown (DDARP-VB) and proposed a novel GPU-based adaptive large neighborhood search algorithm to solve this problem. Several new realistic constraints are modeled to accurately capture the effects of disruption in dial-a-ride problems with a vehicle breakdown. The general idea of the ALNS solution algorithm is the removal and reinsertion of requests based on their past performances. During the reinsertion of a request, there are many combinations of solution outcomes. Since the sequential evaluation of all these combinations is computationally expensive, we parallelize this operation by means of CUDA kernels. The results show that the proposed G-ALNS performs faster than its CPU counterpart while performing better than other existing solution methods in a short time. The G-ALNS with fleet size minimization strategy attains solutions with a significant reduction in fleet cost, at the expense of a negligible increase in travel cost. With realistic estimations of operational cost, the G-ALNS-FSM attains lower operational cost compared to G-ALNS, when tested on various standard instances from the literature.

In Chapter 5, we have introduced mobile energy disseminator (MED), an electric vehicle (EV) recharging mechanism, for the electric autonomous dial-a-ride system. First, we have formulated the electric autonomous dial-a-ride problem (e-ADARP) with MEDs. Second, we have developed a GPU-based optimization methodology and scheduling algorithm for e-ADARP. Simulations are conducted on benchmark instances and UBER taxi trip instances. The proposed solution algorithm produces competitive solution quality when compared to the optimal solutions with static charging stations while generating several new best solutions with the usage of mobile energy disseminators for vehicle recharging. The effects of including and excluding a MED vehicle on overall solution quality are analyzed. Furthermore, when compared to using static charging stations (SCS) for e-ADARP, the usage of MED attains improvements in fleet travel time, fleet idle time, fleet waiting time, user ride time, and user excess ride time. We conclude that the modern EV charging method such as MED that acts as a moving charging station can improve the overall operational quality of electric autonomous dial-a-ride systems when compared to conventional charging methodologies.

To achieve a real-time adaptive dial-a-ride system, it is essential to develop efficient mathematical models that capture the most advanced form of vehicular technologies (autonomous & electric vehicles and their recharging infrastructures) and novel optimization and scheduling algorithms that are capable of generating high-quality solutions in *real-time* by leveraging on current-generation computing architectures (e.g. GPU), while enabling *adaptive* mechanisms to tackle disruptive scenarios (breakdowns, weather, accidents). We believe that the research of this dissertation could act as a key step towards the commercial deployment of autonomous mobility-on-demand systems for future urban mobility. The following are some of the limitations. First, we modeled a realistic operational cost function for disruption management based on Singapore's data, while the transport network of San Francisco City was considered for the electric autonomous dial-a-ride problem. To facilitate the integration of the above models, it is necessary to design a network model that reflects real-world scenarios of the same city. The key challenge involved during the integration is that the simultaneous consideration of disruptions and electric autonomous vehicles might further increase the problem complexity. Particularly, the fleet minimization strategy used in the disruption management might not be favorable for electric autonomous vehicles since many visits to recharging stations might be required for EVs which might increase the overall transportation costs. Thus, new fleet management strategies might be required to handle the disruptions for autonomous EV-based DARP. Second, to ensure a fair comparison with existing approaches, we presented results without including the factor of traffic congestion in the transport networks. Traffic flow dynamics need to be modeled to enhance the accuracy of the transport models and to fully reflect the real-world implications. In today's world, the sophisticated machine learning and deep learning approaches can dynamically estimate the variations in travel times induced by congestion much more accurately. The presented real-time adaptive algorithms need to be integrated with such learning approaches to realize a non-oblivious dial-a-ride system. Therefore, methodological frameworks to fully integrate such machine learning and optimization approaches will further enhance the system accuracy. Third, the institutional challenge of integrating autonomous electric vehicles into a daily transit system is that it may not be feasible to fully replace the fossil fuel-powered combustion engine vehicles with immediate effect. Therefore, the integration of e-AV into daily transport is expected to a slow process. In other words, the e-AV will co-exist with combustion engine cars for a

few years before fully taking over the road. Hence, it might be timely and essential to study the integration of multiple types of vehicles into the road-network and assess the possible real-world implications. We further elaborate on some of these potential future research directions in the next section.

6.2 Future Directions

In this dissertation, we have introduced new modeling frameworks and GPU-based solution methodologies for the static and dynamic dial-a-ride problem with electric autonomous vehicles. This research has provided rigorous answers to many questions in this field while opening many new exciting research directions. In this section, we outline some additional directions for future research.

6.2.1 Peer-to-peer EV recharging scheme for electric vehicle dial-a-ride systems

Electric vehicles (EV) are transforming the fundamental design of transportation operations due to the need to reduce our carbon footprint. The manufacturing companies like Tesla, Nissan, Chevrolet, etc., have been constantly endeavoring to make the EV more appealing and affordable. However, the adoption of EVs in vehicle routing operations and private mobility usage remains slow due to the concerns regarding battery life, battery range, and limited access to charging stations [132]. The major cities in the developed countries may have many recharging stations, but it may be still insufficient due to their slender infrastructure or unable to support a larger EV population [115]. Meanwhile, the charging stations are certainly not abundant in remote areas.

Although there are several innovations in EV technologies, the charging time and battery range are two of the most critical elements for future widespread EV adoption. Some new battery technologies such as battery swapping [111], mobile charging [216], mobile energy disseminators [132], and solar-powered roads [222] have their specific advantages and disadvantages.

FIGURE 6.1: Peer-to-Peer EV-recharging mechanism [2].

The latest advancement in the charging technology is called peer-to-peer (P2P) EV recharging, which is similar to the mobile energy disseminator (MED). Figure 6.1 shows the mechanism of P2P when compared to mobile energy disseminators, otherwise referred to as mobile charging stations (MOCs). Differently from the MED, the novel P2P scheme does not require an external recharging carrier to be roaming in the roads. In this P2P charging scheme, multiple EVs can share their battery power by transferring electrical energy wired or wirelessly through the inductive power transfer (IPT) mechanism. Furthermore, a cloud-based control system can be activated to keep track of the debits/credits based on the power transaction between multiple EVs. Today's state-of-the-art blockchain-based encrypted transactions can make the P2P scheme a very attractive option in the future.

A study on EV traffic simulation was conducted in [2] with the use of the peer-to-peer charging technique, and it shows the benefit of sharing energy in 240 km transit highways. Meanwhile, DARP is a fundamental optimization problem in today's ride-sharing people transportation [131]. Therefore, it is necessary to design mathematical models and solution algorithms for the optimization and scheduling of EVs with a P2P charging scheme for dial-a-ride operations to fully assess the scalability and economic benefits of such new recharging infrastructures.

6.2.2 Modeling frameworks and accelerated algorithms for real-world emergency dial-a-ride operations:

The decision-making time in emergency vehicle routing operations (ambulance/-fire/police/rescue vehicles) can greatly affect the probability of lives that will be saved. There are several obstacles involved in expediting the emergency vehicle movement such as traffic lights, congestion, travel time, etc. The general idea of using sirens and flashlights may not be sufficient in the current world due to the extremely congested road networks in major cities of the developed countries especially during the rush hours [223]. Various modeling and solution approaches have been discussed in the literature to address the emergency routing operations ranging from traffic light synchronization to the path planning process.

FIGURE 6.2: A general framework for real-world emergency routing operations.

However, very few studies have considered stochastic/dynamic information for the decision-making process while solving DARPs. Kim et al. 2016 [224] present a dynamic vehicle routing problem with traffic congestion. Song et al. 2018 [118] present a DARP with time-dependent travel times. The considerations of stochastic and time-dependent information such as travel time, weather conditions, and accidents are essential to realistically model the vehicle routing operations. The

exponential increase in the available historical data can be used to design efficient machine learning algorithms for estimating the stochastic information about future [118]. With advancements in algorithmic technologies, it is possible to accurately predict information about the future.

We show a generic framework in Figure 6.2 that encompasses various real-world characteristics (accident, weather, traffic, signals) for emergency dial-a-ride operations. The framework can be split into three modules: simulation/real-world environment, traffic evaluation, and DARP manager. For testing purposes, a simulator (e.g. VISSIM [225], SUMO [2], MATSIM [226]) can be employed for mimicking the real-world transportation environment. For prediction modules, developing efficient machine learning algorithms are necessary to estimate stochastic information about traffic congestion and signals. For DARP solvers, accelerated algorithms using GPU may be necessary to perform "real-time" re-optimization according to the new information provided by the prediction modules or other disruptive factors [131].

Therefore, we encourage researchers to focus on the modeling of realistic dial-a-ride operations, specifically for emergency routing applications and enable GPU-accelerated solvers coupled with prediction modules to advance the state-of-the-art in emergency routing systems.

6.2.3 Routing a mix of vehicular technologies and recharging stations with new energy consumption models:

Recently, there has been a steady increase in fuel costs and governmental regulations for reducing greenhouse gas emissions in the transportation sectors. Due to the rising global and environmental concerns, vehicle manufacturers are turning into electric vehicles (EV). Although the EVs are a serious alternative soon for fuel-powered vehicles, it has been hardly competitive from the cost point of view [227]. Due to this factor, it has been difficult for the logistics and transportation companies to fully switch to the usage of electric vehicles instead of conventional internal combustion (IC) engine vehicles. Furthermore, the autonomous vehicular technologies are not fully mature yet, therefore a licensed driver is required to operate an electric vehicle whose wage may be higher due to the requirement to frequently visit the recharging stations. Therefore, there exists a trade-off between

EV and IC fleet for the logistics and transportation sectors now. As a result, the adoption of EVs for transportation systems may be gradual, particularly for centralized vehicle routing operations. The cooperative usage of conventional IC and EV fleet seems to be an essential first step towards the full-fledged usage of EVs in the future. However, the benefits of combining the usage of EV and IC fleet are yet to be fully studied in the context of dial-a-ride operations.

FIGURE 6.3: Calculation of required energy for the mixed fleet.

Besides, the energy consumption is not a linear function of travel time or distance (e.g., Erdogan and Miller-Hooks (2012) [228] and Schneider, Stenger, and Goeke (2014) [213]), but depend on several factors such as vehicle speed, mass, gradients, resistance, gravity, environmental temperature, and cargo load. Recently, the modeling of energy consumption has been studied in the literature (e.g., Bektas and Laporte, 2011 [229]; Jabali, Van Woensel, and de Kok, 2012 [230]). Figure 6.3 shows an energy consumption model that is based on the model of Goeke and Schneider (2015) [231] with an additional parameter to consider the effect of "outside temperature" which effects the battery capacity. In this model, the mechanical power determines energy consumption based on several factors (mass, speed, gradient, road surface, vehicle dimensions, engine properties, outside temperature).

Thereafter, the mechanical power can be translated into electrical power/fuel consumption. Realistic modeling of energy consumption ensures more efficient solutions or scheduling of visits to charging stations during the routing operations. In the literature, the impacts of outside temperature on battery capacity, the dependency between charge level and battery efficiency, and the non-linear relationship between charge level and recharging duration for EV in DARP-context are not thoroughly studied.

On the other hand, the simultaneous utilization of multiple recharging infrastructures has shown to be more efficient for electric vehicle routing when compared to using a single recharging technique. Kosmanos et al. 2018 [25] show the benefits of combining MED and SCS within the framework of the shortest path routing problem. Therefore, we encourage the researchers to focus on mathematical modeling for routing a mix of conventional and electric fleets in the dial-a-ride context and devise routing and scheduling algorithms while considering the features of multiple vehicular technologies and recharging infrastructures into account.

6.2.4 Accelerating the exact solution methods in GPU and solving multi-objective DARPs:

The NP-hard combinatorial optimization problems such as DARP and the CPU-based solution algorithms are time-intensive for practical applicability. This motivated the core idea of this thesis, in which we present a novel GPU-accelerated solution methodology for the DARP while integrating approximation methods to achieve high-quality solutions in a short time. However, there exists a great potential for adapting the proposed GPU-based methodology to accelerate the exact solution methods for optimally solving DARP.

Exact methods such as branch-and-bound (B&B) and branch-and-cut (B&C) algorithms [11][24][8] are efficient in solving DARP to optimality. In these exact algorithms, the search space is explored by dynamically building a tree whose root node is the original problem, the intermediate nodes are sub-problems, and the leaves are solutions. This algorithm runs for several iterations in which the best solution (lower bound) is progressively improved. During the exploration, some branches of the tree are cut or not explored if they are not likely to lead optimality. Although this mechanism allows a reduction of the size of the search space,

it still takes a very long time to attain optimal solutions for large-scale DARP instances [5].

FIGURE 6.4: GPU-accelerated branch-and-bound framework [3].

Parallel computing technologies (e.g., GPU) are turning out to be an essential tool for solving complex real-world problems in computer science. The challenge of adopting GPU for exact methods may involve a significant number of challenges [130] ranging from optimally mapping the data structures to designing efficient CUDA kernels. Melab et al. 2012 [3] developed a GPU-accelerated branch-and-bound for the flowshop scheduling problem. Figure 6.4 illustrates their implementation of B&B, in which the sub-problems are evaluated simultaneously using GPU cores. Despite the importance of attaining optimality using accelerated computing, this topic is under-explored and remains in the very beginning phase.

Therefore, we encourage the researchers to further develop GPU-based exact solution methods for solving different variants of the DARP. Besides, most of the exact solution methods in the literature are introduced for solving single-objective DARPs, but not multi-objective DARPs [5]. We encourage the researchers to further focus on the possibility of devising (GPU-based) exact methods or hybrids

of exact and heuristics (matheuristics) for solving multi-objective and complex variants of DARPs.

6.2.5 Improving the methodological aspects of the proposed GPU framework:

From the methodological perspective of GPU-based solution techniques, the new Cuda Dynamic Parallelism (CDP) feature of CUDA has a significant potential to enhance the kernel launch throughput thereby reducing the overall computational time. Dynamic parallelism offers flexibility to launch kernels inside another kernel without control transfer to CPU. The kernels in the first and second levels of depth are called parent kernel and child kernel, respectively. The parent kernel runs on GPU but launched from the CPU. The child kernel runs on GPU and it is launched from the parent kernel in GPU.

The CUDA kernels based on device streams exhibit significantly higher launch throughput for the concurrent kernels when compared to the CUDA kernels launched by the host streams. Deng and Mu (2013) [4] provided an illustration (shown in Figure 6.5) of cuda dynamic parallelism, in which each box indicates a kernel.

FIGURE 6.5: Cuda Dynamic Parallelism [4].

On the other hand, the global memory coalescing can significantly reduce the number of memory read/write operations in the GPU, and hence increase the overall speedup (e.g. Kang and Moon 2012 [232], Fauzia et al. 2015 [233], Kim 2017 [234]). The GPU executes instructions of each warp in Single Instruction Multiple Data (SIMD) fashion. A coalesced memory transaction is the one in which all the threads in a half-warp access global memory at the same time (i.e., consecutive threads access consecutive memory addresses). Designing CUDA kernels for efficient global memory coalescing is a very interesting future direction to accelerate DARP applications. It is also interesting to explore the benefits of coding the GPU solvers in two different programming styles: array-of-structures and structure-of-arrays. Therefore, we encourage the researchers to focus on the methodological research in GPU-based optimization methods while considering the new CDP feature of CUDA and design kernels to tackle the global memory coalescing issue in order to improve computational efficiency for solving DARP.

Appendix A

Detailed Analysis of Computational Speed-up for Various GPU-based Metaheuristics

Tables [A.1](#), [A.2](#), and [A.3](#) provide detailed analysis of magnitude difference from the best speedup attained for the proposed GPU-based methodologies in comparison with their respective CPU counterparts on DARP instances of [\[7\]](#).

Case	n	m	TS											
			C_1	C_1+opt_1	C_1+opt_2	C_1+opt_3	C_1+opt_4	C_1+opt_5	P_1-opt_1	P_1-opt_2	P_1-opt_3	P_1-opt_4	P_1-opt_5	P_1
R1a	24	3	24	23	23	23	24	15	6	8	22	1	21	0
R2a	48	5	46	44	42	45	46	30	16	16	41	6	38	0
R3a	72	7	61	56	54	59	60	41	34	19	50	6	46	0
R4a	96	9	57	53	48	55	52	20	2	2	13	2	44	0
R5a	120	11	56	50	50	55	55	40	38	10	42	5	38	0
R6a	144	13	35	34	32	34	35	15	5	8	7	1	27	0
R7a	36	4	32	33	31	31	32	19	3	1	14	1	29	0
R8a	72	6	59	56	53	56	56	25	1	9	13	2	48	0
R9a	108	8	54	47	46	52	53	42	39	7	37	0	32	0
R10a	144	10	39	30	32	37	38	32	30	4	21	2	16	0
R1b	24	3	19	18	18	18	19	12	3	7	17	1	16	0
R2b	48	5	43	41	38	42	42	28	18	15	37	3	35	0
R3b	72	7	58	56	51	53	57	25	2	14	17	6	48	0
R4b	96	9	65	64	57	63	62	24	3	10	14	0	53	0
R5b	120	11	54	48	47	53	53	39	36	4	39	1	36	0
R6b	144	13	52	45	44	50	51	39	36	6	36	0	32	0
R7b	36	4	32	31	29	31	32	21	10	13	29	2	27	0
R8b	72	6	64	59	57	60	62	23	1	1	15	5	53	0
R9b	108	8	77	69	67	75	75	59	56	12	56	3	51	0
R10b	144	10	40	31	33	37	39	32	31	4	21	3	17	0
Avg.	86	8	48	44	43	46	47	29	19	8	27	2	35	0

TABLE A.1: Magnitude difference from the best speed up for the G-TS methodology.

Case	n	m	VNS											
			C_2	C_2+opt_1	C_2+opt_2	C_2+opt_3	C_2+opt_4	C_2+opt_5	P_2-opt_1	P_2-opt_2	P_2-opt_3	P_2-opt_4	P_2-opt_5	P_2
R1a	24	3	6	5	5	6	6	5	3	2	4	2	2	0
R2a	48	5	7	5	6	6	6	6	6	5	5	4	5	0
R3a	72	7	7	6	5	5	6	2	6	4	6	4	4	0
R4a	96	9	25	24	20	24	26	24	5	24	18	5	12	0
R5a	120	11	6	4	4	3	3	3	11	9	10	11	7	0
R6a	144	13	23	20	15	12	20	17	1	9	13	1	9	0
R7a	36	4	7	7	5	5	7	5	1	4	3	1	2	0
R8a	72	6	19	19	11	14	18	15	2	9	7	5	6	0
R9a	108	8	27	19	18	24	26	24	12	7	7	3	6	0
R10a	144	10	23	18	18	11	22	21	32	32	40	25	26	0
R1b	24	3	7	6	5	6	7	6	3	3	3	2	1	0
R2b	48	5	9	8	7	8	9	8	7	5	7	4	5	0
R3b	72	7	21	18	12	15	17	17	3	14	10	4	6	0
R4b	96	9	26	22	14	17	19	15	11	17	8	2	6	0
R5b	120	11	8	4	6	8	6	6	9	7	9	6	8	0
R6b	144	13	9	7	5	7	6	4	12	11	11	10	12	0
R7b	36	4	11	9	7	10	10	9	6	5	7	4	4	0
R8b	72	6	22	19	13	20	22	13	5	19	10	2	7	0
R9b	108	8	6	3	3	3	3	2	9	8	9	7	6	0
R10b	144	10	24	18	20	22	24	22	12	31	32	18	31	0
Avg.	86	8	15	12	10	11	13	11	8	11	11	6	8	0

TABLE A.2: Magnitude difference from the best speed up for the G-VNS methodology.

Case	n	m	ALNS											
			C_3	C_3+opt_1	C_3+opt_2	C_3+opt_3	C_3+opt_4	C_3+opt_5	P_3-opt_1	P_3-opt_2	P_3-opt_3	P_3-opt_4	P_3-opt_5	P_3
R1a	24	3	6	4	2	5	4	4	3	3	3	3	4	0
R2a	48	5	6	6	3	6	5	6	5	6	6	5	6	0
R3a	72	7	14	14	6	14	10	8	8	13	8	8	11	0
R4a	96	9	17	17	9	15	14	13	9	16	12	12	14	0
R5a	120	11	20	19	10	18	15	14	9	11	12	11	15	0
R6a	144	13	22	19	12	18	15	16	9	12	12	9	14	0
R7a	36	4	8	8	3	8	6	6	5	6	5	5	6	0
R8a	72	6	18	18	5	15	15	15	9	13	9	10	12	0
R9a	108	8	24	20	12	20	20	20	12	13	11	12	17	0
R10a	144	10	29	22	14	24	21	21	11	14	13	12	18	0
R1b	24	3	5	4	3	4	4	4	3	3	2	2	3	0
R2b	48	5	9	9	3	8	8	8	6	9	6	7	7	0
R3b	72	7	17	15	8	16	15	15	10	17	11	10	13	0
R4b	96	9	20	18	10	17	15	13	11	15	12	10	15	0
R5b	120	11	21	18	10	19	16	15	9	15	12	12	16	0
R6b	144	13	23	20	15	20	22	19	11	12	17	14	20	0
R7b	36	4	6	5	2	5	4	5	3	5	3	3	4	0
R8b	72	6	20	15	6	17	15	15	10	15	10	11	13	0
R9b	108	8	20	15	8	13	16	14	9	18	10	10	13	0
R10b	144	10	26	20	15	18	21	16	13	14	15	16	18	0
Avg.	86	8	17	14	8	14	13	12	8	12	9	9	12	0

TABLE A.3: Magnitude difference from the best speed up for the G-ALNS methodology.

Appendix B

Detailed Analysis of Fleet Size Minimization on Passenger Experiences

Tables [B.1–B.4](#) show the best solutions with fleet size minimization scheme attained by the GPU-based TS algorithm on various DARP instances. The stopping criterion was set as 3 min. Figures [B.1–B.11](#) illustrate the impacts of fleet size minimization scheme in fleet size, travel cost, vehicle idle time, passenger ride time, passenger idle time, vehicle early arrival time, vehicle late arrival time, and vehicle route duration.

Instance	Demand	Available fleet size	Optimized fleet size	Travel cost	Vehicle route duration	Vehicle idle time	Passenger ride time	Passenger idle time	Vehicle early arrival time	Vehicle late arrival time	Fleet size variation	Fleet savings (%)	Travel cost deviation (%)
R1a	24	3	2	196.40	791.36	114.96	889.74	2257.48	21.30	1981.82	-1	+33.33	+3.36
R2a	48	5	4	312.24	1673.23	400.99	2297.08	5045.72	224.48	3525.36	-1	+20.00	+3.62
R3a	72	7	6	554.46	2416.80	422.33	3386.67	7614.71	207.05	5375.55	-1	+14.28	+4.22
R4a	96	9	6	647.71	2770.42	202.71	4561.73	10977.4	34.51	7533.78	-3	+33.33	+13.59
R5a	120	11	8	688.51	3360.64	272.12	5810.06	13587.6	89.82	9145.20	-3	+27.27	+9.82
R6a	144	13	9	884.37	4013.50	249.13	7112.36	16205.3	21.93	10738.5	-4	+30.76	+12.62
R7a	36	4	3	327.85	1172.86	125.01	1360.13	3285.10	28.69	2649.73	-1	+25.00	+12.39
R8a	72	6	5	552.66	2117.81	125.15	3521.47	7591.86	27.32	5109.99	-1	+16.66	+13.29
R9a	108	8	8	708.64	3132.33	263.69	5269.97	11765.76	76.19	8302.64	0	0.00	+7.65
R10a	144	10	9	942.96	4130.19	307.22	6783.11	15339.78	136.48	10841.99	-1	+10.00	+10.70
R1b	24	3	2	171.73	799.68	147.95	1129.53	3647.17	29.20	2785.82	-1	+33.33	+4.42
R2b	48	5	3	337.87	1330.64	32.76	1854.46	5634.62	29.95	4789.01	-2	+40.00	+14.287
R3b	72	7	5	520.96	2031.93	70.96	3615.61	9792.42	31.43	7265.25	-2	+28.57	+7.45
R4b	96	9	6	637.93	2598.55	40.62	4312.97	13632.0	15.26	11120.2	-3	+33.33	+20.52
R5b	120	11	8	616.21	3417.06	400.84	5960.81	16301.9	75.65	11924.9	-3	+27.27	+6.74
R6b	144	13	9	777.63	3880.99	223.35	7523.28	19544.6	59.55	14243.6	-4	+30.76	+6.43
R7b	36	4	3	255.57	1032.78	57.20	1804.56	4659.68	17.85	3748.20	-3	+25.00	+2.97
R8b	72	6	5	477.83	2258.06	340.23	3599.88	9383.82	117.14	7329.49	-1	+16.66	+4.16
R9b	108	8	7	632.45	3088.54	296.08	5026.38	13821.6	98.81	10517.6	-1	+12.50	+6.57
R10b	144	10	9	854.07	3902.73	168.65	6775.08	18905.9	23.04	14494.1	-1	+10.00	+8.71
Total				11098.05	49920.10	4961.95	82594.88	208994.4	1365.65	153422.7			

TABLE B.1: Results on imposing fleet size minimization on PR DARP instances of [7].

Instance	Demand	Available fleet size	Optimized fleet size	Travel cost	Vehicle route duration	Vehicle idle time	Passenger ride time	Passenger idle time	Vehicle early arrival time	Vehicle late arrival time	Fleet size variation	Fleet savings (%)	Travel cost deviation (%)
a2-16	16	2	2	294.24	777.56	387.31	270.02	587.29	299.15	477.85	0	0.00	0.00
a2-20	20	2	2	344.83	1070.26	605.43	325.95	878.16	433.69	668.74	0	0.00	0.00
a2-24	24	2	2	431.12	1111.90	536.78	436.76	922.29	365.26	712.99	0	0.00	0.00
a3-24	24	3	2	387.27	839.27	307.99	368.26	954.12	140.39	713.37	-1	+33.33	+12.31
a3-30	30	3	3	494.84	1396.46	721.61	547.82	1078.30	456.27	762.04	0	0.00	0.00
a3-36	36	3	3	583.18	1674.68	875.49	658.76	1333.11	543.08	950.50	0	0.00	0.00
a4-32	32	4	3	493.9	1134.30	448.40	536.92	1248.03	254.19	948.76	-1	+25.00	+1.73
a4-40	40	4	3	601.47	1545.42	703.94	638.86	1558.10	458.63	1208.35	-1	+25.00	+7.85
a4-48	48	4	3	697.06	1927.22	942.15	794.95	1913.80	598.92	1410.77	-1	+25.00	+4.22
a5-40	40	5	3	541.06	1257.79	476.72	606.97	1612.34	240.99	1269.75	-2	+40.00	+8.56
a5-50	50	5	3	710.81	1488.20	477.38	945.48	1998.31	289.52	1615.49	-2	+40.00	+3.52
a5-60	60	5	4	849.99	2241.23	1031.24	1068.49	2395.71	626.72	1815.90	-1	+20.00	+5.14
a6-48	48	6	4	623.32	1419.10	507.77	822.76	1891.51	253.92	1443.54	-2	+33.33	+3.18
a6-60	60	6	4	858.76	2078.20	859.44	1122.05	2201.10	567.30	1746.92	-2	+33.33	+4.82
a6-72	72	6	4	971.67	2475.54	1071.86	1230.08	2684.42	652.89	2000.74	-2	+33.33	+6.07
a7-56	56	7	4	779.67	1653.04	537.36	1011.55	2151.61	258.51	1633.84	-3	+42.85	+7.68
a7-70	70	7	4	989.71	2069.33	659.61	1313.63	2750.22	384.91	2214.26	-3	+42.85	+11.31
a7-84	84	7	4	1120.61	2435.04	810.43	1483.31	3253.14	384.88	2505.18	-3	+42.85	+8.42
a8-64	64	8	5	780.54	2045.73	881.19	1160.56	2407.10	581.75	1853.81	-3	+37.50	+4.43
a8-80	80	8	4	1007.35	2258.37	771.01	1411.35	3113.78	379.13	2392.12	-4	+50.00	+6.50
a8-96	96	8	5	1317.98	3155.64	1261.66	1762.16	3854.94	677.29	2920.54	-3	+37.50	+7.11
Total				14879.50	36054.41	14874.90	18516.77	40787.48	8847.50	31265.56			

TABLE B.2: Results on imposing fleet size minimization on Set A DARP instances of [8].

Instance	Demand	Available fleet size	Optimized fleet size	Travel cost	Vehicle route duration	Vehicle idle time	Passenger ride time	Passenger idle time	Vehicle early arrival time	Vehicle late arrival time	Fleet size variation	Fleet savings (%)	Travel cost deviation (%)
b2-16	16	2	2	309.41	791.86	386.45	307.61	735.65	254.75	576.81	0	0.00	0.00
b2-20	20	2	2	332.64	958.80	458.16	226.75	828.13	197.90	567.86	0	0.00	0.00
b2-24	24	2	2	444.71	1094.59	475.88	391.36	1111.43	203.18	803.82	0	0.00	0.00
b3-24	24	3	3	394.51	1181.08	612.57	307.92	1075.60	291.97	742.00	0	0.00	0.00
b3-30	30	3	3	531.44	1533.93	788.49	510.33	1468.66	398.41	1063.59	0	0.00	0.00
b3-36	36	3	3	605.68	1789.37	917.69	577.93	1710.25	540.33	1286.38	0	0.00	+0.31
b4-32	32	4	3	525.22	1315.21	545.99	408.89	1313.89	294.88	1046.70	-1	+25.00	+6.14
b4-40	40	4	4	663.77	2081.31	1101.54	637.15	1800.63	637.94	1256.45	0	0.00	+1.09
b4-48	48	4	4	692.80	2413.50	1408.70	682.72	2122.77	921.80	1530.60	0	0.00	+2.76
b5-40	40	5	4	619.77	1567.22	691.45	688.57	1851.76	319.72	1367.62	-1	+20.00	+0.99
b5-50	50	5	5	781.10	2487.70	1342.60	817.24	2280.82	760.10	1561.40	0	0.00	+2.59
b5-60	60	5	5	969.80	2923.28	1575.49	941.16	2697.62	931.24	1948.27	0	0.00	+7.51
b6-48	48	6	6	726.50	2277.57	1185.07	754.78	2174.29	623.81	1506.63	0	0.00	+1.63
b6-60	60	6	5	884.85	2549.34	1230.48	1073.86	3075.10	618.52	2277.20	-1	+17.00	+2.88
b6-72	72	6	5	1034.92	3245.08	1708.17	1195.04	3375.74	1059.91	2544.86	-1	+17.00	+5.75
b7-56	56	7	6	864.45	2219.28	954.83	1007.45	2770.76	444.73	2034.40	-1	+14.00	+4.86
b7-70	70	7	6	967.41	2750.92	1251.50	1039.60	3109.66	770.31	2419.13	-1	+14.00	+6.00
b7-84	84	7	7	1342.12	4364.85	2458.72	1367.27	3951.74	1586.05	2901.56	0	0.00	+11.53
b8-64	64	8	7	862.30	2617.50	1293.19	1069.38	3122.72	698.34	2312.23	-1	+13.00	+2.60
b8-80	80	8	6	1107.50	2974.86	1259.35	1311.61	3480.14	755.77	2678.33	-2	+25.00	+6.83
b8-96	96	8	8	1331.05	4827.40	2862.35	1583.93	4707.68	1926.96	3443.41	0	0.00	+12.14
Total				15991.95	47964.64	24508.69	16900.54	48765.05	14236.63	35869.25			

TABLE B.3: Results on imposing fleet size minimization on Set B DARP instances of [8].

FIGURE B.1: Fleet size for various instances.

FIGURE B.2: Travel Cost for various instances.

Improvement (%) achieved by the proposed HTS vs BKS [9][10][11][12] on various DARP benchmark instances [7][8]					
Quantifier	Average over PR instances	Average over Set-A instances	Average over Set-B instances	Cumulative Average	
Fleet size	+19.73	+29.70	+8.57	+19.08	
Travel cost	-9.19	-5.18	-4.46	-6.11	
Total vehicle route duration	+8.78	+22.93	+4.88	+11.84	
Total vehicle idle time	+50.40	+43.59	+11.48	+30.78	
Total passenger ride time	+5.28	+0.30	+6.09	+4.65	
Total passenger idle time	+1.37	+0.40	+2.64	+1.45	
Total vehicle early arrival time	+67.98	+53.30	+14.25	+38.59	
Total vehicle late arrival time	-2.04	-4.00	-1.14	-2.19	

TABLE B.4: Improvement attained by the fleet size minimization scheme when compared to the best-solutions without fleet size minimization [9][10][11][12] on single-objective DARP.

FIGURE B.3: DARP instance Set PR: Relationship between travel cost and fleet size.

FIGURE B.4: DARP instance Set A: Relationship between travel cost and fleet size.

FIGURE B.5: DARP instance Set B: Relationship between travel cost and fleet size.

FIGURE B.6: Vehicle Route Duration: Quantitative analysis of solutions for various instances.

FIGURE B.7: Vehicle Idle Time: Quantitative analysis of solutions for various instances.

FIGURE B.8: Passenger Ride Time: Quantitative analysis of solutions for various instances.

FIGURE B.9: Passenger Idle Time: Quantitative analysis of solutions for various instances.

FIGURE B.10: Vehicle Early Arrival Time: Quantitative analysis of solutions for various instances.

FIGURE B.11: Vehicle Late Arrival Time: Quantitative analysis of solutions for various instances.

Appendix C

Detailed Solutions for Electric Autonomous Dial-A-Ride Problems

Tables [C.1](#) and [C.2](#) show the detailed best solution attained by G-ALNS with static charging stations (SCS) for e-ADARP instances. Tables [C.3](#) and [C.4](#) show the detailed best solution attained by G-ALNS with mobile energy disseminators (MED) for e-ADARP instances. The stopping criterion was set as 2 min. The problem instances and BKS are provided by [\[24\]](#).

Case	Size	m	LB	Gap	Obj.	TC	ERT	RD	VWT	PRT
a2-16-0.1	32	2	237.38	0.00	237.38	299.26	51.73	804.28	409.02	239.23
a2-20-0.1	40	2	279.08	0.00	279.08	347.89	72.63	1064.68	596.79	278.36
a2-24-0.1	48	2	346.21	0.00	346.21	433.26	85.08	1119.22	541.96	360.30
a3-18-0.1	36	3	236.81	0.00	236.81	304.34	34.24	674.24	261.90	225.73
a3-24-0.1	48	3	274.80	0.00	274.80	355.78	31.87	1174.91	675.13	265.57
a3-30-0.1	60	3	413.27	0.00	413.27	500.69	150.99	1405.37	724.67	480.66
a3-36-0.1	72	3	481.72	0.00	481.72	601.58	122.12	1719.68	902.10	529.82
a4-16-0.1	32	4	222.49	0.00	222.49	286.45	30.60	543.07	160.62	210.95
a4-24-0.1	48	4	310.84	0.00	310.84	388.83	76.88	1087.16	554.33	337.47
a4-32-0.1	64	4	393.96	0.00	393.95	502.34	68.80	1332.86	638.52	417.00
a4-40-0.1	80	4	453.84	0.00	453.84	562.44	128.04	1916.45	1114.01	544.32
a5-40-0.1	80	5	414.51	0.00	414.51	524.75	83.78	1851.61	1086.86	472.84
Avg.	53	3	338.74	0.00	338.74	425.63	78.06	1224.46	638.83	363.52
a2-16-0.4	32	2	237.38	0.00	237.38	299.26	51.73	869.79	424.02	239.23
a2-20-0.4	40	2	280.70	0.00	280.70	350.06	72.63	1114.05	594.62	278.36
a2-24-0.4	48	2	348.04	-0.29	347.04	434.36	85.08	1199.39	440.91	360.30
a3-18-0.4	36	3	236.82	0.00	236.81	304.34	34.24	674.24	261.90	225.73
a3-24-0.4	48	3	274.80	0.00	274.80	355.78	31.87	1233.81	684.93	265.57
a3-30-0.4	60	3	413.37	0.00	413.37	500.83	150.99	1435.12	671.60	480.66
a3-36-0.4	72	3	484.14	0.00	484.14	615.10	91.29	1890.91	911.14	498.99
a4-16-0.4	32	4	222.49	0.00	222.49	286.45	30.60	384.83	160.62	210.95
a4-24-0.4	48	4	311.03	0.00	311.03	388.83	76.88	1087.16	556.89	337.47
a4-32-0.4	64	4	394.26	0.00	394.26	502.34	68.80	1394.48	650.02	417.00
a4-40-0.4	80	4	453.84	0.00	453.84	562.44	128.04	1958.10	1114.01	544.32
a5-40-0.4	80	5	414.51	0.00	414.50	524.75	83.78	1851.61	1086.86	472.84
Avg.	53	3	339.28	-0.02	339.20	427.04	75.49	1257.79	629.80	360.95
a2-16-0.7	32	2	240.66	0.00	240.66	303.64	51.73	923.28	361.19	239.23
a2-20-0.7	40	2	287.17	2.18	293.42	350.06	72.63	1084.49	614.43	278.36
a2-24-0.7	48	2	358.21	-1.40	353.18	442.48	85.08	1310.27	387.20	360.30
a3-18-0.7	36	3	240.58	0.00	240.58	308.63	34.24	803.29	218.78	225.73
a3-24-0.7	48	3	277.72	-0.63	275.97	357.33	31.87	1248.81	555.56	265.57
a3-30-0.7	60	3	417.06	2.14	425.98	510.05	173.77	1594.49	619.36	503.44
a3-36-0.7	72	3	485.91	2.32	497.20	617.88	123.61	1955.25	646.42	531.31
a4-16-0.7	32	4	223.13	0.00	223.13	282.68	44.49	573.75	164.39	224.84
a4-24-0.7	48	4	318.19	-0.55	316.44	401.69	60.68	1135.81	380.20	321.28
a4-32-0.7	64	4	387.99	3.67	402.22	514.45	65.33	1553.48	549.68	413.52
a4-40-0.7	80	4	443.62	12.30	498.16	616.86	126.27	2015.92	720.18	542.56
a5-40-0.7	80	5	405.99	3.20	419.00	526.20	96.43	2091.79	1038.58	485.49
Avg.	53	3	340.52	1.93	348.83	436.00	80.51	1357.55	521.33	365.97

Obj.: Objective function of the best solution; TC: Total travel time of EV fleet; ERT: Total excess user ride time, RD: Total route duration of EV fleet; VWT: Total waiting time of EV fleet; PRT: Total passenger ride time.

TABLE C.1: Benchmark instances: Details of the best solution attained by ALNS[GPU] for e-ADARP with SCS.

Case	Size	m	LB	Gap	Obj.	TC	ERT	RD	VWT	PRT
u2-16-0.1	32	2	57.61	0.00	57.61	76.81	0.00	235.49	131.56	41.96
u2-20-0.1	40	2	55.59	0.00	55.59	73.70	1.24	299.66	188.77	35.94
u2-24-0.1	48	2	91.27	-0.40	90.91	118.74	7.43	350.17	156.59	70.33
u3-18-0.1	36	3	50.74	0.00	50.74	67.65	0.00	358.91	273.25	34.78
u3-24-0.1	48	3	67.56	0.00	67.56	86.08	12.01	436.12	326.05	55.92
u3-30-0.1	60	3	76.75	0.00	76.75	100.83	4.50	505.04	374.21	53.86
u3-36-0.1	72	3	104.04	0.00	104.04	133.79	14.80	702.27	496.15	88.73
u4-16-0.1	32	4	53.58	-1.18	52.95	67.91	8.06	149.83	62.71	39.31
u4-24-0.1	48	4	89.83	0.01	89.84	118.20	4.67	782.36	590.83	65.64
u4-32-0.1	64	4	99.29	0.06	99.35	129.27	9.59	528.02	356.79	81.02
u4-40-0.1	80	4	133.11	0.44	133.70	169.20	27.25	1480.40	1258.44	133.86
u5-40-0.1	80	5	114.12	7.98	123.23	156.54	23.31	820.30	583.80	120.09
Avg.	53	3	82.79	0.58	83.52	108.23	9.40	554.05	399.93	68.45
u2-16-0.4	32	2	57.65	0.00	57.65	76.86	0.00	252.86	131.56	41.96
u2-20-0.4	40	2	56.34	0.00	56.34	74.70	1.24	321.55	187.77	35.94
u2-24-0.4	48	2	91.63	0.00	91.63	117.46	14.14	383.75	149.01	77.04
u3-18-0.4	36	3	50.74	0.00	50.74	67.65	0.00	382.77	273.25	34.78
u3-24-0.4	48	3	67.56	0.00	67.56	86.08	12.01	436.12	326.05	55.92
u3-30-0.4	60	3	76.75	0.00	76.75	100.83	4.50	574.33	374.21	53.86
u3-36-0.4	72	3	104.06	0.83	104.92	133.37	19.57	757.92	500.12	93.50
u4-16-0.4	32	4	53.58	-1.18	52.95	67.91	8.06	149.83	62.71	39.31
u4-24-0.4	48	4	89.83	-0.04	89.79	118.16	4.67	1272.43	801.16	65.64
u4-32-0.4	64	4	99.29	0.66	99.95	130.08	9.59	654.82	398.09	81.02
u4-40-0.4	80	4	133.91	0.73	134.89	173.43	19.27	1762.48	1383.55	125.88
u5-40-0.4	80	5	112.58	9.98	123.81	156.27	26.42	801.65	557.00	123.21
Avg.	53	3	82.83	0.91	83.92	108.57	9.96	645.88	428.71	69.01
u2-16-0.7	32	2	59.19	0.00	59.19	78.93	0.00	242.70	77.26	41.96
u2-20-0.7	40	2	56.86	0.00	56.86	75.40	1.24	321.55	156.57	35.94
u2-24-0.7	48	2	90.83	2.12	92.76	121.21	7.43	380.04	80.69	70.33
u3-18-0.7	36	3	50.99	0.00	50.99	67.99	0.00	424.34	234.03	34.78
u3-24-0.7	48	3	68.39	0.88	68.99	86.69	15.90	499.09	277.72	59.81
u3-30-0.7	60	3	78.14	-0.08	78.08	102.06	6.29	579.10	301.24	55.64
u3-36-0.7	72	3	104.37	2.01	106.47	135.44	19.57	746.26	392.10	93.50
u4-16-0.7	32	4	53.87	0.19	53.97	69.28	8.06	248.05	53.64	39.31
u4-24-0.7	48	4	89.96	0.01	89.97	118.40	4.67	1308.50	827.75	65.64
u4-32-0.7	64	4	99.50	2.45	101.94	133.06	8.59	693.07	353.79	80.02
u4-40-0.7	80	4	133.01	3.51	137.68	176.17	22.20	1750.58	1320.17	128.81
u5-40-0.7	80	5	109.28	16.04	126.81	161.31	23.31	665.58	456.27	120.09
Avg.	53.33	3.25	82.87	2.26	85.31	110.49	9.77	654.91	377.60	68.82

Obj.: Objective function of the best solution; TC: Total travel time of EV fleet; ERT: Total excess user ride time, RD: Total route duration of EV fleet; VWT: Total waiting time of EV fleet; PRT: Total passenger ride time.

TABLE C.2: UBER instances: Details of the best solution attained by ALNS[GPU] for e-ADARP with SCS.

Case	Size	m	LB	Gap	Obj.	TC	ERT	RD	VWT	PRT	MED-m	MED-TC	MED-RD
a2-16-0.1	32	2	237.38	0.00	237.38	299.26	51.73	804.28	409.02	239.23	3	0.00	0.00
a2-20-0.1	40	2	279.08	0.00	279.08	347.89	72.63	1064.68	596.79	278.36	3	0.00	0.00
a2-24-0.1	48	2	346.21	0.00	346.21	433.26	85.08	1119.22	541.96	360.30	3	0.00	0.00
a3-18-0.1	36	3	236.81	0.00	236.81	304.34	34.24	674.24	261.90	225.73	3	0.00	0.00
a3-24-0.1	48	3	274.80	0.00	274.80	355.78	31.87	1174.91	675.13	265.57	3	0.00	0.00
a3-30-0.1	60	3	413.27	0.00	413.27	500.69	150.99	1405.37	724.67	480.66	3	0.00	0.00
a3-36-0.1	72	3	481.72	0.00	481.72	601.58	122.12	1719.68	902.10	529.82	3	0.00	0.00
a4-16-0.1	32	4	222.49	0.00	222.49	286.45	30.60	543.07	160.62	210.95	3	0.00	0.00
a4-24-0.1	48	4	310.84	0.00	310.84	388.83	76.88	1087.16	554.33	337.47	3	0.00	0.00
a4-32-0.1	64	4	393.96	0.00	393.96	502.35	68.80	1332.86	638.52	417.00	3	0.00	0.00
a4-40-0.1	80	4	453.84	0.00	453.84	562.44	128.04	1916.45	1114.01	544.32	3	0.00	0.00
a5-40-0.1	80	5	414.51	0.00	414.50	524.75	83.78	1851.61	1086.86	472.84	3	0.00	0.00
Avg.	80	5	338.74	0.00	338.74	425.63	78.06	1224.46	638.83	363.52	3	0.00	0.00
a2-16-0.4	32	2	237.38	0.00	237.38	299.26	51.73	819.28	424.02	239.23	3	12.72	81.55
a2-20-0.4	40	2	280.70	-0.58	279.08	347.89	72.63	1069.49	601.60	278.36	3	38.03	120.96
a2-24-0.4	48	2	348.04	-0.53	346.21	433.26	85.08	1132.12	554.87	360.30	3	22.52	177.06
a3-18-0.4	36	3	236.82	0.00	236.81	304.34	34.24	674.24	261.90	225.73	3	0.00	0.00
a3-24-0.4	48	3	274.80	0.00	274.80	355.78	31.87	1184.71	684.93	265.57	3	8.91	91.01
a3-30-0.4	60	3	413.37	-0.02	413.27	500.69	150.99	1418.23	737.53	480.66	3	30.18	102.10
a3-36-0.4	72	3	484.14	-0.50	481.72	601.58	122.12	1900.68	917.10	529.82	3	36.55	271.18
a4-16-0.4	32	4	222.49	0.00	222.49	286.45	30.60	543.07	160.62	210.95	3	0.00	0.00
a4-24-0.4	48	4	311.03	-0.06	310.84	388.83	76.88	1089.72	556.89	337.47	3	10.64	27.55
a4-32-0.4	64	4	394.26	-0.08	393.95	502.34	68.80	1382.57	638.52	417.00	3	23.46	76.17
a4-40-0.4	80	4	453.84	0.00	453.84	562.44	128.04	1917.89	1115.46	544.32	3	11.88	23.45
a5-40-0.4	80	5	414.51	0.00	414.50	524.75	83.78	1851.61	1086.86	472.84	3	0.00	0.00
Avg.	53	3	339.28	-0.15	338.74	425.63	78.06	1248.63	645.02	363.52	3	16.24	80.92
a2-16-0.7	32	2	240.66	-1.36	237.38	299.26	51.73	819.28	424.02	239.23	3	59.32	269.62
a2-20-0.7	40	2	287.17	-2.25	280.70	350.06	72.63	1084.49	614.43	278.36	3	68.89	298.75
a2-24-0.7	48	2	358.21	-3.35	346.21	433.26	85.08	1243.01	554.16	360.30	3	55.30	345.73
a3-18-0.7	36	3	240.58	-1.57	236.81	304.34	34.24	704.24	291.90	225.73	3	36.61	61.61
a3-24-0.7	48	3	277.72	-1.05	274.80	355.78	31.87	1199.71	699.93	265.57	3	36.39	244.86
a3-30-0.7	60	3	417.06	-0.34	415.63	503.84	150.99	1572.83	756.44	480.66	3	62.56	421.21
a3-36-0.7	72	3	485.91	0.70	489.31	611.70	122.12	1955.25	907.52	529.82	3	84.10	634.77
a4-16-0.7	32	4	223.13	-0.29	222.49	286.45	30.60	588.75	175.62	210.95	3	28.58	104.61
a4-24-0.7	48	4	318.19	-2.14	311.38	394.94	60.68	1110.60	571.66	321.28	3	61.58	255.12
a4-32-0.7	64	4	387.99	1.74	394.74	503.38	68.80	1433.76	738.38	417.00	3	57.79	229.90
a4-40-0.7	80	4	443.62	3.89	460.89	572.99	124.58	1772.12	959.13	540.86	3	63.44	422.24
a5-40-0.7	80	5	405.99	2.39	415.71	527.08	81.60	1764.85	997.77	470.66	3	50.40	282.09
Avg.	53	3	340.52	-0.30	340.50	428.59	76.24	1270.74	640.91	361.70	3	55.41	297.54

TABLE C.3: Benchmark instances: Details of the best solution attained by ALNS[GPU] for e-ADARP with MED.

Obj.: Objective function of the best solution; TC: Total travel time of EV fleet; ERT: Total excess user ride time;
RD: Total route duration of EV fleet; VWT: Total waiting time of EV fleet; PRT: Total passenger ride time;
MED-m: MED fleet size; MED-TC: Total travel time of MED fleet; MED-RD: Total route duration of MED fleet.

Case	Size	m	LB	Gap	Obj.	TC	ERT	RD	VWT	PRT	MED-m	MED-TC	MED-RD
u2-16-0.1	32	2	57.61	0.00	57.61	76.81	0.00	225.33	132.51	41.96	5	2.24	23.51
u2-20-0.1	40	2	55.59	0.00	55.59	73.70	1.24	297.47	203.77	35.94	5	5.22	20.56
u2-24-0.1	48	2	91.27	-1.76	89.66	114.83	14.14	342.34	203.51	77.04	5	11.80	47.75
u3-18-0.1	36	3	50.74	0.00	50.74	67.65	0.00	358.91	273.25	34.78	5	0.00	0.00
u3-24-0.1	48	3	67.56	0.00	67.56	86.08	12.01	436.12	326.05	55.92	5	0.00	0.00
u3-30-0.1	60	3	76.75	0.00	76.75	100.83	4.50	505.04	374.21	53.86	5	0.00	0.00
u3-36-0.1	72	3	104.04	-0.52	103.50	131.48	19.57	680.94	513.45	93.50	5	3.81	58.09
u4-16-0.1	32	4	53.58	-1.18	52.95	67.91	8.06	149.83	62.71	39.31	5	0.00	0.00
u4-24-0.1	48	4	89.83	-0.19	89.66	117.99	4.67	1527.38	957.32	65.64	5	3.65	148.90
u4-32-0.1	64	4	99.29	0.14	99.43	129.38	9.59	518.06	356.68	81.02	5	0.00	0.00
u4-40-0.1	80	4	133.11	0.02	133.14	168.44	27.25	816.51	1387.83	133.86	5	21.84	369.10
u5-40-0.1	80	5	114.12	7.15	122.28	154.24	26.42	0.00	487.95	123.21	5	13.35	79.16
Avg.	53	3	82.79	0.31	83.24	107.45	10.62	488.16	439.94	69.67	5	5.16	62.26
u2-16-0.4	32	2	57.65	-0.07	57.61	76.81	0.00	225.23	132.51	41.96	5	6.20	30.61
u2-20-0.4	40	2	56.34	-1.33	55.59	73.70	1.24	297.47	203.77	35.94	5	8.17	56.36
u2-24-0.4	48	2	91.63	-2.15	89.66	114.83	14.14	383.75	203.51	77.04	5	22.95	75.68
u3-18-0.4	36	3	50.74	0.00	50.74	67.65	0.00	380.91	295.25	34.78	5	4.22	28.48
u3-24-0.4	48	3	67.56	0.00	67.56	86.08	12.01	480.57	370.49	55.92	5	14.72	139.39
u3-30-0.4	60	3	76.75	0.00	76.75	100.83	4.50	526.90	396.07	53.86	5	8.38	70.98
u3-36-0.4	72	3	104.06	-0.54	103.50	131.48	19.57	709.17	528.45	93.50	5	9.72	147.79
u4-16-0.4	32	4	53.58	-1.18	52.95	67.91	8.06	149.83	62.71	39.31	5	0.00	0.00
u4-24-0.4	48	4	89.83	-0.19	89.66	117.99	4.67	1262.64	816.34	65.64	5	13.77	365.27
u4-32-0.4	64	4	99.29	0.14	99.43	129.38	9.59	530.96	369.59	81.02	5	9.56	86.09
u4-40-0.4	80	4	133.91	-0.58	133.14	168.44	27.25	1770.81	1387.83	133.86	5	21.84	369.10
u5-40-0.4	80	5	112.58	8.62	122.28	154.24	26.42	660.38	458.15	123.21	5	0.00	0.00
Avg.	53	3	82.83	0.23	83.24	107.45	10.62	614.89	435.39	69.67	5	9.96	114.14
u2-16-0.7	32	2	59.19	-2.67	57.61	76.81	0.00	238.86	146.05	41.96	5	10.39	70.69
u2-20-0.7	40	2	56.83	-2.23	55.59	73.70	1.24	302.67	208.97	35.94	5	11.01	76.62
u2-24-0.7	48	2	90.83	-1.29	89.66	114.83	14.14	154.06	203.51	77.04	5	38.38	154.06
u3-18-0.7	36	3	50.99	-0.49	50.74	67.65	0.00	395.91	310.25	34.78	5	11.44	88.27
u3-24-0.7	48	3	68.39	-1.21	67.56	86.08	12.01	480.57	370.49	55.92	5	14.72	139.39
u3-30-0.7	60	3	78.14	-1.78	76.75	100.83	4.50	505.04	374.21	53.86	5	0.00	0.00
u3-36-0.7	72	3	104.37	-0.83	103.50	131.48	19.57	705.82	538.34	93.50	5	23.44	214.68
u4-16-0.7	32	4	53.87	-1.71	52.95	67.91	8.06	244.82	69.28	39.31	5	10.02	125.46
u4-24-0.7	48	4	89.96	-0.33	89.66	117.99	4.67	1096.67	949.89	65.64	5	14.57	266.17
u4-32-0.7	64	4	99.50	-0.07	99.43	129.38	9.59	644.47	376.48	81.02	5	21.57	195.12
u4-40-0.7	80	4	133.01	0.10	133.14	168.44	27.25	1770.81	1387.83	133.86	5	21.84	369.10
u5-40-0.7	80	5	109.28	12.55	122.99	155.18	26.42	907.58	497.38	123.21	5	24.52	253.10
Avg.	53	3	82.87	0.00	83.30	107.52	10.62	620.61	452.72	69.67	5	16.82	162.72

Obj.: Objective function of the best solution; TC: Total travel time of EV fleet; ERT: Total excess user ride time;
RD: Total route duration of EV fleet; VWT: Total waiting time of EV fleet; PRT: Total passenger ride time;
MED-m: MED fleet size; MED-TC: Total travel time of MED fleet; MED-RD: Total route duration of MED fleet.

TABLE C.4: UBER instances: Details of the best solution attained by ALNS[GPU] for e-ADARP with MED.

Appendix D

Detailed Analysis of Depot Optimization in Dial-A-Ride Problems

Here, we present a brief overview of our bilevel optimization framework for DARP with depot optimization, which consists of two levels of optimization layers: the outer layer optimizes the depot locations and the inner layer optimizes the route plan. A new deterministic annealing (DA) meta-heuristic is developed for depot location optimization. The objective here is to minimize the travel cost, which we consider as the travel time of the passengers. Here, we define a solution as a combination of two variables: depot locations (outer layer) and route plan (inner layer). For the outer layer, we use the proposed deterministic annealing for depot location optimization, and the inner layer uses a simulated annealing heuristic for route optimization.

Deterministic annealing is a variant of simulated annealing meta-heuristic for solving optimization problems. In this approach, a temperature-controlled random walk is performed to identify the optimal combination of depot locations that yield a route plan with minimum travel cost. On each iteration, the next solution is selected by performing a local search from the current solution, and the acceptance decision is based on the temperature of the system. There are two types of acceptance: metropolis and threshold criterion. Previous studies have shown

that SA has higher computational complexity due to the utilization of a metropolis function. Therefore, we employ DA to deterministically optimize the system. A certain number of iterations is carried out at each temperature, in turn, the algorithm avoids getting trapped at local minimum solutions while exploring the vast amount of search space in an attempt to find the global optimal solution. An overview of the proposed deterministic annealing is outlined in Fig. D.1. For full details of the algorithms, the readers are referred to our conference paper in [115].

Figures D.2–D.7 show the effect of depot optimization in reducing the travel cost for the various DARP instances. Tables D.1–D.4 show the detailed results on the solutions attained for the depot optimization problem in the dial-a-ride problems.

Instance	BKS	Deterministic annealing								Travel cost
		Depot			Vehicle					
		Label	Coordinate x	Coordinate y	Tag	Departure	Arrival	Passengers	Duration	
a2-16 [8]	294.24 [8]	d_1	5.40	-1.14	v_1	d_1	d_2	10	395.06	289.18
		d_2	-5.20	-1.15	v_2	d_2	d_2	6	377.44	
b2-16 [8]	309.41 [8]	d_1	4.28	-4.02	v_1	d_1	d_2	6	407.97	295.14
		d_2	-4.40	2.73	v_2	d_2	d_1	10	276.02	
a2-20 [8]	344.83 [8]	d_1	-4.06	1.77	v_1	d_1	d_2	14	565.83	335.68
		d_2	6.04	-2.68	v_2	d_2	d_2	6	467.03	
b2-20 [8]	332.64 [8]	d_1	2.34	3.18	v_1	d_1	d_2	16	450.70	331.13
		d_2	-2.80	-3.05	v_2	d_2	d_2	12	506.59	

TABLE D.1: Results attained by the proposed DA on small-scale DARP instances [8].

Instance	BKS	Deterministic annealing								Travel cost
		Depot			Vehicle					
		Label	Coordinate x	Coordinate y	Tag	Departure	Arrival	Passengers	Duration	
a2-24 [8]	431.12 [8]	d_1	-4.18	1.62	v_1	d_1	d_2	12	554.20	418.29
		d_2	6.80	3.14	v_2	d_2	d_1	12	529.53	
b2-24 [8]	444.71 [8]	d_1	-0.73	-5.85	v_1	d_1	d_1	12	526.77	435.33
		d_2	1.51	6.66	v_2	d_2	d_2	12	558.44	
R1a [7]	190.02 [9]	d_1	-1.69	6.36	v_1	d_1	d_3	9	346.92	178.36
		d_2	-0.26	-0.16	v_2	d_2	d_2	2	441.37	
		d_2	0.40	0.65	v_2	d_3	d_3	13	81.20	
R1b [7]	164.46 [9]	d_1	2.80	2.53	v_1	d_1	d_1	8	255.50	158.62
		d_2	-5.12	6.28	v_2	d_2	d_3	8	411.14	
		d_2	-3.50	-0.76	v_2	d_3	d_3	8	363.02	

TABLE D.2: Results attained by the proposed DA on medium-scale DARP instances [7][8].

Instance	BKS	Deterministic annealing										Travel cost
		Depot				Vehicle						
		Label	Coordinate x	Coordinate y	Tag	Departure	Arrival	Passengers	Duration			
R2a [7]	301.33 [9]	d_1	1.22	1.90	v_1	d_1	d_3	8	386.11	285.76		
		d_2	-3.75	-6.10	v_2	d_2	d_1	8	378.49			
		d_3	5.56	6.08	v_3	d_2	d_5	14	406.29			
		d_4	-4.02	4.39	v_4	d_2	d_5	6	340.00			
		d_5	5.08	-0.98	v_5	d_2	d_1	12	457.46			
R2b [7]	295.66 [9]	d_1	5.56	6.08	v_1	d_1	d_1	12	398.44	278.81		
		d_2	-4.02	4.39	v_2	d_2	d_4	9	372.94			
		d_3	5.08	-0.98	v_3	d_3	d_4	14	373.46			
		d_4	1.22	1.90	v_4	d_4	d_1	6	186.71			
		d_5	-3.75	-6.10	v_5	d_5	d_4	7	186.05			
R3a [7]	532.42 [9]	d_1	-4.75	-2.90	v_1	d_1	d_7	13	462.89	525.39		
		d_2	0.80	-3.88	v_2	d_2	d_5	13	401.18			
		d_3	-4.38	6.18	v_3	d_3	d_2	11	370.31			
		d_4	6.45	-4.88	v_4	d_4	d_7	8	375.83			
		d_5	0.71	1.98	v_5	d_5	d_5	14	393.58			
		d_6	-7.43	2.01	v_6	d_6	d_2	13	458.40			
		d_7	0.00	0.00	v_7	d_7	d_7	0	0.00			
R3b [7]	484.82 [10]	d_1	2.87	0.10	v_1	d_1	d_6	8	360.72	456.35		
		d_2	-0.77	3.24	v_2	d_2	d_5	9	325.52			
		d_3	-5.07	2.64	v_3	d_3	d_7	13	372.26			
		d_4	-0.22	-5.20	v_4	d_4	d_5	9	365.55			
		d_5	-6.56	4.78	v_5	d_5	d_2	9	362.48			
		d_6	6.19	-5.56	v_6	d_6	d_3	11	356.50			
		d_7	0.00	0.00	v_7	d_7	d_2	13	383.05			

TABLE D.3: Results attained by the proposed DA on large-scale DARP instances [7].

Problem	Instance	Demand	Fleet	BKS			Deterministic annealing		
				Depots	Travel cost	Savings (%)	Depots	Travel cost	Savings (%)
Small-scale	a2-16 [8]	16	2	1	294.24 [8]	0.00	2	289.18	1.72
	b2-16 [8]	16	2	1	309.41 [8]	0.00	2	295.14	4.61
	a2-20 [8]	20	2	1	344.83 [8]	0.00	2	335.68	2.65
	b2-20 [8]	20	2	1	332.64 [8]	0.00	2	331.13	0.45
Medium-scale	a2-24 [8]	24	2	1	431.12 [8]	0.00	2	418.29	2.97
	b2-24 [8]	24	2	1	444.71 [8]	0.00	2	435.33	2.10
	R1a [7]	24	3	1	190.02 [9]	0.00	3	178.36	6.13
	R1b [7]	24	3	1	164.46 [9]	0.00	3	158.62	3.55
Large-scale	R2a [7]	48	5	1	301.33 [9]	0.00	5	285.76	5.16
	R2b [7]	48	5	1	295.66 [9]	0.00	5	278.81	5.69
	R3a [7]	72	7	1	532.42 [9]	0.00	7	525.39	1.24
	R3b [7]	72	7	1	484.82 [10]	0.00	7	456.35	5.87

Note: '' highlighted numbers indicate the best travel cost by which all passengers on the instance can be served.*

TABLE D.4: Comparison of results attained by the proposed DA vs BKS [9][8][10] on DARP instances [7][8].

FIGURE D.1: Flowchart: Bi-level optimization framework for solving DARP with depot optimization.

FIGURE D.2: Small-scale instance: BKS for b2-16 instance.

FIGURE D.3: Small-scale instance: DA for b2-16 instance.

FIGURE D.4: Medium-scale instance: BKS for pr01 instance.

FIGURE D.5: Medium-scale instance: DA for pr01 instance.

FIGURE D.6: Large-scale instance: BKS for pr12 instance.

FIGURE D.7: Large-scale instance: DA for pr12 instance.

List of Author's Publications¹

Journal Articles

- **Ramesh Ramasamy Pandi**, Song Guang Ho, Sarat Chandra Nagavarapu, and Justin Dauwels, “A Generic GPU-Accelerated Framework for the Dial-A-Ride Problem”, *In IEEE Transactions on Intelligent Transportation Systems*, DOI: 10.1109/TITS.2020.2992560.
- **Ramesh Ramasamy Pandi**, Song Guang Ho, Sarat Chandra Nagavarapu, Twinkle Tripathy, and Justin Dauwels, “Disruption Management for Dial-A-Ride Systems”, *In IEEE Intelligent Transportation Systems Magazine*, DOI: 10.1109/MITS.2020.3014429.
- **Ramesh Ramasamy Pandi** and Justin Dauwels, “The Electric Autonomous Dial-A-Ride Problem with Mobile Energy Disseminators”, under review, *IEEE Transactions on Intelligent Transportation Systems*.

Book Chapters

- Twinkle Tripathy, Sarat Chandra Nagavarapu, Kaveh Azizian, **Ramesh Ramasamy Pandi** and Justin Dauwels, “*Solving Dial-A-Ride Problems Using Multiple Ant Colony System with Fleet Size Minimisation*”, *Advances in Computational Intelligence Systems (ACIS)*, Springer, vol. 650, pp. 325-336, 2018.

¹The superscript * indicates joint first authors

Conference Proceedings

- **Ramesh Ramasamy Pandi**, Song Guang Ho, Sarat Chandra Nagavarapu, and Justin Dauwels, “*Adaptive Algorithm for Dial-A-Ride Problem with Vehicle Breakdown*”, Proceedings of the 18th European Control Conference (ECC), Naples, Italy, pp. 2682-2688, 2019.
- **Ramesh Ramasamy Pandi**, Song Guang Ho, Sarat Chandra Nagavarapu, Twinkle Tripathy, and Justin Dauwels, “*GPU-Accelerated Tabu Search Algorithm for the Dial-A-Ride Problem*”, Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems (ITSC), Hawaii, USA, 2018.
- Song Guang Ho, Hongwei Koh, **Ramesh Ramasamy Pandi**, Sarat Chandra Nagavarapu and Justin Dauwels, “*Solving Time-dependent Dial-a-ride Problem using Greedy Ant Colony Optimization*”, Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems (ITSC), Hawaii, USA, 2018.
- **Ramesh Ramasamy Pandi**, Song Guang Ho, Sarat Chandra Nagavarapu, and Justin Dauwels, “*Deterministic Annealing for Depot Optimization: Applications to the Dial-A-Ride Problem*”, Proceedings of the 9th IEEE Symposium Series on Computational Intelligence (SSCI), Bangalore, India, 2018.
- Song Guang Ho, Sarat Chandra Nagavarapu, **Ramesh Ramasamy Pandi**, and Justin Dauwels, “*Improved Tabu Search Heuristic for Static Dial-A-Ride Problem: Faster and Better Convergence*”, Proceedings of the 25th World Congress on Intelligent Transport Systems (ITSWC), Copenhagen, 2018.
- Song Guang Ho, **Ramesh Ramasamy Pandi**, Sarat Chandra Nagavarapu, and Justin Dauwels, “*Multi-Atomic Annealing Heuristic for the Dial-a-ride Problem*”, Proceedings of the 12th IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI), Singapore, pp. 272-277, 2018.

Bibliography

- [1] Demand responsive transport. https://en.wikipedia.org/wiki/Demand_responsive_transport. Accessed:01-06-2020. [xxv](#), [17](#)
- [2] Prabuddha Chakraborty, Robert C Parker, Tamzidul Hoque, Jonathan Cruz, and Swarup Bhunia. P2c2: Peer-to-peer car charging. *arXiv preprint arXiv:2002.07268*, 2020. [xxvii](#), [188](#), [190](#)
- [3] Nouredine Melab, Imen Chakroun, Mohand Mezmaz, and Daniel Tuytens. A gpu-accelerated branch-and-bound algorithm for the flow-shop scheduling problem. In *2012 IEEE International Conference on Cluster Computing*, pages 10–17. IEEE, 2012. [xxvii](#), [193](#)
- [4] YD Deng and Shuai Mu. A survey on gpu based electronic design automation computing. *Foundation and Trends in Electronics Design Automation*, 7(1-2):1–176, 2013. [xxvii](#), [194](#)
- [5] Sin C Ho, WY Szeto, Yong-Hong Kuo, Janny MY Leung, Matthew Petering, and Terence WH Tou. A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological*, 2018. [xxx](#), [4](#), [5](#), [48](#), [55](#), [57](#), [58](#), [60](#), [112](#), [114](#), [142](#), [143](#), [152](#), [153](#), [193](#)
- [6] Ramesh Ramasamy Pandi, Song Guang Ho, Sarat Chandra Nagavarapu, Twinkle Tripathy, and Justin Dauwels. GPU-accelerated tabu search algorithm for dial-a-ride problem. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2519–2524. IEEE, 2018. [xxx](#), [30](#), [54](#), [62](#), [83](#), [87](#), [122](#), [143](#), [152](#), [153](#), [162](#)
- [7] Jean-François Cordeau and Gilbert Laporte. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579–594, 2003. [xxxi](#), [17](#), [32](#), [34](#), [41](#), [42](#), [44](#), [46](#), [47](#), [57](#), [58](#), [67](#), [68](#), [82](#), [83](#), [84](#), [90](#), [91](#), [112](#), [115](#), [117](#), [129](#), [152](#), [154](#), [163](#), [164](#), [169](#), [197](#), [202](#), [207](#), [226](#), [227](#), [228](#)
- [8] Stefan Ropke, Jean-François Cordeau, and Gilbert Laporte. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, 49(4):258–272, 2007. [xxxi](#), [20](#), [26](#), [27](#), [39](#), [41](#), [192](#), [203](#), [204](#), [207](#), [225](#), [226](#), [228](#)

- [9] Sophie N Parragh, Karl F Doerner, and Richard F Hartl. Variable neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, 37(6):1129–1138, 2010. [xxxi](#), [28](#), [35](#), [44](#), [57](#), [58](#), [85](#), [91](#), [97](#), [117](#), [152](#), [207](#), [226](#), [227](#), [228](#)
- [10] Sophie N Parragh and Verena Schmid. Hybrid column generation and large neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, 40(1):490–497, 2013. [xxxi](#), [20](#), [28](#), [41](#), [46](#), [57](#), [117](#), [207](#), [227](#), [228](#)
- [11] Kris Braekers, An Caris, and Gerrit K Janssens. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. *Transportation Research Part B: Methodological*, 67:166–186, 2014. [xxxi](#), [29](#), [39](#), [45](#), [47](#), [57](#), [58](#), [112](#), [117](#), [192](#), [207](#)
- [12] Maxime Chassaing, Christophe Duhamel, and Philippe Lacomme. An els-based approach with dynamic probabilities management in local search for the dial-a-ride problem. *Engineering Applications of Artificial Intelligence*, 48:119–133, 2016. [xxxi](#), [30](#), [45](#), [46](#), [57](#), [58](#), [152](#), [207](#)
- [13] Marco Pavone. Autonomous mobility-on-demand systems for future urban mobility. In *Autonomes Fahren*, pages 399–416. Springer, 2015. [3](#), [54](#), [148](#)
- [14] Christopher E Borroni-Bird, Lawrence D Burns, and William J Mitchell. *Reinventing the Automobile: Personal urban mobility for the 21st century*. MIT Press, 2010. [3](#), [54](#), [148](#)
- [15] PhD Alessandro Lori. Are self-driving cars safe? *Verizon Connect*, 2020. [3](#)
- [16] Wang Zheng, Zhou Chun-yue, and Hu Xiang-pei. A simulation-based method for evaluating the reliability of vehicle routing schemes in the vaccine delivery with vehicle breakdown. In *Management Science and Engineering (ICMSE), 2013 International Conference on*, pages 258–262. IEEE, 2013. [5](#), [113](#)
- [17] Xuping Wang, Xu Wu, and Xiangpei Hu. A study of urgency vehicle routing disruption management problem. In *Information Engineering (ICIE), 2010 WASE International Conference on*, volume 3, pages 452–455. IEEE, 2010. [5](#), [49](#), [113](#)
- [18] Vehicle breakdowns hit record high in u.s. despite new car tech. <http://fortune.com/2016/07/20/aaa-breakdown-new-tech-study/>. [6](#), [49](#), [113](#)
- [19] Matti van Engelen, Oded Cats, Henk Post, and Karen Aardal. Enhancing flexible transport services with demand-anticipatory insertion heuristics. *Transportation Research Part E: Logistics and Transportation Review*, 110: 110–121, 2018. [6](#), [30](#), [36](#), [49](#), [114](#)
- [20] Gitae Kim, Yew Soon Ong, Taesu Cheong, and Puay Siew Tan. Solving the dynamic vehicle routing problem under traffic congestion. *IEEE Transactions on Intelligent Transportation Systems*, 17(8):2367–2380, 2016. [6](#), [49](#), [114](#)

- [21] Yonghua Song, Xia Yang, and Zongxiang Lu. Integration of plug-in hybrid and electric vehicles: Experience from china. In *IEEE PES General Meeting*, pages 1–6. IEEE, 2010. [6](#), [148](#)
- [22] Nic Lutsey and Michael Nicholas. Update on electric vehicle costs in the united states through 2030. *Int. Counc. Clean Transp*, pages 1–12, 2019. [6](#), [148](#)
- [23] Gerhard Hiermann, Richard F Hartl, Jakob Puchinger, and Thibaut Vidal. Routing a mix of conventional, plug-in hybrid, and electric vehicles. *European Journal of Operational Research*, 272(1):235–248, 2019. [6](#), [148](#), [150](#), [151](#)
- [24] Claudia Bongiovanni, Mor Kaspi, and Nikolas Geroliminis. The electric autonomous dial-a-ride problem. *Transportation Research Part B: Methodological*, 122:436–456, 2019. [7](#), [31](#), [33](#), [47](#), [50](#), [57](#), [117](#), [150](#), [151](#), [153](#), [154](#), [155](#), [156](#), [158](#), [163](#), [167](#), [169](#), [170](#), [173](#), [174](#), [192](#), [217](#)
- [25] Dimitrios Kosmanos, Leandros A Maglaras, Michalis Mavrovouniotis, Sotiris Moschoyiannis, Antonios Argyriou, Athanasios Maglaras, and Helge Janicke. Route optimization of electric vehicles based on dynamic wireless charging. *IEEE Access*, 6:42551–42565, 2018. [7](#), [151](#), [152](#), [153](#), [169](#), [192](#)
- [26] Nigel HM Wilson, JM Sussman, Ho-Kwan Wong, and T Higonnet. *Scheduling algorithms for a dial-a-ride system*. Massachusetts Institute of Technology. Urban Systems Laboratory, 1971. [17](#), [56](#), [152](#)
- [27] Harilaos N Psaraftis. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14(2):130–154, 1980. [17](#), [41](#)
- [28] Jean-François Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54(3):573–586, 2006. [20](#), [26](#), [39](#), [41](#), [57](#), [59](#), [90](#), [112](#), [116](#), [117](#), [152](#), [169](#)
- [29] Rene Munch Jorgensen, Jesper Larsen, and Kristin Berg Bergvinsdottir. Solving the dial-a-ride problem using genetic algorithms. *Journal of the operational research society*, 58(10):1321–1331, 2007. [27](#), [33](#), [35](#), [45](#), [57](#), [58](#), [63](#), [64](#), [97](#)
- [30] Ying Luo and Paul Schonfeld. A rejected-reinsertion heuristic for the static dial-a-ride problem. *Transportation Research Part B: Methodological*, 41(7):736–755, 2007. [27](#), [33](#)
- [31] Emanuel Melachrinoudis, Ahmet B Ilhan, and Hokey Min. A dial-a-ride problem for client transportation in a health-care organization. *Computers & Operations Research*, 34(3):742–759, 2007. [27](#), [33](#), [35](#), [44](#)
- [32] Roberto Wolfler Calvo and Alberto Colorni. An effective and fast heuristic for the dial-a-ride problem. *JOR*, 5(1):61–73, 2007. [27](#), [57](#), [152](#)

- [33] Luca Quadrifoglio, Maged M Dessouky, and Fernando Ordóñez. A simulation study of demand responsive transit system design. *Transportation Research Part A: Policy and Practice*, 42(4):718–737, 2008. [27](#)
- [34] Holly A Waisanen, Devavrat Shah, and Munther A Dahleh. A dynamic pickup and delivery problem in mobile networks under information constraints. *IEEE Transactions on Automatic Control*, 53(6):1419–1433, 2008. [27](#)
- [35] Zhihai Xiang, Chengbin Chu, and Haoxun Chen. The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments. *European Journal of Operational Research*, 185(2):534–551, 2008. [27](#), [35](#), [36](#)
- [36] Claudio Cubillos, Enrique Urrea, and Nibaldo Rodríguez. Application of genetic algorithms for the darptw problem. *International Journal of Computers Communications & Control*, 4(2):127–136, 2009. [27](#), [45](#)
- [37] Carl H Häll, Henrik Andersson, Jan T Lundgren, and Peter Värbrand. The integrated dial-a-ride problem. *Public Transport*, 1(1):39–54, 2009. [27](#), [32](#)
- [38] Suleyman Karabuk. A nested decomposition approach for solving the para-transit vehicle scheduling problem. *Transportation Research Part B: Methodological*, 43(4):448–465, 2009. [27](#), [32](#), [42](#)
- [39] Geraldo Regis Mauri, Luiz Antonio, and Nogueira Lorena. Customers’ satisfaction in a dial-a-ride problem. *IEEE Intelligent Transportation Systems Magazine*, 1(3):6–14, 2009. [27](#)
- [40] Sophie N Parragh, Karl F Doerner, Richard F Hartl, and Xavier Gandibleux. A heuristic two-phase solution approach for the multi-objective dial-a-ride problem. *Networks*, 54(4):227–242, 2009. [27](#), [35](#), [46](#)
- [41] Thomas Hanne, Teresa Melo, and Stefan Nickel. Bringing robustness to patient flow management through optimized patient transports in hospitals. *Interfaces*, 39(3):241–255, 2009. [26](#), [27](#), [32](#), [36](#), [45](#)
- [42] Esa Hyttiä, Aleksi Penttinen, and Reijo Sulonen. Congestive collapse and its avoidance in a dynamic dial-a-ride system with time windows. In *International Conference on Analytical and Stochastic Modeling Techniques and Applications*, pages 397–408. Springer, 2010. [27](#), [36](#)
- [43] Alexandre Beaudry, Gilbert Laporte, Teresa Melo, and Stefan Nickel. Dynamic transportation of patients in hospitals. *OR spectrum*, 32(1):77–107, 2010. [26](#), [27](#), [32](#), [36](#), [44](#)
- [44] Cristián E Cortés, Martín Matamala, and Claudio Contardo. The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *European Journal of Operational Research*, 200(3):711–724, 2010. [27](#), [32](#)

- [45] Thierry Garaix, Christian Artigues, Dominique Feillet, and Didier Josselin. Vehicle routing problems with alternative paths: An application to on-demand transportation. *European Journal of Operational Research*, 204(1): 62–75, 2010. [28](#)
- [46] Anupam Gupta, MohammadTaghi Hajiaghayi, Viswanath Nagarajan, and Ramamoorthi Ravi. Dial a ride from k-forest. *ACM Transactions on Algorithms (TALG)*, 6(2):1–21, 2010. [28](#)
- [47] Dag Haugland and Sin C Ho. Feasibility testing for dial-a-ride problems. In *International Conference on Algorithmic Applications in Management*, pages 170–179. Springer, 2010. [28](#)
- [48] Jiafu Tang, Yuan Kong, Henry Lau, and Andrew WH Ip. A note on “efficient feasibility testing for dial-a-ride problems”. *Operations Research Letters*, 38(5):405–407, 2010. [28](#)
- [49] Gerardo Berbeglia, Gilles Pesant, and Louis-Martin Rousseau. Checking the feasibility of dial-a-ride instances using constraint programming. *Transportation Science*, 45(3):399–412, 2011. [28](#), [41](#), [46](#)
- [50] Murat Firat and Gerhard J Woeginger. Analysis of the dial-a-ride problem of hunsaker and savelsbergh. *Operations Research Letters*, 39(1):32–35, 2011. [28](#)
- [51] Thierry Garaix, Christian Artigues, Dominique Feillet, and Didier Josselin. Optimization of occupancy rate in dial-a-ride problems via linear fractional column generation. *Computers & Operations Research*, 38(10):1435–1442, 2011. [28](#), [34](#), [41](#)
- [52] Lauri Häme. An adaptive insertion algorithm for the single-vehicle dial-a-ride problem with narrow time windows. *European Journal of Operational Research*, 209(1):11–22, 2011. [28](#), [36](#)
- [53] Sophie N Parragh. Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. *Transportation Research Part C: Emerging Technologies*, 19(5):912–930, 2011. [26](#), [28](#), [39](#), [47](#)
- [54] Michael Schilde, Karl F Doerner, and Richard F Hartl. Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. *Computers & operations research*, 38(12):1719–1730, 2011. [28](#), [35](#), [36](#), [44](#)
- [55] Géraldine Heilporn, Jean-François Cordeau, and Gilbert Laporte. An integer l-shaped algorithm for the dial-a-ride problem with stochastic customer delays. *Discrete Applied Mathematics*, 159(9):883–895, 2011. [28](#), [37](#), [41](#)
- [56] Sin C Ho and Dag Haugland. Local search heuristics for the probabilistic dial-a-ride problem. *Or Spectrum*, 33(4):961–988, 2011. [28](#), [36](#), [44](#)

- [57] Esa Hyttiä, Aleksi Penttinen, and Reijo Sulonen. Non-myopic vehicle and route selection in dynamic darp with travel time and workload objectives. *Computers & Operations Research*, 39(12):3021–3030, 2012. [28](#)
- [58] Gerardo Berbeglia, Jean-François Cordeau, and Gilbert Laporte. A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem. *INFORMS Journal on Computing*, 24(3):343–355, 2012. [28](#), [34](#), [36](#), [46](#)
- [59] Gerardo Berbeglia, Gilles Pesant, and Louis-Martin Rousseau. Feasibility of the pickup and delivery problem with fixed partial routes: a complexity analysis. *Transportation Science*, 46(3):359–373, 2012. [28](#)
- [60] Rémy Chevrier, Arnaud Liefoghe, Laetitia Jourdan, and Clarisse Dhaenens. Solving a dial-a-ride problem with a hybrid evolutionary multi-objective approach: Application to demand responsive transport. *Applied Soft Computing*, 12(4):1247–1258, 2012. [28](#), [45](#)
- [61] Craig D’Souza, SN Omkar, and J Senthilnath. Pickup and delivery problem using metaheuristics techniques. *Expert Systems with Applications*, 39(1): 328–334, 2012. [28](#)
- [62] Sophie N Parragh, Jean-François Cordeau, Karl F Doerner, and Richard F Hartl. Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints. *OR spectrum*, 34(3):593–633, 2012. [26](#), [28](#), [32](#), [41](#)
- [63] Issam Zidi, Khaled Mesghouni, Kamel Zidi, and Khaled Ghedira. A multi-objective simulated annealing for the multi-criteria dial a ride problem. *Engineering Applications of Artificial Intelligence*, 25(6):1121–1131, 2012. [28](#)
- [64] Timothy A Carnes, Shane G Henderson, David B Shmoys, Mahvareh Ahghari, and Russell D MacDonald. Mathematical programming guides air-ambulance routing at ornge. *Interfaces*, 43(3):232–239, 2013. [28](#)
- [65] Francesca Guerriero, Maria Elena Bruni, and Francesca Greco. A hybrid greedy randomized adaptive search heuristic to solve the dial-a-ride problem. *Asia-Pacific Journal of Operational Research*, 30(01):1250046, 2013. [28](#), [35](#), [44](#)
- [66] Lauri Häme and Harri Hakula. Routing by ranking: A link analysis method for the constrained dial-a-ride problem. *Operations Research Letters*, 41(6): 664–669, 2013. [28](#), [41](#)
- [67] Dominik Kirchler and Roberto Wolfler Calvo. A granular tabu search algorithm for the dial-a-ride problem. *Transportation Research Part B: Methodological*, 56:120–135, 2013. [28](#), [33](#), [35](#), [44](#), [47](#), [57](#), [58](#), [91](#), [97](#)
- [68] Santiago Muelas, Antonio LaTorre, and José-María Peña. A variable neighborhood search algorithm for the optimization of a dial-a-ride problem in a large city. *Expert Systems with Applications*, 40(14):5516–5531, 2013. [28](#), [44](#)

- [69] Julie Paquette, Jean-François Cordeau, Gilbert Laporte, and Marta MB Pascoal. Combining multicriteria analysis and tabu search for dial-a-ride problems. *Transportation Research Part B: Methodological*, 52:1–16, 2013. [28](#), [35](#), [44](#)
- [70] Yuan Qu and Jonathan F Bard. The heterogeneous pickup and delivery problem with configurable vehicle capacity. *Transportation Research Part C: Emerging Technologies*, 32:1–20, 2013. [26](#), [29](#), [45](#)
- [71] Line Blander Reinhardt, Tommy Clausen, and David Pisinger. Synchronized dial-a-ride transportation of disabled passengers at airports. *European Journal of Operational Research*, 225(1):106–117, 2013. [29](#), [42](#)
- [72] Chung-Wei Shen and Luca Quadrifoglio. Evaluating centralized versus decentralized zoning strategies for metropolitan ada paratransit services. *Journal of transportation engineering*, 139(5):524–532, 2013. [29](#)
- [73] Carl H Häll and Anders Peterson. Improving paratransit scheduling using ruin and recreate methods. *Transportation planning and technology*, 36(4):377–393, 2013. [29](#)
- [74] Ahmed Atahran, Christophe Lenté, and Vincent T’kindt. A multicriteria dial-a-ride problem with an ecological measure and heterogeneous vehicles. *Journal of Multi-Criteria Decision Analysis*, 21(5-6):279–298, 2014. [29](#), [33](#), [45](#)
- [75] Lei Feng, Elise Miller-Hooks, Paul Schonfeld, and Matthew Mohebbi. Optimizing ridesharing services for airport access. *Transportation Research Record*, 2467(1):157–167, 2014. [29](#)
- [76] Hagai Ilani, Elad Shufan, Tal Grinshpoun, Aviad Belulu, and Alex Fainberg. A reduction approach to the two-campus transport problem. *Journal of Scheduling*, 17(6):587–599, 2014. [29](#)
- [77] Renaud Masson, Fabien Lehuédé, and Olivier Péton. The dial-a-ride problem with transfers. *Computers & Operations Research*, 41:12–23, 2014. [29](#), [32](#), [45](#), [57](#), [58](#), [85](#), [89](#), [117](#), [125](#), [152](#)
- [78] Fabien Lehuédé, Renaud Masson, Sophie N Parragh, Olivier Péton, and Fabien Tricoire. A multi-criteria large neighbourhood search for the transportation of disabled people. *Journal of the Operational Research Society*, 65(7):983–1000, 2014. [29](#), [34](#), [35](#), [45](#)
- [79] Ka-Io Wong, AF Han, and CW Yuen. On dynamic demand responsive transport services with degree of dynamism. *Transportmetrica A: Transport Science*, 10(1):55–73, 2014. [29](#)
- [80] Maher Maalouf, Cameron A MacKenzie, Sridhar Radakrishnan, and Mary Court. A new fuzzy logic approach to capacitated dynamic dial-a-ride problem. *Fuzzy Sets and Systems*, 255:30–40, 2014. [29](#)

- [81] Alfredo Núñez, Cristián E Cortés, Doris Sáez, Bart De Schutter, and Michel Gendreau. Multiobjective model predictive control for dynamic pickup and delivery problems. *Control Engineering Practice*, 32:73–86, 2014. [29](#)
- [82] Michael Schilde, Karl F Doerner, and Richard F Hartl. Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. *European journal of operational research*, 238(1):18–30, 2014. [29](#), [33](#), [35](#), [44](#)
- [83] Carl H Häll, Jan T Lundgren, and Stefan Voss. Evaluating the performance of a dial-a-ride service using simulation. *Public Transport*, 7(2):139–157, 2015. [29](#)
- [84] Lauri Häme and Harri Hakula. A maximum cluster algorithm for checking the feasibility of dial-a-ride instances. *Transportation Science*, 49(2):295–310, 2015. [29](#), [41](#)
- [85] Nikola Marković, Rahul Nair, Paul Schonfeld, Elise Miller-Hooks, and Matthew Mohebbi. Optimizing dial-a-ride services in maryland: benefits of computerized routing and scheduling. *Transportation Research Part C: Emerging Technologies*, 55:156–165, 2015. [29](#)
- [86] Douglas O Santos and Eduardo C Xavier. Taxi and ride sharing: A dynamic dial-a-ride problem with money as an incentive. *Expert Systems with Applications*, 42(19):6728–6737, 2015. [29](#)
- [87] Diego Muñoz-Carpintero, Doris Sáez, Cristián E Cortés, and Alfredo Núñez. A methodology based on evolutionary algorithms to solve a dynamic pickup and delivery problem under a hybrid predictive control approach. *Transportation Science*, 49(2):239–253, 2015. [29](#)
- [88] Hamid R Sayarshad and Joseph YJ Chow. A scalable non-myopic dynamic dial-a-ride and pricing problem. *Transportation Research Part B: Methodological*, 81:539–554, 2015. [29](#)
- [89] Timo Gschwind and Stefan Irnich. Effective handling of dynamic time windows and its application to solving the dial-a-ride problem. *Transportation Science*, 49(2):335–354, 2015. [29](#), [41](#)
- [90] Ta-Yin Hu and Chin-Ping Chang. A revised branch-and-price algorithm for dial-a-ride problems with the consideration of time-dependent travel cost. *Journal of Advanced Transportation*, 49(6):700–723, 2015. [29](#)
- [91] Mengyang Liu, Zhixing Luo, and Andrew Lim. A branch-and-cut algorithm for a realistic dial-a-ride problem. *Transportation Research Part B: Methodological*, 81:267–288, 2015. [26](#), [29](#), [32](#), [41](#)
- [92] Santiago Muelas, Antonio LaTorre, and Jose-Maria Pena. A distributed vns algorithm for optimizing dial-a-ride problems in large-scale scenarios. *Transportation Research Part C: Emerging Technologies*, 54:110–130, 2015. [29](#)

- [93] Sophie N Parragh, Jorge Pinho de Sousa, and Bernardo Almada-Lobo. The dial-a-ride problem with split requests and profits. *Transportation Science*, 49(2):311–334, 2015. [29](#), [32](#), [33](#), [34](#), [41](#), [44](#)
- [94] Yuan Qu and Jonathan F Bard. A branch-and-price-and-cut algorithm for heterogeneous pickup and delivery problems with configurable vehicle capacity. *Transportation Science*, 49(2):254–270, 2015. [29](#), [32](#), [41](#)
- [95] Enrique Urrea, Claudio Cubillos, and Daniel Cabrera-Paniagua. A hyper-heuristic for the dial-a-ride problem with time windows. *Mathematical Problems in Engineering*, 2015, 2015. [29](#)
- [96] Zhenzhen Zhang, Mengyang Liu, and Andrew Lim. A memetic algorithm for the patient transportation problem. *Omega*, 54:60–71, 2015. [26](#), [29](#), [32](#), [45](#)
- [97] Kris Braekers and Attila A Kovacs. A multi-period dial-a-ride problem with driver consistency. *Transportation Research Part B: Methodological*, 94:355–377, 2016. [30](#), [32](#), [45](#)
- [98] Timo Gschwind and Michael Drexl. Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. *Technical Report LM-2016-08. Johannes Gutenberg University, Mainz*. <http://logistik.bwl.uni-mainz.de/Dateien/LM-2016-08.pdf>, 53(2):480–491, 2019. [30](#), [46](#)
- [99] Mohamed Amine Masmoudi, Manar Hosny, Kris Braekers, and Abdelaziz Dammak. Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. *Transportation Research Part E: Logistics and Transportation Review*, 96:60–80, 2016. [30](#), [32](#), [45](#), [47](#)
- [100] Ulrike Ritzinger, Jakob Puchinger, and Richard F Hartl. Dynamic programming based metaheuristics for the dial-a-ride problem. *Annals of Operations Research*, 236(2):341–358, 2016. [30](#), [34](#), [45](#), [46](#)
- [101] Paolo Detti, Francesco Papalini, and Garazi Zabalo Manrique de Lara. A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare. *Omega*, 70:1–14, 2017. [26](#), [30](#), [44](#)
- [102] Andrew Lim, Zhenzhen Zhang, and Hu Qin. Pickup and delivery service with manpower planning in hong kong public hospitals. *Transportation Science*, 51(2):688–705, 2017. [30](#)
- [103] Mohamed Amine Masmoudi, Kris Braekers, Malek Masmoudi, and Abdelaziz Dammak. A hybrid genetic algorithm for the heterogeneous dial-a-ride problem. *Computers & Operations Research*, 81:1–13, 2017. [30](#), [45](#), [57](#), [112](#), [117](#), [139](#), [152](#)
- [104] Yves Molenbruch, Kris Braekers, and An Caris. Benefits of horizontal cooperation in dial-a-ride services. *Transportation Research Part E: Logistics and Transportation Review*, 107:97–119, 2017. [30](#)

- [105] Yves Molenbruch, Kris Braekers, and An Caris. Operational effects of service level variations for the dial-a-ride problem. *Central European Journal of Operations Research*, 25(1):71–90, 2017. [30](#)
- [106] Yves Molenbruch, Kris Braekers, An Caris, and Greet Vanden Berghe. Multi-directional local search for a bi-objective dial-a-ride problem in patient transportation. *Computers & Operations Research*, 77:58–71, 2017. [26](#), [30](#), [35](#), [44](#), [46](#)
- [107] Victor Pimenta, Alain Quilliot, Hélène Toussaint, and Daniele Vigo. Models and algorithms for reliability-oriented dial-a-ride with autonomous electric vehicles. *European Journal of Operational Research*, 257(2):601–613, 2017. [30](#), [33](#)
- [108] Marcus Posada, Henrik Andersson, and Carl H Häll. The integrated dial-a-ride problem with timetabled fixed route service. *Public Transport*, 9(1-2): 217–241, 2017. [30](#)
- [109] Jörn Schönberger. Scheduling constraints in dial-a-ride problems with transfers: a metaheuristic approach incorporating a cross-route scheduling procedure with postponement opportunities. *Public Transport*, 9(1-2):243–272, 2017. [30](#)
- [110] Twinkle Tripathy, Sarat Chandra Nagavarapu, Kaveh Azizian, Ramesh Ramasamy Pandi, and Justin Dauwels. Solving dial-a-ride problems using multiple ant colony system with fleet size minimisation. In *UK Workshop on Computational Intelligence*, pages 325–336. Springer, 2017. [30](#)
- [111] Mohamed Amine Masmoudi, Manar Hosny, Emrah Demir, Konstantinos N Genikomsakis, and Naoufel Cheikhrouhou. The dial-a-ride problem with electric vehicles and battery swapping stations. *Transportation research part E: logistics and transportation review*, 118:392–420, 2018. [30](#), [33](#), [150](#), [151](#), [153](#), [187](#)
- [112] Oscar Tellez, Samuel Vercraene, Fabien Lehuédé, Olivier Péton, and Thibaud Monteiro. The fleet size and mix dial-a-ride problem with reconfigurable vehicle capacity. *Transportation Research Part C: Emerging Technologies*, 91:99–123, 2018. [30](#), [32](#)
- [113] Martin Riedler and Günther Raidl. Solving a selective dial-a-ride problem with logic-based benders decomposition. *Computers & Operations Research*, 96:30–54, 2018. [30](#), [32](#)
- [114] Richardos Drakoulis, Francesco Bellotti, Ioannis Bakas, Riccardo Berta, Pratheep Kumar Paranthaman, Gautam Ravindra Dange, Panagiotis Lytrivis, Katia Pagle, Alessandro De Gloria, and Angelos Amditis. A gamified flexible transportation service for on-demand public transport. *IEEE Transactions on Intelligent Transportation Systems*, 19(3):921–933, 2018. [30](#), [33](#)

- [115] Ramesh Ramasamy Pandi, Song Guang Ho, Sarat Chandra Nagavarapu, and Justin Dauwels. Deterministic annealing for depot optimization: Applications to the dial-a-ride problem. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 88–95. IEEE, 2018. [30](#), [148](#), [153](#), [187](#), [224](#)
- [116] Songguang Ho, Sarat Chandra Nagavarapu, Ramesh Ramasamy Pandi, and Justin Dauwels. Improved tabu search heuristics for static dial-a-ride problem: Faster and better convergence. *arXiv preprint arXiv: 1801.09547*, 2018. [30](#)
- [117] Song Guang Ho, Ramesh Ramasamy Pandi, Sarat Chandra Nagavarapu, and Justin Dauwels. Multi-atomic annealing heuristic for the dial-a-ride problem. In *2018 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, pages 268–273. IEEE, 2018. [30](#)
- [118] Song Guang Ho, Hong Wei Koh, Ramesh Ramasamy Pandi, Sarat Chandra Nagavarapu, and Justin Dauwels. Solving time-dependent dial-a-ride problem using greedy ant colony optimization. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 778–785. IEEE, 2018. [31](#), [189](#), [190](#)
- [119] Timo Gschwind and Michael Drexl. Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. *Transportation Science*, 53(2):480–491, 2019. [31](#), [45](#), [47](#)
- [120] Mehdi Abedi, Raymond Chiong, Rukshan Athauda, Hany Seidgar, Zbigniew Michalewicz, and Andrew Sturt. A regional multi-objective tabu search algorithm for a green heterogeneous dial-a-ride problem. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 2082–2089. IEEE, 2019. [31](#), [34](#)
- [121] Zhixing Luo, Mengyang Liu, and Andrew Lim. A two-phase branch-and-price-and-cut for a dial-a-ride problem in patient transportation. *Transportation Science*, 53(1):113–130, 2019. [31](#), [41](#)
- [122] Connor Riley, Antoine Legrain, and Pascal Van Hentenryck. Column generation for real-time ride-sharing operations. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 472–487. Springer, 2019. [31](#), [36](#)
- [123] Arslan Ali Syed, Bernd Kaltenhaeuser, Irina Gaponova, and Klaus Bogenberger. Asynchronous adaptive large neighborhood search algorithm for dynamic matching problem in ride hailing services. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3006–3012. IEEE, 2019. [31](#), [36](#)
- [124] Ramesh Ramasamy Pandi, Song Guang Ho, Sarat Chandra Nagavarapu, and Justin Dauwels. Adaptive algorithm for dial-a-ride problem with vehicle breakdown. In *2019 18th European Control Conference (ECC)*, pages 2682–2688. IEEE, 2019. [31](#), [112](#)

- [125] Jan Christiaens, Hatice Çalik, Tony Wauters, Reshma Chirayil Chandrasekharan, and Greet Vanden Berghe. The prisoner transportation problem. *European Journal of Operational Research*, 284(3):1058–1073, 2020. [31](#), [32](#)
- [126] Mohamed Amine Masmoudi, Manar Hosny, Emrah Demir, and Erwin Pesch. Hybrid adaptive large neighborhood search algorithm for the mixed fleet heterogeneous dial-a-ride problem. *Journal of Heuristics*, 26(1):83–118, 2020. [31](#), [32](#)
- [127] Célia Paquay, Yves Crama, and Thierry Pironet. Recovery management for a dial-a-ride system with real-time disruptions. *European Journal of Operational Research*, 280(3):953–969, 2020. [31](#), [36](#)
- [128] Oscar Tellez, Samuel Vercraene, Fabien Lehuédé, Olivier Péton, and Thibaud Monteiro. The time-consistent dial-a-ride problem. 2020. [31](#), [33](#)
- [129] Xiao Liang, Gonçalo Homem de Almeida Correia, Kun An, and Bart van Arem. Automated taxis’ dial-a-ride problem with ride-sharing considering congestion-based dynamic travel times. *Transportation Research Part C: Emerging Technologies*, 112:260–281, 2020. [31](#), [35](#)
- [130] Ramesh Ramasamy Pandi, Song Guang Ho, Sarat Chandra Nagavarapu, and Justin Dauwels. A generic GPU-accelerated framework for the dial-a-ride problem. *IEEE Transactions on Intelligent Transportation Systems*, In Press, Doi: 10.1109/ TITS.2020.2992560., 2020. [31](#), [46](#), [54](#), [193](#)
- [131] Ramesh Ramasamy Pandi, Song Guang Ho, Sarat Chandra Nagavarapu, Twinkle Tripathy, and Justin Dauwels. Disruption management for dial-a-ride systems. *IEEE Intelligent Transportation Systems Magazine*, In Press, Doi: 10.1109/ MITS.2020.3014429., 2020. [31](#), [36](#), [112](#), [188](#), [190](#)
- [132] Ramesh Ramasamy Pandi and Justin Dauwels. The electric autonomous dial-a-ride problem with mobile energy disseminators. *IEEE Transactions on Intelligent Transportation Systems*, submitted, 2020. [31](#), [34](#), [148](#), [187](#)
- [133] Ka-Io Wong and Michael GH Bell. Solution of the dial-a-ride problem with multi-dimensional capacity constraints. *International transactions in operational research*, 13(3):195–208, 2006. [26](#), [35](#), [42](#)
- [134] Zhihai Xiang, Chengbin Chu, and Haoxun Chen. A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints. *European journal of operational research*, 174(2):1117–1139, 2006. [26](#), [35](#)
- [135] Mark D Hickman and Kelly L Blume. An investigation of integrated transit service. Technical report, Southwest University Transportation Center, Texas Transportation Institute . . . , 2001. [32](#)

- [136] Majid Aldaihani and Maged M Dessouky. Hybrid scheduling methods for paratransit operations. *Computers & Industrial Engineering*, 45(1):75–96, 2003. [32](#)
- [137] Jang-Jei Jaw, Amedeo R Odoni, Harilaos N Psaraftis, and Nigel HM Wilson. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B: Methodological*, 20(3):243–257, 1986. [33](#), [42](#), [57](#), [152](#)
- [138] Marco Diana and Maged M Dessouky. A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. *Transportation Research Part B: Methodological*, 38(6):539–557, 2004. [33](#), [35](#), [42](#), [57](#)
- [139] Julie Paquette, Jean-François Cordeau, and Gilbert Laporte. Quality of service in dial-a-ride operations. *Computers & Industrial Engineering*, 56(4):1721–1734, 2009. [33](#), [57](#)
- [140] Roberto Wolfler Calvo and Nora Touati-Moungla. A matheuristic for the dial-a-ride problem. In *International Conference on Network Optimization*, pages 450–463. Springer, 2011. [33](#), [44](#)
- [141] GR Mauri and LA Nogueira Lorena. A multiobjective model and simulated annealing approach for a dial-a-ride problem. In *Workshop dos Cursos de Computação*, 2006. [33](#), [35](#), [45](#)
- [142] Marco Diana, Maged M Dessouky, and Nan Xia. A model for the fleet sizing of demand responsive transportation services with time windows. *Transportation Research Part B: Methodological*, 40(8):651–666, 2006. [34](#)
- [143] Andrea Attanasio, Jean-François Cordeau, Gianpaolo Ghiani, and Gilbert Laporte. Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing*, 30(3):377–387, 2004. [34](#), [36](#), [44](#), [46](#)
- [144] Fanglei Yi and Lei Tian. On the online dial-a-ride problem with time-windows. In *International Conference on Algorithmic Applications in Management*, pages 85–94. Springer, 2005. [34](#), [36](#)
- [145] Esteban Feuerstein and Leen Stougie. On-line single-server dial-a-ride problems. *Theoretical Computer Science*, 268(1):91–105, 2001. [34](#), [36](#)
- [146] Sven O Krumke, Willem E de Paepe, Diana Poensgen, Maarten Lipmann, Alberto Marchetti-Spaccamela, and Leen Stougie. On minimizing the maximum flow time in the online dial-a-ride problem. In *International Workshop on Approximation and Online Algorithms*, pages 258–269. Springer, 2005. [34](#), [36](#)
- [147] Liping Fu. Scheduling dial-a-ride paratransit under time-varying, stochastic congestion. *Transportation Research Part B: Methodological*, 36(6):485–506, 2002. [35](#)

- [148] Luca Coslovich, Raffaele Pesenti, and Walter Ukovich. A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. *European Journal of Operational Research*, 175(3):1605–1615, 2006. [36](#), [42](#)
- [149] Makarena Donoso, Daniel Sandoval, and Claudio Cubillos. Sim-madarp: An agent-based tool for dial-a-ride simulation. In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 191–199. Springer, 2009. [36](#)
- [150] Gerardo Berbeglia, Jean-François Cordeau, and Gilbert Laporte. Dynamic pickup and delivery problems. *European journal of operational research*, 202(1):8–15, 2010. [36](#)
- [151] Norbert Ascheuer, Sven O Krumke, and Jörg Rambau. Online dial-a-ride problems: Minimizing the completion time. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 639–650. Springer, 2000. [36](#)
- [152] Maarten Lipmann, Xiwen Lu, Willem E de Paepe, Rene A Sitters, and Leen Stougie. On-line dial-a-ride problems under a restricted information model. *Algorithmica*, 40(4):319–329, 2004. [36](#)
- [153] Francesco Paolo Defflorio, Bruno Dalla Chiara, Alessandro Murro, and Mizar Automazione SpA. Simulation and performance of drts in a realistic environment. In *Proceedings of the 13th Mini-Euro Conference Handling uncertainty in the analysis of Traffic and Transportation systems and the 9th Meeting of the Euro Working Group on Transportation Intermodality, Sustainability and Intelligent transport systems*, pages 622–628, 2002. [37](#)
- [154] Alberto Coppi, Paolo Detti, and Jessica Raffaelli. A planning and routing model for patient transportation in health care. *Electronic notes in discrete mathematics*, 41:125–132, 2013. [41](#)
- [155] Harilaos N Psaraftis. An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation science*, 17(3):351–357, 1983. [41](#), [57](#), [152](#)
- [156] Jacques Desrosiers, Yvan Dumas, and Francois Soumis. A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *American Journal of Mathematical and Management Sciences*, 6(3-4):301–325, 1986. [41](#), [57](#), [152](#)
- [157] Athanasios Ziliaskopoulos and George Kozanidis. Dynamic programming strategies for the dial a ride problem with time window constraints. In *85th Annual Meeting of the Transportation Research*, pages 1–12, 2006. [41](#)
- [158] ATHANASIOS Lois, A Ziliaskopoulos, and GEORGIA Aifantopoulou. A very large scale neighborhood heuristic algorithm for the multivehicle dial a ride with time windows. In *Proceedings of 87th transportation research board annual meeting, Washington DC, USA*, 2007. [41](#)

- [159] Ying Luo and Paul Schonfeld. Online rejected-reinsertion heuristics for dynamic multivehicle dial-a-ride problem. *Transportation research record*, 2218(1):59–67, 2011. [42](#)
- [160] Irina Ioachim, Jacques Desrosiers, Yvan Dumas, Marius M Solomon, and Daniel Villeneuve. A request clustering algorithm for door-to-door handicapped transportation. *Transportation science*, 29(1):63–78, 1995. [42](#), [57](#), [152](#)
- [161] Ralf Borndörfer, Martin Grötschel, Fridolin Klostermeier, and Christian Küttner. Telebus berlin: Vehicle scheduling in a dial-a-ride system. In *Computer-Aided Transit Scheduling*, pages 391–422. Springer, 1999. [42](#)
- [162] Fred Glover. Tabu search—part i. *ORSA Journal on computing*, 1(3):190–206, 1989. [42](#), [83](#)
- [163] Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100, 1997. [44](#)
- [164] Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472, 2006. [44](#), [87](#), [88](#), [122](#), [123](#), [124](#), [125](#), [160](#), [161](#), [162](#)
- [165] John W Baugh Jr, Gopala Krishna Reddy Kakivaya, and John R Stone. Intractability of the dial-a-ride problem and a multiobjective solution using simulated annealing. *Engineering Optimization*, 30(2):91–123, 1998. [45](#)
- [166] Brahim Rekiek, Alain Delchambre, and Hussain Aziz Saleh. Handicapped person transportation: An application of the grouping genetic algorithm. *Engineering Applications of Artificial Intelligence*, 19(5):511–520, 2006. [45](#)
- [167] Claudia Archetti and M Grazia Speranza. A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization*, 2(4):223–246, 2014. [46](#)
- [168] Marco Oberscheider and Patrick Hirsch. Analysis of the impact of different service levels on the workload of an ambulance service provider. *BMC health services research*, 16(1):487, 2016. [46](#)
- [169] Siddhartha Jain and Pascal Van Hentenryck. Large neighborhood search for dial-a-ride problems. In *International Conference on Principles and Practice of Constraint Programming*, pages 400–413. Springer, 2011. [46](#)
- [170] George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959. [54](#)
- [171] Harilaos N Psaraftis, Min Wen, and Christos A Kontovas. Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31, 2016. [55](#), [114](#)

- [172] Erik Lindholm, John Nickolls, Stuart Oberman, and John Montrym. Nvidia tesla: A unified graphics and computing architecture. *IEEE micro*, 28(2): 39–55, 2008. 55
- [173] Thé Van Luong, Nouredine Melab, and El-Ghazali Talbi. GPU computing for parallel local search metaheuristic algorithms. *IEEE Transactions on Computers*, 62(1):173–185, 2011. 56
- [174] Paolo Toth and Daniele Vigo. Heuristic algorithms for the handicapped persons transportation problem. *Transportation science*, 31(1):60–71, 1997. 56, 152
- [175] Oli BG Madsen, Hans F Ravn, and Jens Moberg Rygaard. A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Annals of operations Research*, 60(1):193–208, 1995. 57, 152
- [176] Thomas R Sexton and Young-Myung Choi. Pickup and delivery of partial loads with “soft” time windows. *American Journal of Mathematical and Management Sciences*, 6(3-4):369–398, 1986. 57, 152
- [177] Yvan Dumas, Jacques Desrosiers, and Francois Soumis. The pickup and delivery problem with time windows. *European journal of operational research*, 54(1):7–22, 1991. 57, 152
- [178] Quan Lu and Maged M Dessouky. A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. *European Journal of Operational Research*, 175(2):672–687, 2006. 57
- [179] Uber’s new wait-time fee ‘fair’. <https://www.todayonline.com/singapore/ubers-new-wait-time-fee-fair-commuters-say>, . Accessed:02-11-2019. 57, 60
- [180] Waiting time for taxis still biggest bugbear: study. <https://www.straitstimes.com/singapore/transport/waiting-time-for-taxis-still-biggest-bugbear-study>, . Accessed:02-11-2019. 57, 60
- [181] New grabshare service with slightly cheaper fares in return for 5 minutes’ wait time. <https://www.straitstimes.com/singapore/transport/new-grabshare-service-with-slightly-cheaper-fares-in-return-for-five-minutes>. Accessed:02-11-2019. 57, 60
- [182] Adam Janiak, Wladyslaw A Janiak, and Maciej Lichtenstein. Tabu search on GPU. *J. UCS*, 14(14):2416–2426, 2008. 58, 59, 61
- [183] Thé Van Luong, Lakhdar Loukil, Nouredine Melab, and El-Ghazali Talbi. A GPU-based iterated tabu search for solving the quadratic 3-dimensional assignment problem. In *ACS/IEEE International Conference on Computer Systems and Applications-AICCSA 2010*, pages 1–8. IEEE, 2010. 58, 59, 61

- [184] John Nickolls, Ian Buck, and Michael Garland. Scalable parallel programming. In *2008 IEEE Hot Chips 20 Symposium (HCS)*, pages 40–53. IEEE, 2008. [58](#)
- [185] Michał Czapiński and Stuart Barnes. Tabu search with two approaches to parallel flowshop evaluation on CUDA platform. *Journal of Parallel and Distributed Computing*, 71(6):802–811, 2011. [58](#), [61](#)
- [186] Jagiełło Szymon and Żelazny Dominik. Solving multi-criteria vehicle routing problem by parallel tabu search on GPU. *Procedia Computer Science*, 18: 2529–2532, 2013. [58](#), [59](#), [61](#)
- [187] IM Coelho, PLA Munhoz, LS Ochi, MJF Souza, C Bentes, and R Farias. An integrated CPU–GPU heuristic inspired on variable neighbourhood search for the single vehicle routing problem with deliveries and selective pickups. *International Journal of Production Research*, 54(4):945–962, 2016. [59](#)
- [188] Imen Chakroun, Mohand Mezmaiz, Nouredine Melab, and Ahcene Bendjoudi. Reducing thread divergence in a GPU-accelerated branch-and-bound algorithm. *Concurrency and Computation: Practice and Experience*, 25(8):1121–1136, 2013. [59](#), [79](#)
- [189] Nouredine Melab, Karima Boufaras, El-Ghazali Talbi, et al. ParadisEO-MO-GPU: a framework for parallel GPU-based local search metaheuristics. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 1189–1196. ACM, 2013. [59](#)
- [190] Jean-Francois Cordeau, Michel Gendreau, Gilbert Laporte, Jean-Yves Potvin, and Frédéric Semet. A guide to vehicle routing heuristics. *Journal of the Operational Research society*, 53(5):512–522, 2002. [61](#)
- [191] Fred W Glover and Gary A Kochenberger. *Handbook of metaheuristics*, volume 57. Springer Science & Business Media, 2006. [61](#)
- [192] John Silberholz and Bruce Golden. Comparison of metaheuristics. In *Handbook of metaheuristics*, pages 625–640. Springer, 2010. [61](#)
- [193] Martin WP Savelsbergh. The vehicle routing problem with time windows: Minimizing route duration. *ORSA journal on computing*, 4(2):146–154, 1992. [67](#), [68](#), [163](#), [164](#), [165](#)
- [194] William P Nanry and J Wesley Barnes. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological*, 34(2):107–121, 2000. [82](#), [162](#)
- [195] Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *International conference on principles and practice of constraint programming*, pages 417–431. Springer, 1998. [86](#), [122](#), [160](#)

- [196] Jean-Yves Potvin and Jean-Marc Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331–340, 1993. [88](#), [124](#), [162](#)
- [197] Susan Shaheen, Adam Cohen, Balaji Yelchuru, Sara Sarkhili, Booz Allen Hamilton, et al. Mobility on demand operational concept report. Technical report, United States. Department of Transportation. Intelligent Transportation., 2017. [112](#)
- [198] Susan Shaheen, Adam Cohen, and Ismail Zohdy. Shared mobility: current practices and guiding principles. Technical report, 2016. [112](#)
- [199] Robert Cervero. *Paratransit in America: Redefining mass transportation*. Greenwood Publishing Group, 1997. [112](#)
- [200] Qianxin Mu, Zhuo Fu, Jens Lysgaard, and Richard Eglese. Disruption management of the vehicle routing problem with vehicle breakdown. *Journal of the Operational Research Society*, 62(4):742–749, 2011. [113](#), [114](#), [117](#), [129](#)
- [201] Jens Clausen, Allan Larsen, Jesper Larsen, and Natalia J Rezanova. Disruption management in the airline industry—concepts, models and methods. *Computers & Operations Research*, 37(5):809–821, 2010. [113](#)
- [202] Jerzy A Filar, Prabhu Manyem, and Kevin White. How airlines and airports recover from schedule perturbations: a survey. *Annals of operations research*, 108(1-4):315–333, 2001. [113](#)
- [203] Richard Eglese and Sofoclis Zambirinis. Disruption management in vehicle routing and scheduling for road freight transport: a review. *Top*, 26(1):1–17, 2018. [113](#), [114](#)
- [204] Xuping Wang, Xu Wu, Zheng Wang, and Xianpei Hu. A model and an improved genetic algorithm for the vehicle routing problem with break-down vehicles. In *Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on*, pages 696–699. IEEE, 2009. [113](#)
- [205] Jing-Quan Li, Pitu B Mirchandani, and Denis Borenstein. Real-time vehicle rerouting problems with time windows. *European Journal of Operational Research*, 194(3):711–727, 2009. [114](#), [117](#)
- [206] Car, taxi and bus - how the costs stack up. <https://www.straitstimes.com/business/invest/car-taxi-and-bus-how-the-costs-stack-up>. [130](#)
- [207] Channelnewsasia. <https://vulcanpost.com/641548/shiokr-on-demand-car-wash-singapore/>. [130](#)
- [208] Sgcarmart. https://www.sgcarmart.com/news/carpark_index.php?LOC=all&TYP=carpark&SRH. [130](#)

- [209] Richard E Wener, Gary W Evans, Donald Phillips, and Natasha Nadler. Running for the 7: 45: The effects of public transit improvements on commuter stress. *Transportation*, 30(2):203–220, 2003. 141
- [210] Ihs markit. https://news.ihsmarkit.com/prviewer/release_only/slug/automotive-average-age-cars-and-light-trucks-us-rises-again-2019-118-. Accessed:18-02-2020. 148
- [211] Greenhouse gas emissions. <https://www.epa.gov/ghgemissions/sources-greenhouse-gas-emissions>, . Accessed:18-02-2020. 148
- [212] Yale environment 360. <https://e360.yale.edu/digest/transportation-replaces-power-in-u-s-as-top-source-of-co2-emissions>, . Accessed:18-02-2020. 148
- [213] Michael Schneider, Andreas Stenger, and Dominik Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4):500–520, 2014. 150, 151, 153, 191
- [214] Merve Keskin and Bülent Çatay. Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 65:111–127, 2016. 150, 151, 153
- [215] Gerhard Hiermann, Jakob Puchinger, Stefan Ropke, and Richard F Hartl. The electric fleet size and mix vehicle routing problem with time windows and recharging stations. *European Journal of Operational Research*, 252(3): 995–1018, 2016. 150, 151, 153
- [216] Shaohua Cui, Hui Zhao, Hui Chen, and Cuiping Zhang. The mobile charging vehicle routing problem with time windows and recharging services. *Computational intelligence and neuroscience*, 2018, 2018. 151, 152, 187
- [217] Aqueel Ahmad, Mohammad Saad Alam, and Rakan Chabaan. A comprehensive review of wireless charging technologies for electric vehicles. *IEEE Transactions on Transportation Electrification*, 4(1):38–63, 2017. 152
- [218] Ryan G Conrad and Miguel Andres Figliozzi. The recharging vehicle routing problem. In *Proceedings of the 2011 industrial engineering research conference*, page 8. IISE Norcross, GA, 2011. 152
- [219] Guy Desaulniers, Fausto Errico, Stefan Irnich, and Michael Schneider. Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, 64(6):1388–1405, 2016. 153
- [220] Samuel Pelletier, Ola Jabali, Gilbert Laporte, and Marco Veneroni. Battery degradation and behaviour for electric vehicles: Review and numerical analyses of several models. *Transportation Research Part B: Methodological*, 103: 158–187, 2017. 153, 156

- [221] Samuel Pelletier, Ola Jabali, and Gilbert Laporte. Charge scheduling for electric freight vehicles. *Transportation Research Part B: Methodological*, 115:246–269, 2018. [153](#)
- [222] Saadullah Khan, Aqueel Ahmad, Furkan Ahmad, Mahdi Shafaati Shemami, Mohammad Saad Alam, and Siddiq Khateeb. A comprehensive review on solar powered electric vehicle charging system. *Smart Science*, 6(1):54–79, 2018. [187](#)
- [223] Hend Kamal Gedawy, M Dias, and K Harras. Dynamic path planning and traffic light coordination for emergency vehicle routing. *Comp. Sci. Dept., Carnegie Mellon Univ., Pittsburgh, Pennsylvania, USA*, 2008. [189](#)
- [224] Gitae Kim, Yew Soon Ong, Taesu Cheong, and Puay Siew Tan. Solving the dynamic vehicle routing problem under traffic congestion. *IEEE Transactions on Intelligent Transportation Systems*, 17(8):2367–2380, 2016. [189](#)
- [225] Sarat Chandra Nagavarapu, Twinkle Tripathy, and Justin Dauwels. Development of a simulation platform to implement vehicle routing algorithms for large scale fleet management systems. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 256–261. IEEE, 2017. [190](#)
- [226] Michał Maciejewski and Kai Nagel. Towards multi-agent simulation of the dynamic vehicle routing problem in matsim. In *International Conference on Parallel Processing and Applied Mathematics*, pages 551–560. Springer, 2011. [190](#)
- [227] IEA Global EV Outlook. to electric mobility. *IEA: Paris, France*, 2019. [190](#)
- [228] Sevgi Erdoğan and Elise Miller-Hooks. A green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):100–114, 2012. [191](#)
- [229] Tolga Bektaş and Gilbert Laporte. The pollution-routing problem. *Transportation Research Part B: Methodological*, 45(8):1232–1250, 2011. [191](#)
- [230] Ola Jabali, T Van Woensel, and AG De Kok. Analysis of travel times and co2 emissions in time-dependent vehicle routing. *Production and Operations Management*, 21(6):1060–1074, 2012. [191](#)
- [231] Dominik Goeke and Michael Schneider. Routing a mixed fleet of electric and conventional vehicles. *European Journal of Operational Research*, 245(1):81–99, 2015. [191](#)
- [232] Soonyoung Kang and Jaekyun Moon. Parallel ldpc decoder implementation on gpu based on unbalanced memory coalescing. In *2012 IEEE International Conference on Communications (ICC)*, pages 3692–3697. IEEE, 2012. [195](#)

-
- [233] Naznin Fauzia, Louis-Noël Pouchet, and P Sadayappan. Characterizing and enhancing global memory data coalescing on gpus. In *2015 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, pages 12–22. IEEE, 2015. [195](#)
- [234] Dae-Hwan Kim. Evaluation of the performance of gpu global memory coalescing. *Evaluation*, 4(4), 2017. [195](#)