



ADVANCING OPTIMIZATION AND EVALUATION FOR DYNAMIC CAPACITATED ARC ROUTING PROBLEMS

By

HAO TONG

A thesis submitted to
the University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
University of Birmingham
May 2024

UNIVERSITY OF
BIRMINGHAM

University of Birmingham Research Archive

e-theses repository

This unpublished thesis/dissertation is copyright of the author and/or third parties. The intellectual property rights of the author or third parties in respect of this work are as defined by The Copyright Designs and Patents Act 1988 or as modified by any successor legislation.

Any use made of information contained in this thesis/dissertation must be in accordance with that legislation and must be properly acknowledged. Further distribution or reproduction in any format is prohibited without the permission of the copyright holder.

© Copyright by HAO TONG, 2024

All Rights Reserved

This project has received funding from the Honda Research Institute Europe.

ABSTRACT

The Capacitated Arc Routing Problem (CARP) is a challenging combinatorial optimization problem abstracted from real-world applications, such as waste collection and road gritting. It aims at scheduling a fleet of vehicles, each with a limited capacity, to serve a series of tasks in a graph spending the lowest possible total cost. In real applications, unexpected dynamic changes usually occur when vehicles are in their service, which potentially cause the current schedule of vehicles become worse or even infeasible. For example, the road may be closed or congested due to an accident, or new tasks might emerge after vehicles have already started their service. Consequently, the current schedule needs to be updated ensuring efficient service continuity, taking into account the different locations and various remaining capacities of the vehicles currently still in services. This dynamic scenario is regarded as Dynamic CARP (DCARP). Within this context, the information including the updated graph, the new set of tasks and the status of all vehicles is denoted as the DCARP instance which requires optimization to ensure an updated and efficient schedule.

This thesis mainly focuses on investigating the characteristics of the DCARP and proposing efficient optimization algorithms for its solution. In particular, we provide the following main contributions in this thesis:

1. We provide the first mathematical formulation of DCARP in the literature and developed a simulator to emulate the behavior of vehicles' service processes in the real world. This simulator offers a novel research platform for generating DCARP benchmarks to support

DCARP studies.

2. We propose a novel generalized meta-heuristic framework based on a virtual-task strategy for DCARP optimization. It enables all meta-heuristic algorithms designed for solving static CARP to be capable of effectively solving DCARP instances. Experimental results demonstrate that the proposed framework was able to significantly improve over state-of-the-art dynamic optimization algorithms in terms of total costs of the final solution.
3. We perform a fitness landscape analysis on various DCARP instances to understand which factors make a DCARP instance harder to solve. Empirical studies reveal that cost-related dynamic events had almost no effect on the difficulty of DCARP instances, while introductions of new tasks are the main factor making DCARP instances harder to solve.
4. We propose an experience-assisted optimization framework based on a novel solution building block strategy to deal with the DCARP scenario. A DCARP scenario comprises a series of DCARP instances that share similarities with each other. The proposed framework was able to improve the DCARP optimization for new DCARP instances in terms of total costs of initial solutions and convergence speed by leveraging optimization experiences extracted from the former DCARP instances.
5. To evaluate the efficacy of proposed algorithms in comparison to the theoretical global optimum, we derive a node matching lower bound for DCARP instances as the theoretical global optimum is always unknown for most DCARP instances. The deviations between the lower bound and DCARP solutions are calculated to measure the performance of our algorithms. The evaluation based on the derived lower bound indicates that there is still room for improvement between the lower bound and the optimized result.
6. To analyze the performance of proposed algorithms on real-world applications, we develop a DCARP benchmarking system based on SUMO (Simulation of Urban Mobility), integrated with real-traffic conditions. A series of DCARP scenarios with different settings are generated based on the proposed SUMO-based system using real-traffic data, thereby

evaluating the proposed algorithms to these scenarios for performance analysis. The evaluation demonstrates that dynamic optimization for CARP is necessary in real-world applications. Moreover, empirical studies also indicate that the adopted artificial DCARP benchmarks were sufficient for measuring the performance of optimization algorithms.

ACKNOWLEDGMENTS

Firstly, I want to express my deepest gratitude to my supervisors: Prof. Xin Yao, Dr. Leandro L. Minku, Dr. Stefan Menzel and Prof. Bernhard Sendhoff for their selfless and tireless guidance throughout my Ph.D. journey. Their insightful suggestions and constructive comments were invaluable, especially when I faced challenges or was uncertain about future work. Thanks to their mentorship, I have been shaped to a mature researcher and equipped with skills to conduct meaningful research, which is the treasure for my future work.

I am also immensely grateful to the members of my Thesis Group: Dr. Per Kristian Lehre and Dr. Mingqing Li. In each RSMG meeting, they always highlighted the weakness of my research and offered valuable advice, which significantly enrich my research.

There are too many people who gave me more or less help during my Ph.D. studies. While I cannot list every kind and lovely person, my heartfelt thanks go out to all of them. In particular, I want to my sincerely thanks Dr. Jialin Liu who provided immense help to get through the pandemic when I was a visiting student in Southern University Of Science and Technology during the Covid-19. Additionally, I still would like to express my special acknowledgments to these people, listed in alphabetical order by surname: Ms Bingyan Chen, Mr Haoxuan Ding, Mr Chao Jiang, Ms Jacqueline Benson, Mr Wei Li, Mr Ziming Liang, Mr Shishen Lin, Ms Ying Lu, Ms Ling Li, Dr. Yuan Ling, Ms Shanshan Mao, Ms Ke Ning, Mr Jiyuan Pei, Dr. Xiaoyu Qin, Mr Guangchen Xiao, Ms Mengjie Xu, Ms Yan Xu, Dr Feifei Zhao, Mr Qingquan Zhang, Mr Qiang Zhang, Mr Tianyang Zhang.

I would also like to thank the administration team of University of Birmingham and the

financial support from Honda Research Institute Europe (HRI-EU). Their supports were instrumental in the successful completion of my Ph.D. studies.

Lastly, I owe a debt of gratitude to my family for their unwavering support. My father, Mr Qiufu Tong, my mother, Ms Aijin Zeng and my sister, Ms Jin Tong, have always been the strongest shield behind me no matter what happens in my past and future life.

Reflecting on my journey, I also want to thank myself for never giving up for numerous challenges. As I embark on this new chapter of my life, I dedicate the poem "Let life be beautiful like summer flowers and death like autumn leaves" to myself and my future endeavors.

Contents

	Page
1 Introduction	1
1.1 Problem Description	4
1.1.1 Capacitated Arc Routing Problems	4
1.1.2 Dynamic Capacitated Arc Routing Problems	6
1.2 Research Questions	10
1.2.1 Optimization Framework for Dynamic CARP	10
1.2.2 Fitness Landscape Analysis for DCARP Instances	11
1.2.3 Experience Incorporation for Dynamic Optimization	11
1.2.4 Performance Evaluation for Dynamic Optimization	12
1.3 Contributions	13
1.3.1 Publications	15
1.4 Thesis Outline	16
2 Background	19
2.1 Mathematical Formulation	20
2.2 Literature Review	24
2.2.1 Solution Representation for CARP	24
2.2.2 Optimization Algorithms for CARP	25
2.2.3 Optimization Algorithms for DCARP	36
2.2.4 Overview of Other Dynamic Optimization Studies	40

2.3	Simulation and Benchmarking System	43
2.4	Summary	47
3	A Novel Generalized Framework for DCARP	49
3.1	Introduction	50
3.2	A Generalized Optimization Framework	52
3.2.1	Virtual Task Strategy	52
3.2.2	Proposed Framework based on Virtual-Task Strategy	57
3.3	Experimental Studies	59
3.3.1	Experimental Settings	60
3.3.2	Evaluating the Necessity of Rescheduling in DCARP	61
3.3.3	Analysis of the Effects of the Virtual-Task Strategy	65
3.3.4	Analysis of the Proposed Framework	68
3.4	Summary	74
4	Fitness Landscape Analysis for DCARP	77
4.1	Introduction	78
4.2	Fitness Landscape Analysis Approaches	79
4.3	Hybrid Local Search Algorithm	80
4.4	Fitness Landscape Analysis on DCARP instances	82
4.4.1	Instance Generation	82
4.4.2	Auto-Correlation	84
4.4.3	Local Optima	86
4.4.4	Approximate Global Optimum	93
4.5	Summary	96
5	Benefiting from Experience through Building Blocks	99
5.1	Introduction	100

5.2	Solution Building Block Strategy based Dynamic Optimization Framework	102
5.2.1	Building Blocks as Experiences	103
5.2.2	SBBA for DCARP-OC Instances	105
5.2.3	SBBA for DCARP-CT Instances	106
5.2.4	Discussion	108
5.3	Experimental Studies	109
5.3.1	Experimental Settings	109
5.3.2	Performance Measurement	112
5.3.3	Can SBBA Extract Optimization Experience	112
5.3.4	Can SBBA Promote Dynamic Optimization	118
5.3.5	How Well SBBA Performs under Different Severities of Dynamic Changes	122
5.4	Summary	123
6	Performance Measurement with Lower Bound	127
6.1	Introduction	128
6.2	Lower Bound for Static CARP Instance	129
6.3	Performance Evaluation with Node Matching Lower Bound for DCARP	134
6.3.1	Node Matching Lower Bound Algorithm	134
6.3.2	An Example of NMLB Approach	138
6.3.3	Graph Pruning Strategy	140
6.4	Experimental Studies	142
6.4.1	Experimental Settings	143
6.4.2	Tightness of the NMLB	144
6.4.3	Performance Measurement for the Optimization Algorithm	146
6.4.4	Efficiency of the Graph Pruning Strategy	147
6.5	Summary	150
7	Performance Evaluation with Real-world Traffic Simulation	153

7.1	Introduction	154
7.2	SUMO-based Benchmarking Framework	157
7.3	Experimental Studies	160
7.3.1	Experimental Setup	160
7.3.2	Efficacy of Algorithms in SUMO-based DCARP Scenarios	164
7.3.3	Comparing Algorithms with SUMO-based DCARP Instances	168
7.3.4	Further Practical Performance Analysis	172
7.4	Summary	176
8	Conclusions and Outlook	181
8.1	Conclusions	181
8.2	Future Work	186
A	Supplemental Materials	189
A.1	Full Results for Section 3.3.4: Analysis of the Effects of the Virtual-Task Strategy (Table 3.3)	189
A.2	The Pseudo-Code of the Instance Generator for Fitness Landscape Analysis .	194
	References	197

List of Figures

1.1	An example of the CARP instance. The depot is denoted by a triangle. The vertices are represented by circles. The gray solid lines represent the normal traversing edges. The triple solid line represents tasks.	5
1.2	The illustration of a DCARP scenario.	8
1.3	The illustration of generating a DCARP instance (right) from a static CARP instance (left) due to dynamic events. Road congestions happen on edges (v_0, v_8) and (v_0, v_9) , and a new task emerges on edge (v_0, v_1)	9
2.1	The architecture of our simulation system.	44
3.1	An example of constructing virtual tasks.	55
3.2	An example of demonstrating why the RF strategy is not efficient enough when outside vehicles have enough remaining capacities.	64
3.3	An example of demonstrating the advantages of considering new vehicles. . .	68
3.4	The number of instances for the restart strategy (RS) wins over ($\succ$), draws against ($\approx$) and loses to ($\prec$) the sequence transfer strategy (STS) for different ranges of average remaining tasks.	73
3.5	Critical difference diagram for the comparison of 8 algorithms against each other on <i>egl</i> with Friedman test and Nemenyi test. Groups of algorithms which are	74
4.1	Landscape analysis results in terms of correlation length.	85
4.2	Number of local optima over 10 DCARP instances for each of 64 settings. . .	87

4.3	Average time over all 100000 solutions to reach the corresponding local optima (unit: s). The presented result for each configuration is the mean result over 10 DCARP instances.	88
4.4	Landscape analysis results in terms of cost distribution.	90
4.5	Landscape analysis results in terms of distance between local optima.	92
4.6	Fitness distance correlation of global optimum for all 64 settings. The presented result for each configuration is the mean result over 10 DCARP instances.	94
4.7	Return probability of global optimum with one random step of <i>Single Insertion</i>	96
5.1	The structure of the SBBA-based dynamic optimization framework for the DCARP scenario. The left part is an illustration of the DCARP scenario, where dynamic optimization process starts with an initial static CARP instance and the control center will detect changes in the environment and determine to generate DCARP instances or not. The dynamic strategy at the right is used to assist the optimization for DCARP instances.	102
5.2	The difference of the generated solution's quality between the SBBA and other two strategies in all DCARP instances for DCARP-OC and DCARP-CT scenarios.	115
5.3	Results of the NC and Sim for all solutions generated by three strategies in all DCARP scenarios of all maps. White contours represent the estimated distribution for all data.	116
5.4	The distribution of solutions' Sim and NC for path-scanning (PS), greedy generation (GRD), and SBBA strategies.	117
5.5	The convergence curves of MAENS [113] and RTS [78] with SBBA (<i>blue lines</i>) and GRD (<i>orange lines</i>) strategies in some DCARP-OC and DCARP-CT scenarios.	119

5.6	The performance comparison between path-scanning (PS), greedy generation (GRD) and SBBA strategies on DCARP scenarios with different severity of dynamic changes.	122
6.1	An example of a DCARP instance. Node ‘0’ represents the depot. Values in brackets denote the cost and demand of the corresponding edge.	138
6.2	Example of applying NMLB approach in a DCARP instance.	139
6.3	A demonstration of proof for Theorem 2.	143
6.4	The demonstration of the relation between the tightness of the lower bound and the characteristics of the DCARP instance. Points with bigger size indicate instances with larger value of Gap	146
6.5	The distribution of all APD for DCARP-OC and DCARP-CT instances. . .	149
6.6	Relationship between the difference of computational time and the number of vertices in G_R whose degree are greater than 2.	149
7.1	The structure of SUMO-based benchmarking framework. Our main contribution is the “ <i>Controller</i> ” part.	157
7.2	Information about road network of Dublin.	161
7.3	The dynamic <i>busy degree</i> of each area for every 60 seconds.	162
7.4	The distribution of busy area and uncrowded area. The two positions of the depot.	163
7.5	One screenshot of DCARP scenario simulation in SUMO-based benchmarking framework in our experiment. It contains all cars and vehicles defined in the Dublin real traffic data, but only all unserved tasks (in blue) and in-service vehicles (in red) are highlighted.	164
7.6	An example of one vehicle’s route’s updates in the scenario 1 of Table 7.1. .	167

7.7	The convergence curves of MAENS-SBBA (<i>blue lines</i>) and MAENS-GRD (<i>orange lines</i>) algorithms in the first DCARP-OC and DCARP-CT scenarios. The results have been normalized according to Eq. (5.3), where TC_{min} and TC_{max} are the minimal and maximum total costs obtained during the optimization process.	172
7.8	The similarity (Sim calculated by Eq. 5.5) between solutions before and after dynamic optimization for DCARP-OC and DCARP-CT instances.	174
7.9	The vehicles' total working time in all DCARP scenarios after using dynamic optimization.	176

List of Tables

2.1	Types of dynamic events in DCARP.	43
3.1	Results of virtual-task strategy (VT) and return-first strategy (RF) on DCARP instances with different settings of remaining capacities from the <i>egl</i> dataset. The value in each cell represents “Mean $\pm$ Std” over 25 independent runs and the bold ones denote the better result on the DCARP instance based on the Wilcoxon signed-rank test with a significance level of 0.05. The penultimate row summaries the number of win-draw-lose of RF strategy versus VT strategy and the last row provides the p-values of the Wilcoxon signed-rank test with a significance level of 0.05 on instances with the same settings of all maps.	62
3.2	Results of MASDC and VT-MASDC on DCARP instances from the <i>egl</i> dataset. The value in each cell represents “Mean $\pm$ Std” over 25 independent runs and the bold ones denote the better result on the DCARP instance based on the Wilcoxon signed-rank test with the level of significance 0.05. The last row summaries the number of win-draw-lose of MASDC versus VT-MASDC. . .	66

3.3	Results of VT-RTS, VTtr-RTS, VT-ILMA, VTtr-ILMA, VT-MAENS, VTtr-MAENS, VT-MASDC and VTtr-MASDC on the <i>egl</i> dataset. The values in each cell represent the “MEAN $\pm$ STD” with the average ranking (in the brackets) w.r.t the average total cost over 25 independent runs. The bold values are the better results under the Wilcoxon signed-rank test with a significance level of 0.05 between restart and inheriting strategies in an algorithm for a DCARP instance. The last three rows summarize the number of win-draw-lose of restart strategy versus inheriting strategy in each algorithm and the p-values of the Wilcoxon signed-rank test with a significance level of 0.05 on all instances.	71
4.1	Dynamic factors and state factors investigated in this work	84
5.1	Results of generated solutions’ quality for SBBA, PS and GRD strategies on DCARP-OC (left part) and DCARP-CT (right part) scenarios from the EGL dataset. The value in each cell represents “MEAN $\pm$ STD” of NC for all DCARP instances in 10 generated DCARP scenarios for each map. The highlighted values denote the best results among three strategies under the Friedman test with a significance level of 0.05.	113
5.2	Comparison of MAENS and RTS meta-heuristic algorithms using SBBA against GRD in DCARP-OC and DCARP-CT scenarios. The values in each cell represent the number of “win-draw-lose” for the SBBA strategy based on Wilcoxon signed-rank tests with a significance level of 0.05 in 10 DCARP scenarios of each map.	120

6.1	The gap between the obtained lower bound and the global optimum, i.e., Gap calculated by Eq. 6.5, on DCARP instances created based on the gdb static dataset. N_n , N_v and N_t denote the number of nodes, the number of required vehicles and the number of tasks, respectively. $NMLB$ and OPT represent the NMLB and the global optimal solution's total costs for the DCARP instance.	145
6.2	The approximate percentage deviation performance, i.e., APD , for the MAENS-SBBA on a subset of DCARP-OC (<i>left</i>) and DCARP-CT (<i>right</i>) instances used in Chapter 5, calculated by Equation (6.6). $Rank$ represents the rank of the instance after sorting by the value of APD . N_n represents the number of nodes in the corresponding required graph. N_t represents the number of tasks on the instance. $NMLB$ represents the lower bound obtained by NMLB approach. $Best^*$ represents the median best solution's cost obtained by the MAENS-SBBA over 25 independent runs on the corresponding instance. T_O and T_P denote the computational time (unit: <i>sec</i>) for the calculating the lower bound with and without pruning strategy, respectively.	148
7.1	The results of using dynamic optimization (MAENS-SBBA algorithm) and without dynamic optimization in each generated DCARP-CT scenario. . . .	166
7.2	Results of generated solutions' quality for SBBA, PS and GRD strategies, and results of the best solutions obtained by MAENS-SBBA, MAENS-GRD and MASDC algorithms on SUMO-based DCARP-OC scenarios. For the strategy comparison, the results represent the total costs of the best solution produced by each strategy, with the best result highlighted. The algorithm comparison results present the "MEAN $\pm$ STD" of the best solutions acquired over 25 runs for each algorithm and the best result(s) under statistical tests are highlighted.	169

7.3	Results of generated solutions' quality for SBBA, PS and GRD strategies, and results of the best solutions obtained by MAENS-SBBA, MAENS-GRD and MASDC algorithms on SUMO-based DCARP-CT scenarios. For the strategy comparison, the results represent the total costs of the best solution produced by each strategy, with the best result highlighted. The algorithm comparison results present the "MEAN $\pm$ STD" of the best solutions acquired over 25 runs for each algorithm and the best result(s) under statistical tests are highlighted.	170
A.1	Results of VT-RTS, VTtr-RTS, VT-ILMA, VTtr-ILMA, VT-MAENS, VTtr-MAENS, VT-MASDC and VTtr-MASDC on the <i>egl</i> dataset (E1-A $\sim$ E2-C). The values in each cell represent the "MEAN $\pm$ STD" with the average ranking (in the brackets) w.r.t the average total cost over 25 independent runs. The bold values are the better results under the Wilcoxon signed-rank test with the level of significance 0.05 between restart and inheriting strategies in an algorithm for a DCARP instance.	190
A.2	Results of VT-RTS, VTtr-RTS, VT-ILMA, VTtr-ILMA, VT-MAENS, VTtr-MAENS, VT-MASDC and VTtr-MASDC on the <i>egl</i> dataset (E3-A $\sim$ E4-C). The values in each cell represent the "MEAN $\pm$ STD" with the average ranking (in the brackets) w.r.t the average total cost over 25 independent runs. The bold values are the better results under the Wilcoxon signed-rank test with the level of significance 0.05 between restart and inheriting strategies in an algorithm for a DCARP instance.	191

A.3	Results of VT-RTS, VTtr-RTS, VT-ILMA, VTtr-ILMA, VT-MAENS, VTtr-MAENS, VT-MASDC and VTtr-MASDC on the <i>egl</i> dataset (S1-A ~ S2-C). The values in each cell represent the “MEAN $\pm$ STD” with the average ranking (in the brackets) w.r.t the average total cost over 25 independent runs. The bold values are the better results under the Wilcoxon signed-rank test with the level of significance 0.05 between restart and inheriting strategies in an algorithm for a DCARP instance.	192
A.4	Results of VT-RTS, VTtr-RTS, VT-ILMA, VTtr-ILMA, VT-MAENS, VTtr-MAENS, VT-MASDC and VTtr-MASDC on the <i>egl</i> dataset (S3-A ~ S4-C). The values in each cell represent the “MEAN $\pm$ STD” with the average ranking (in the brackets) w.r.t the average total cost over 25 independent runs. The bold values are the better results under the Wilcoxon signed-rank test with the level of significance 0.05 between restart and inheriting strategies in an algorithm for a DCARP instance.	193

List of Acronyms

ACO	Ant Colony Optimization.
ARP	Arc Routing Problem.
CARP	Capacitated Arc Routing Problem.
CI	Cost Increase Dynamic Event.
CD	Cost Decrease Dynamic Event.
CPP	Chinese Postman Problem.
CVRP	Capacitated Vehicle Routing Problem.
DCARP	Dynamic Capacitated Arc Routing Problem.
DCARP-CT	DCARP Scenarios/Instances with Both Cost-Changing and Task-Changing Dynamic Events.
DCARP-OC	DCARP Scenarios/Instances only with Cost-Changing Dynamic Events.
DO-SBBA	Solution Building Block Adaptation Strategy based Dynamic Optimization Framework.
DSS	Distance-based Split Scheme.
DVRP	Dynamic Vehicle Routing Problem.
FDC	Fitness Distance Correlation.
GOFVT	Generalized Optimization Framework with Virtual Tasks.
GP	Genetic Programming.
GRD	Greedy Generation Strategy.

HyLS	Hybrid Local Search Algorithm.
ILMA	Improved Lacomme's Memetic Algorithm.
LMA	Lacomme's Memetic Algorithm.
LON	Local Optimal Network.
MAENS	Memetic Algorithm with Extended Neighborhood Search.
MAENS-GRD	MAENS with Virtual Task Strategy with GRD Solution Constructive Heuristic.
MAENS-SBBA	MAENS with Virtual Task Strategy Embed to DO-SBBA Framework.
MASDC	Memetic Algorithm with New Split Scheme for DCARP.
MCMP	Minimum Cost Matching Problem.
MLB	Matching Lower Bound.
ND	New Demand Dynamic Event.
NDLB	Node Duplication Lower Bound.
NSLB	Node Scanning Lower Bound.
NT	New Tasks Dynamic Event.
NMLB	Node Matching Lower Bound.
OX	Order Crossover.
OV	Outside Vehicles.
PS	Path-Scanning Strategy.
RF	Return-First.
RPP	Rural Postman Problem.
RK-OPX	Random Key based One Point Crossover.
RS	Restart Strategy.
RQ	Remaining Capacities of Outside Vehicles.
RTS	Tabu Search Algorithm with a Global Repair Operator for DCARP.
RTS-GRD	RTS with Virtual Task Strategy with GRD Strategy.

RTS-SBBA	RTSwith Virtual Task Strategy Embed to DO-SBBA.
SA	Solution Archive.
SBBA	Solution Building Block Adaptation Strategy.
SBX	Sequence Based Crossover.
SI	Single Insertion.
STS	Sequence Transfer Strategy.
SUMO	Simulation of Urban MObility (A Traffic Simulation Software).
TSA	Tabu Search Algorithm.
TSP	Travelling Salesman Preroblem.
VRP	Vehicle Routing Problem.
VT	Virtual-Task.
VT-ILMA	ILMA Algorithm Integrated to GOFVT with the Restart Strategy.
VT-MAENS	MAENS Algorithm Integrated to GOFVT with Restart Strategy.
VT-MASDC	MASDC Algorithm Integrated to GOFVT with Restart Strategy.
VT-RTS	RTS Algorithm Integrated to GOFVT with Restart Strategy.
VTtr-ILMA	ILMA Algorithm Integrated to GOFVT with Sequence Transfer Strategy.
VTtr-MAENS	MAENS Algorithm Integrated to GOFVT Sequence Transfer Strategy.
VTtr-MASDC	MASDC Algorithm Integrated to GOFVT Sequence Transfer Strategy.
VTtr-RTS	RTS Algorithm Integrated to GOFVT with Sequence Transfer Strategy.

List of Symbols

G	Graph $G = (V, A)$.
V	Set of vertices.
A	Set of arcs.
E	Set of edges.
v_0	Depot.
$dc(u)$	Deadheading(traversing) cost of an arc $u \in A$ (ID).
$sc(u)$	Serving cost of an arc $u \in A$ (ID).
$dm(u)$	Demand of an arc $u \in A$ (ID).
I_t	DCARP instance generated at a time point t .
E_R	Set of tasks.
N_T	Number of tasks.
N_{veh}	Maximum number of vehicles.
Q	Capacity of empty vehicles.
N_{ov}	Number of outside vehicles.
q_k	Remaining capacity of the k^{th} outside vehicle.
sp_k	Stop point of the k^{th} outside vehicle.
V_{ov}	Set of outside vehicles' stop locations.
RQ	Set of outside vehicles' remaining capacities.
S	DCARP solution.

r_k	k^{th} route in a solution.
$t_{k,i}$	i^{th} task in k^{th} route.
l_k	Number of tasks in k^{th} route.
RC_{r_k}	Total cost of k^{th} route r_k .
$TC(S)$	Total cost of solution S .
$mdc(v_i, v_j)$	Minimal total deadheading cost from vertex v_i to v_j .
$head_{t_{k,i}}$	Head node of task $t_{k,i}$.
$tail_{t_{k,i}}$	Tail node of task $t_{k,i}$.
$\overline{dc(u)}$	Expected deadheading cost of an arc $u \in A$ (ID).
vt	Virtual task.
NE_R	Set of newly added tasks.
nt_i	i^{th} newly added task.
N_{NE_R}	Number of new tasks.
N_{RT}	Number of remaining tasks.
G_R	Required graph $G_R = (V_R, E_R)$.
V_R	Set of nodes including the depot and nodes which are incident with at least one task.
A_R	Set of arcs between each pair of vertices in V_R .
$d(v_i)$	Degree of $v_i \in G_R$.
NS	New solution.
bb	Building block.
$BLOCK$	Set of building block.
W_T	Total demands of all tasks.
k_0	Minimum number of vehicles required for a (D)CARP instance.
Sim	Similarity between CARP solutions.
C^*	Cost of the optimal (D)CARP instance.

C_T	Total serving costs of all tasks.
C_1	Total costs of artificial edges directly linking to the depot in a (D)CARP solution.
C_2	Total costs of artificial edges linking tasks in a (D)CARP solution.
C_{links}^*	Total costs of all artificial links in S^* of a DCARP instance.
G_x	Auxiliary graph.
V_x	Set of nodes for building G_x .
E_x	Set of edges for building G_x .
A,B,C,D	Subsets of nodes in V_x and $V_x = \mathbf{A} \cup \mathbf{B} \cup \mathbf{C} \cup \mathbf{D}$.
C_{MCMP}	Cost of optimal solution of the minimum cost matching problem on G_x .
$psize$	Population size for the meta-heuristic algorithm.
$R(\tau)$	Auto-correlation.
l	Correlation Length.
NC	Normalized Cost.
ANC	Average Normalized Cost.
Gap	Disparity between the lower bound and the global optimum.
OPT	Objective value of the global optimum.
LB	Lower bound.
$Best^*$	Best optimized objective value.
APD	Approximate percentage deviation.
T_O	Computational time for the lower bound approach.
T_P	Computational time for the lower bound approach with pruning strategy.
T_{WDO}	Computational time without using dynamic optimization.
T_{DO}	Computational time using dynamic optimization.
T_{Diff}	Time difference.
$\hat{v}$	Average speed of vehicles in a road in the last 60 seconds.

List of Symbols

v^{max}	Limited maximum speed of a road.
N_{roads}	Number of roads in an area.
α	Crowded degree of a road.
β	Busy degree of an area.

Chapter One

Introduction

In response to the increasing demand of mobility services across various sectors in the real world, such as foundational city services, bike-sharing programs, last-mile delivery services, etc., the transportation and logistics industry play an essential role in providing mobility services for demand management. Consequently, optimizing operations and improving efficiency have become increasingly important for this industry. The mobility solutions offer an option to achieve this goal allowing decision makers to effectively manage and optimize fleets' routes, track shipments in real time, and enhance the overall services of operations. The optimal routing and path planning, commonly known as the routing problem, is one of the key challenges within mobility solutions for improving the service efficiency and reducing the service costs such as energy and maintenance expenditures.

The routing problem is one of the most extensively researched topics within the area of combinatorial optimization and operational research. It is aimed at scheduling a fleet of vehicles to provide service in a graph with a set of feasible routes [101], often with a focus on minimizing overall costs. Moreover, the route corresponding to a specific vehicle represents the pathway it traverses in the graph for completing its designated service. Routing problem [101] has two main categories: the node routing problem, known as the Vehicle Routing

Problem (VRP) [13, 28], and the Arc Routing Problem (ARP) [20, 86]¹. The VRP has been widely investigated since its initial formalization in 1954 by Dantzig et al. [21] studying a large-scale travelling salesman problem. Given a graph comprising a set of vertices and edges, there exists some customers on a subset of vertices in the graph requiring vehicles to serve their demands. These vehicles locate at one or multiple nodes, known as the depots, in the graph. The target of the VRP is to find routes for these vehicles to serve all demands and minimize the overall operational costs. In general, each customer must be served once by one vehicle and all vehicles are required to start from the depot, and return to the depot after the service as well. Thus, the travelling salesman problem (TSP) can be viewed as a special case of the VRP in which all vertices in the graph are required to be served.

Similarly, the ARP also targets to assign vehicles from a depot to serve customers in a graph composed of vertices and edges. Unlike the VRP, however, customers in ARP spread over edges in the graph rather than the vertices, and the dual service directions for each edge significantly increase the complexity for solving the ARP compared to the VRP [96]. The study of the ARP can be traced back to 1735 when Euler investigated the Königsberg bridge problem [105]. After that, the first formal optimization for the ARP was introduced in 1960 by Guan when he addressed the mail delivery problem, known as Chinese Postman Problem (CPP) [83]. In the CPP, each arc within the graph has demands requiring to be served. The objective is to assign vehicles to serve all these arcs in the graph while minimizing the total costs. In addition, if customers in the CPP only distribute over a subset of arcs in the graph instead of all arcs of the graph, it becomes to the well-known Rural Postman Problem (RPP) [93].

In real-world applications, vehicles usually have a maximum capacity, and the aggregate demands in a problem are also often greater than the capacity of one vehicle. Accordingly, we have to consider that the total demands allocated to one vehicle cannot exceed its ca-

¹Some studies also classified the arc routing problem as a subtype of the vehicle routing problem.

capacity. This is the well-known variant of the routing problem, commonly referred to as the Capacitated Vehicle Routing Problem (CVRP) and the Capacitated Arc Routing Problem (CARP). The CARP constitutes the main focus in this thesis. Compared with the CPP and RPP, the CARP is more realistic and also much harder to solve due to the more intricate search space caused by the additional capacity constraints.

Since the first investigation of the CARP in 1981 by Golden and Wong [37], many optimization algorithms have been developed to address the CARP including both exact algorithms and heuristic algorithms. Although the exact algorithm, such as branch-cut-and-price [70] and column generation [99], can solve the problem to the optimality, it is only practical for relatively low dimensional CARP instances since the CARP is an NP-hard problem [37]. Therefore, many heuristic algorithms and meta-heuristic algorithms [24], are proposed in the literature for solving the problem based on the characteristics of the CARP [86]. For example, the Tabu Search [14], the Genetic Algorithm [53], the Memetic Algorithm [113], and the Ant Colony Optimization [106]. The special heuristic search operators were designed in these algorithms to facilitate the finding of relatively near-optimal solutions for the CARP, rendering them powerful for both small and large instances of the CARP. As a result, heuristic algorithms, particularly those of meta-heuristic algorithms, have attracted much attention in the literature.

In the practical CARP application, a lot of unforeseen dynamic changes often occur during the service process arising from the transportation environment or inherent factors of the problem itself. These uncertain factors are likely to substantially impact the vehicles' follow-on service. Therefore, the service routes are required to be updated due to the influence of dynamic events in the CARP. This refers to a significant variant of the CARP known as the Dynamic Capacitated Arc Routing Problem (DCARP). In this thesis, we focus on the meta-heuristic algorithms designed for DCARP. In the literature, the specific DCARP was first investigated when considering a salting route optimization problem in 2005 [44].

Subsequently, Liu et al. [59, 60] first focused on the general DCARP, which takes into account a broader range of dynamic events, such as road congestion, road closures, and new tasks. So far, only a few studies have concentrated on addressing DCARP optimization, and effective algorithms for solving DCARP are still missing. These issues will be further discussed in Section 2.2. Consequently, the primary research objective of this thesis is to leverage meta-heuristic algorithms efficiently solving the DCARP.

The structure of this chapter is organized as follows. Section 1.1 gives the general descriptions of the CARP and DCARP with clear examples. Section 1.2 outlines the research problems addressed in this work. Section 1.3 summarizes the key contributions made by this thesis. Section 1.4 provides the overall organization of the thesis.

1.1 Problem Description

1.1.1 Capacitated Arc Routing Problems

The Capacitated Arc Routing Problem (CARP) is a classical combinatorial optimization problem with a range of collection and delivery applications in the real world, such as waste-collection [54], winter gritting [44], mail delivery [27] and others. A map, regarded as a graph, contains a set of vertices V and arcs E in a graph $G = (V, E)$. For each edge $e = (v_i, v_j) \in E$, it has a demand $dm(e)$. If the $dm(e)$ is greater than zero, this edge is required to be served by a vehicle with a serving cost $sc(e)$. Otherwise, it does not need to be served and only has a deadheading cost (traversing cost) $dc(e)$. In general, edges with demands are called *tasks* in the context of CARP. Suppose there are a fleet of vehicles with limited capacity of Q located at the depot, the CARP aims to assign these vehicles to serve all tasks in the graph with as low as possible total cost. Take the waste-collection problem as an example,

there are lots of wastes in many streets of the city, and trucks with the limited capacity are assigned to collect all wastes starting from the depot [54]. The streets required to be served, i.e., some waste is required to be collected, have a serving cost for trucks, while trucks only need to cost the deadheading cost when traversing the streets without wastes.

To clearly illustrate the CARP, we present a specific CARP instance as an example in Figure 1.1. In this figure, the depot is denoted by a triangle, while the vertices are represented by circles. In addition, gray solid lines represent the normal traversing edges, and triple solid lines represent tasks required to be served. In this example, three vehicles with predefined maximum capacities are scheduled to serve all tasks.

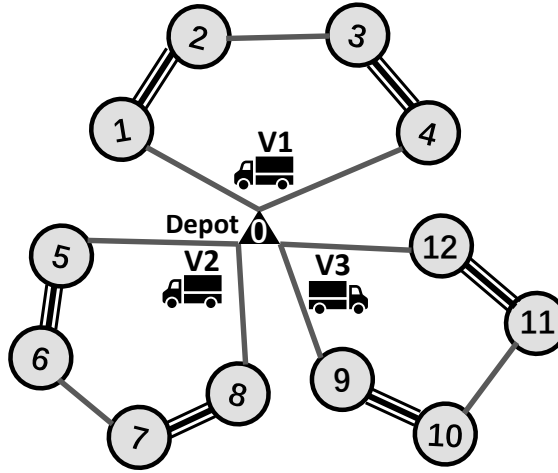


Figure 1.1: An example of the CARP instance. The depot is denoted by a triangle. The vertices are represented by circles. The gray solid lines represent the normal traversing edges. The triple solid line represents tasks.

Over the past decades, many variants of the CARP have been researched in the literature [19, 20, 86]. The basic CARP introduced above is known as the undirected CARP, in which each edge is direction-agnostic and can be traversed from both directions. However, in practice, many streets only can be accessed from only one direction. Therefore, the directed CARP variant addresses such a scenario that considers edges with specific traversal directions, termed arcs [69]. Moreover, when a CARP instance contains both undirected

edges and directed arcs, it is referred to as the mixed CARP [38]. Besides, a lot of practical conditions have also been considered in the CARP, leading to a series of specialized CARP variants. For instance, Xing et al. [133] took multiple depots into account and optimized routes with different starting and ending depots. Sipahioglu et al. [110] modeled the multi-robot path-planning problem to a special CARP, where robots consume the capacity, i.e. battery's energy, even when they are traversing an edge without demands. Belenguer et al. [7] introduced the split-delivery CARP allowing the edge demand being able to be served by multiple vehicles. Furthermore, time-related variants of CARP have also been investigated in the literature. For example, researchers considered the CARP with time-window, where tasks are required to be completed within a pre-defined time window [121], and the periodic CARP where edges should be served with a frequency [79]. Additionally, Mei et al. [80] addressed a multi-objective CARP that balances two conflicting objectives: minimizing total costs and the makespan of routes simultaneously. Large-scale CARP with a very large number of tasks and vertices has also attracted significant attention in the literature. Mei et al. proposed various strategies for managing the expansive search space of such practical applications [75–77]. Furthermore, Tang et al. [114] also handled the large-scale CARP employing hierarchical decomposition strategies to enhance scalability.

1.1.2 Dynamic Capacitated Arc Routing Problems

However, all these studies concentrate on static CARPs, where the problem remains static during the entire time of a solution's execution. In real applications, dynamic changes usually happen when vehicles are in service, i.e., when a solution is partially executed, thus influencing the vehicles' follow-on service. For example, a road may be closed due to an accident or new tasks may emerge during the vehicles' service². When that happens, a new graph, i.e. a new problem instance, is formed, in which vehicles would stop at different

²A full list of considered dynamic events in this thesis listed in Table 2.1 in Section 2.3.

locations, labelled as *outside vehicles*, with various amounts of remaining capacities. As a result, the current schedule may become inferior or even infeasible. Dynamic CARP (DCARP) in our work thus aims at re-scheduling the service plan [59, 60]. For clarity, the following three different concepts are used throughout the thesis:

- **DCARP:** A variant of CARP where the status of a graph is changed due to dynamic events occurring during a CARP solution's execution.
- **DCARP Instance:** The updated graph with some outside vehicles after the dynamic events happen.
- **DCARP Scenario:** A scenario contains a series of DCARP instances with the whole service process, starting from executing an initial solution in the original CARP map until all tasks are served.

In general, the DCARP scenario considered in this thesis is presented in Figure 1.2 starting with an initial static CARP instance. An optimization algorithm can be employed to optimize the static CARP instance, and the best solution obtained from the optimization will be used to schedule vehicles to serve the pre-defined tasks. Subsequently, we assume there is a control center which can receive all traffic and environment information, and it will detect changes in the environment [85]. Whenever any dynamic events occur while vehicles are in service, the control center will report the type of dynamic events and generate a new DCARP instance according to the occurred dynamic events. The optimization algorithm will then dynamically optimize the newly generated DCARP instance. After the dynamic optimization process for the new DCARP instance, an updated solution will be obtained and deployed to the environment to update the service plan of all vehicles. The DCARP scenario will terminate only when all tasks have been served.

Therefore, the DCARP instance is the main optimization problem in a DCARP scenario,

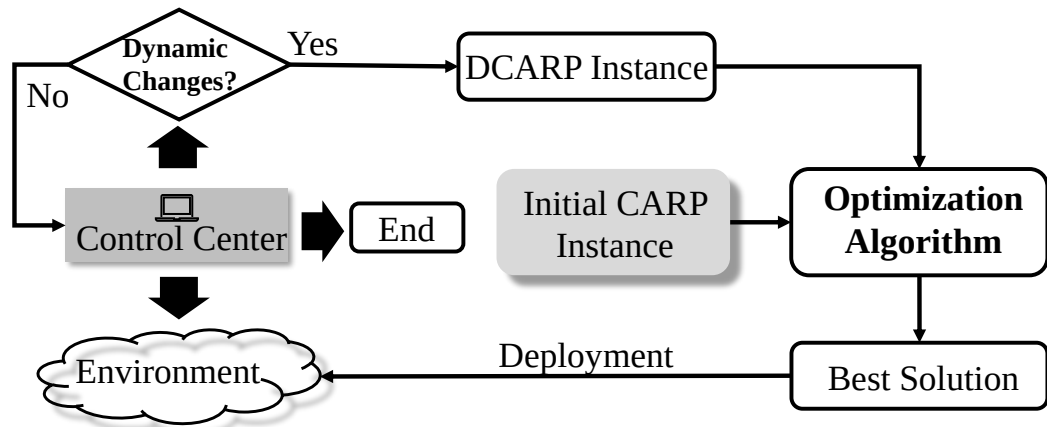


Figure 1.2: The illustration of a DCARP scenario.

generated when dynamic events occur during the service process. We present an example illustrating the generation of a new DCARP instance in Figure 1.3. The left panel displays the initial CARP instance and the scheduled routes for vehicles. The right panel presents the generated DCARP instance due to dynamic events, where three dynamic events happen including road congestions on edges (v_0, v_8) and (v_0, v_9) , and a new task emerges on edge (v_0, v_1) . Simultaneously, three vehicles are located at v_2, v_7, v_{10} with different remaining capacities since they have already served some tasks according to their original schedule. Thus, it requires to update the schedule for remaining services route of these vehicles to efficiently complete all remaining tasks, which is known as the optimization for the DCARP instance. A detailed mathematical formulation of DCARP is given later in Section 2.1.

It is worth noting the term “DCARP” was also used for various different DCARP variants. Maniezzo and Roffilli [69] used the term “DCARP” to represent the directed CARP where each edge in the graph can only be traversed in a specific direction. Mei et al. [81] also used the term “DCARP” to denote the uncertain CARP (UCARP), which is different from the meaning of DCARP in this thesis. Uncertain CARP focuses on robust optimization [58, 81], where one is interested in finding solutions that are *robust* to uncertainties, such as changing degrees of congestion or level of demands. In these CARP variants, demands

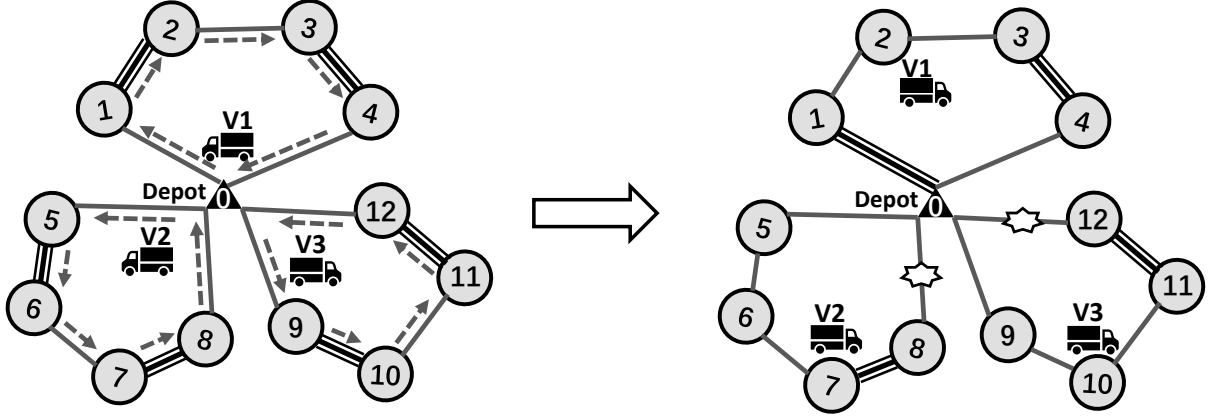


Figure 1.3: The illustration of generating a DCARP instance (right) from a static CARP instance (left) due to dynamic events. Road congestions happen on edges (v_0, v_8) and (v_0, v_9) , and a new task emerges on edge (v_0, v_1) .

of tasks and the costs of arcs in the problem are assumed to follow random distributions. Then, the problem is solved offline and the solution will not update during vehicles' services once the solution is deployed. Besides, the problem instances to be solved can also be samples from deterministic CARP instances, which are collected from real-word applications [124] and a robust solution is optimized for these deterministic CARP instances. These are solutions obtained offline *before deployment* that one hopes will not deteriorate too much under different environment conditions that may be encountered during deployment. Therefore, UCARP is a considerably different problem from the one tackled in our thesis. Moreover, even though the UCARP solutions are hoped not to deteriorate too much under certain environment conditions (e.g., different amounts of congestion), they are unlikely to be optimal for each individual environment condition that may be encountered during deployment. As a result, dynamic optimization, which is the focus of our thesis, is required to track the moving optimum and update the schedule for completing the remaining tasks when dynamic events deteriorate the current deployed solution during vehicles' services.

Moreover, it is also worth noting that the dynamic optimization is also ubiquitous in both continuous and combinatorial problem domains, and numerous effective algorithms have

been developed to address the dynamic changes in these problems [72, 89]. Typically, for almost all dynamic optimization problems investigated in the literature, uncertain factors only affect the objective functions, causing the change of global optima. However, in the case of our studied DCARP, dynamic events influence the decision variable space as well, wherein the problem's dimension and the previous solution's feasibility is likely to change after dynamic events. For example, tasks that are already completed will be removed while the newly emerged tasks will be added, thereby changing the number of tasks. Moreover, the potential road closure event makes previous solutions infeasible. As a result, the decision variable space for the newly generated DCARP instance is different with the former DCARP instances causing solutions obtained for former DCARP instances are not suitable for the new DCARP instances.

1.2 Research Questions

1.2.1 Optimization Framework for Dynamic CARP

The CARP has been studied a lot over the past decades and there is a rich literature on existing CARP optimization algorithms. However, they are not applicable to DCARP instances since they use the assumption that all vehicles start at the depot and have the same capacities, which is not the case in DCARP. For the DCARP, the calculation of cost and capacity violation of the routes corresponding to the outside vehicles are required to be specifically considered due to the fact that outside vehicles have different locations and remaining capacities. Since both CARP and DCARP aim to schedule vehicles to complete a set of tasks in a graph, the solution representation and search operators for the CARP can be extended to solve DCARP. It is more complicated to propose the optimization algorithm for the DCARP from scratch compared with benefiting from the existing CARP algorithms

[113], which have been shown to have powerful optimization ability, being able to efficiently solve complex CARP instances. It is naturally leading to that applications for solving CARP provide valuable resources to address the challenges of DCARP. Therefore, our first research question:

RQ1: How can we benefit from the powerful algorithms existing for CARP in order to better solve DCARP?

1.2.2 Fitness Landscape Analysis for DCARP Instances

Existing literature lacks information about which dynamic factors are likely to cause DCARP instances to be harder to solve. This seriously hinders the ability to improve the performance of DCARP algorithms. Fitness landscape analysis could bridge this gap. It is a very popular approach for revealing the structure of search space. It targets to investigate features of optimization problems, such as the fitness distribution in the search space and the structure of optima [67]. It helps to better understand the characteristics of problem instances, supporting the design of more suitable algorithms for the specific problem [67]. So, our second research question is:

RQ2: What makes the DCARP instance hard to solve for optimization algorithms?

1.2.3 Experience Incorporation for Dynamic Optimization

As introduced above, a DCARP scenario is composed of a series of DCARP instances. A new DCARP instance is generated based on the preceding DCARP instance and a partially executed solution. As a result, a new DCARP instance in a DCARP scenario shares similarities

with its preceding DCARP instance. Intuitively, the optimization experience and solution information of the former DCARP instance could be extracted to promote the optimization for the new DCARP instance in a DCARP scenario. However, current existing works do not consider and utilize such optimization experience, wasting valuable knowledge that could be used to enhance the optimization process for DCARP instances. Instead, they always optimize the new DCARP instance from scratch once dynamic events happen. Therefore, one research question arise for experience assisted dynamic optimization:

RQ3: How to extract optimization experience from a DCARP instance and integrate into a meta-heuristic framework for enhanced performance in different DCARP scenarios?

1.2.4 Performance Evaluation for Dynamic Optimization

Currently, the optimization performance of the proposed optimization algorithms is evaluated in a set of artificially designed benchmark problems and compared to various state-of-the-art algorithms through statistical tests. In this way, our experimental studies can only demonstrate the proposed algorithms are better than the existing algorithms. However, we still have no insight about the absolute performance of the proposed algorithms relative to the theoretical global optimum, which is crucial for assessing the remaining potential for improvement in the optimization algorithms. Nevertheless, the theoretical optimum is often unknown for most DCARP instances since DCARP is also an NP-hard problem. Thus, the lower bound would be very important for measuring the performance of the optimization algorithm. Therefore, we have the research question:

RQ4: How can we derive a lower bound for evaluating the DCARP solution quality rather than relying solely on comparison with baseline algorithms?

On the other hand, the dynamic events in the existing benchmarks are all artificially generated based on various distributions instead of any real-world scenarios. Effective algorithms demonstrated only in such conditions are likely to fail in real traffic environments, especially for DCARP, where the re-scheduling highly depends on actual traffic conditions and the status of instances. A dynamic environment containing real traffic data and conditions is required to evaluate the proposed algorithms. Therefore, a further research question is:

RQ5: How can we evaluate and how well do our proposed DCARP algorithm perform in real-world scenarios in addition to artificial benchmarks?

1.3 Contributions

In this thesis, our main focus is on investigating the characteristics of DCARP and developing effective optimization algorithms to solve the DCARP. In order to have better solutions when solving DCARP, our main contributions in this thesis are as follows:

1. We provide a new mathematical formulation of DCARP and design a simulator to simulate the behavior of vehicles' service processes in the real world in Section 2.1 and 2.3 of Chapter 2 as preliminary studies for our work. The simulator offers a novel research platform for generating DCARP benchmarks to support DCARP studies.
2. A novel framework capable of generalizing almost all existing algorithms designed for static CARP to the DCARP context is proposed in Chapter 3 to answer RQ1. The framework converts a DCARP instance into a "static" CARP instance by introducing the idea of "virtual tasks", which enables outside vehicles (with potentially partial capacity) to be interpreted as vehicles located at the depot (with their full capacity). The DCARP instance can then be solved as if it was a static CARP instance, enabling

the powerful existing static CARP algorithms to solve it.

3. The first systematic fitness landscape analysis for DCARP is performed, including several local and global optima related features, in Chapter 4 to answer RQ2. Multiple landscape properties are compared between instances generated with different dynamic events to investigate what factors make DCARP instances easier or harder to solve. The empirical studies show that new tasks are the main factor influencing the difficulty of the instances, and instances requiring additional vehicles for the remaining tasks are harder to solve than instances with other features.
4. To leverage the optimization experience from the former instance to facilitate dynamic optimization in a DCARP scenario, a new concept of *building block* for extracting optimization experience contained in historical solutions is introduced. The solution *building block* as the experience is employed to generate better initial solutions for new DCARP instances. Consequently, a novel experience-assisted dynamic optimization framework is proposed to enhance the optimization of new DCARP instances in Chapter 5 to answer RQ3.
5. The performance of the proposed meta-heuristic algorithms for DCARP is evaluated through assessing the quality of DCARP solutions based on a newly derived lower bound in Chapter 6 to answer RQ4. The first lower bound approach for DCARP, named node matching lower bounds (NMLB), is proposed, converting the lower bound problem to a minimum cost matching problem (MCMP).
6. To evaluate the efficacy of algorithms in a real-world environment, we developed the first DCARP scenario benchmarking system containing real traffic data in Chapter 7 to answer RQ5. The framework is able to integrate the solution produced by DCARP algorithms into a transportation environment that simulates real traffic conditions. A set of DCARP scenarios are generated based on the designed simulation system

with real transportation data, thereby facilitating the performance evaluation of the proposed algorithms in the real-world application.

1.3.1 Publications

The published or submitted papers related to this thesis during my doctoral studies are as follows:

- **Published papers**

1. Tong, Hao, Leandro L. Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao. “Towards novel meta-heuristic algorithms for dynamic capacitated arc routing problems.” In *Parallel Problem Solving from Nature-PPSN XVI*, pp. 428-440. Springer International Publishing, 2020. (This paper is an early investigation on DCARP for an improved splitting scheme. But it is not included into this thesis.)
2. Tong, Hao, Leandro L. Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao. “A novel generalized meta-heuristic framework for dynamic capacitated arc routing problems.” *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 6, pp. 1486-1500, Dec. 2022. (This paper mainly addresses contributions 1 and 2, and support Sections 2.1, 2.3 and Chapter 3.)
3. Tong, Hao, Leandro L. Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao. “A hybrid local search framework for the dynamic capacitated arc routing problem.” In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 139-140. 2021. (This paper addresses part of the contribution 3, and supports Chapter 4.)
4. Tong, Hao, Leandro L. Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao. “What makes the dynamic capacitated arc routing problem hard to solve: insights

from fitness landscape analysis.” In Proceedings of the Genetic and Evolutionary Computation Conference, pp. 305-313. 2022. (This paper mainly addresses the contribution 3, and supports Chapter 4.)

5. Tong, Hao, Leandro L. Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao. “Benchmarking dynamic capacitated arc routing algorithms using real-world traffic simulation.” In 2022 IEEE Congress on Evolutionary Computation (CEC), pp. 1-8. IEEE, 2022. (This paper mainly addresses the contribution 6, and supports Chapter 7.)

• **Papers under review:**

1. Tong, Hao, Leandro L. Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao. “Evaluating meta-heuristic algorithms for dynamic capacitated arc routing problems by deriving lower bounds.” In submission to IEEE Computation Intelligence Magazine, 2023. (This paper mainly addresses the contribution 5, and supports Chapter 6.)
2. Tong, Hao, Leandro L. Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao. “Building blocks as experiences in dynamic capacitated arc routing problems.” In submission to CAAI Transactions on Intelligence Technology, 2023. (This paper mainly addresses the contribution 4, and supports Chapter 5.)

1.4 Thesis Outline

The structure of this thesis is organized as follows. Chapter 2 first presents a comprehensive mathematical formulation of the DCARP, and introduces a simulation and benchmarking system for DCARP which offers a novel research platform for the DCARP research. Following that, Chapter 2 provides a literature review about the heuristic and meta-heuristic

algorithms for DCARP over the past decades. Based on the research gap of the existing literature, Chapter 3 proposes a novel generalized optimization framework to solve DCARP instances which is capable of leveraging the wealth of existing algorithms for static CARP, answering RQ1. Then, to further analyze which factors make DCARP hard to solve for algorithm enhancement, a fitness landscape analysis is performed and a series of key insights are obtained in Chapter 4, answering RQ2. Building upon these insights from fitness landscape analysis, the characteristics of different DCARP scenarios are considered specifically in Chapter 5. Thus, an experience-assisted dynamic optimization framework for different DCARP scenarios by utilizing the solution building blocks as the experience from the optimization results of the former DCARP instance in Chapter 5, answering RQ3. After that, the performance of our proposed algorithms are evaluated based on the derived node matching lower bound in Chapter 6 and subsequently evaluated through the designed DCARP scenario benchmarking system containing real-world traffic data in Chapter 7, for answering RQ4 and RQ5, respectively. Finally, Chapter 8 summarizes this thesis and highlights the potential future works.

Chapter Two

Background

Sections 2.1 and 2.3 are based on the following publication: *Tong, Hao, Leandro Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao. “A novel generalized meta-heuristic framework for dynamic capacitated arc routing problems.” IEEE Transactions on Evolutionary Computation, vol. 26, no. 6, pp. 1486-1500, Dec. 2022. <https://doi.org/10.1109/tevc.2022.3147509>.*

In practical applications, many logistic problems can be abstracted into the CARP, such as the mail delivery, the waste collection and salt gritting. However, these applications exhibit varying characteristics. For example, in the mail delivery problem, each parcel has a unique destination and must be delivered to the specific customer. In such cases, each vehicle has to complete all initially assigned tasks by itself even though the dynamic events occur to influence the original schedule. Additionally, when new tasks/demands emerge, these new tasks/demands must either be accommodated by vehicles currently in service after they return to the depot or assigned to new vehicles from the depot. For simplicity, this thesis will not consider such scenarios. Instead, we consider the collection or gritting application, where tasks originally assigned to one vehicle can also be reassigned to another regardless of any dynamic events, and vehicles can continue the service paths without requiring to return to the depot when new tasks/demands appear. Our thesis primarily focuses on the corresponding DCARP scenarios for such collection or gritting applications.

For the considered DCARP scenario, in this chapter, we will provide the mathematical formulation first in Section 2.1 since a formal mathematical formulation of DCARP is still missing in the literature. Then, the systematical literature review of solution representation and optimization algorithms for both CARP and DCARP will be presented in Section 2.2. Section 2.2 will also provide a brief overview of dynamic optimization in the literature. After that, a new simulation and benchmarking system for our DCARP research are introduced in Section 2.3. Finally, Section 2.4 summarizes this chapter.

2.1 Mathematical Formulation

A DCARP scenario is composed of a series of DCARP instances: $\mathcal{I} = \{I_0, I_1, \dots, I_t, \dots, I_M\}$. Each DCARP instance corresponds to a problem state, which contains all the information

regarding the state of the map and vehicles involved in the routing problem, and highly depends on the previous instance and the execution of solution. The initial problem instance I_0 is a conventional static CARP, in which all vehicles are located at the depot having the same full capacities. We can obtain an initial solution in I_0 and execute this solution in the graph. During the execution, some dynamics¹ [60], such as road congestion or new tasks, happen at random points in time when vehicles are in service, thus changing the problem instance and potentially requiring a new better solution. Vehicles then continue to serve tasks from the positions they had stopped (stop points). DCARP terminates when all tasks are served, and all vehicles have returned to the depot. In a DCARP scenario, the key objective is to achieve a schedule cost, which should be as low as possible for each individual DCARP instance. All symbols used for the mathematical formulation in this section are in the *List of Symbols* at the beginning of this thesis.

The map for any DCARP instance I_t is provided as a graph G . Suppose the map of a DCARP instance I_t is represented by $G = (V, A)$ with a set of vertices V and arcs (directed links) A . There is a depot $v_0 \in V$ in the graph, which contains vehicles that are not yet serving any tasks. The set A is given by

$$A = \{u = \langle v_i, v_j \rangle | v_i, v_j \in V\}$$

where for each arc u , i.e. $\langle v_i, v_j \rangle \in A$, v_i is the head vertex and v_j is the tail vertex. A given arc $\langle v_i, v_j \rangle$ only exists if it is possible to traverse from vertex v_i to vertex v_j without passing through other vertices. Each arc u in the graph is associated with a deadheading (traversing) cost $dc(u)$, a serving cost $sc(u)$ and a demand $dm(u)$. The deadheading cost of an arc means the cost that the vehicle just traverse this arc without serving while the serving cost is the cost when vehicles serve this arc. For simplicity, the deadheading cost are assumed to be symmetric in this thesis. The deadheading cost has been included in the

¹The full list of dynamic events considered in our work are presented in Table 2.1.

serving cost such that the deadheading cost is not required to be calculated when the vehicle serves an arc. A subset $E_R \subseteq A$ contains all arcs required to be served in the graph. Each arc $u \in E_R$ is named as ‘task’ and has a positive demand $dm(u) > 0$. For convenience, we use t to represent a task, and use an arc ID for identification. For example, t_1 represents the task, with its id being ‘1’.

The DCARP instance I_0 only contains vehicles at the depot. As for DCARP instances $I_t(t > 0)$, in addition to vehicles that are currently at the depot, there may also be outside vehicles with remaining capacities. These are vehicles that had already started to serve tasks when a dynamic event occurs. Suppose there are N_{veh} vehicles in total with a maximum capacity Q at the depot and N_{ov} ($N_{ov} \leq N_{veh}$) outside vehicles with remaining capacities $RQ = \{q_1, q_2, \dots, q_{N_{ov}}\}$. The stop points (locations) of the outside vehicles are labelled as $V_{ov} = \{sp_1, sp_2, \dots, sp_{N_{ov}}\}$. In practical scenarios, vehicles often locate in the middle of a road when dynamic events occur. These stop locations can be regarded as new vertices of the map to aid in optimization. In this thesis, for simplicity, we assume that the vehicle stops at the nearest next existing vertex on the map when dynamic events occur. The optimization algorithms we propose under this assumption are also suitable for real-world situations where vehicles’ stop locations are not align precisely with a map vertex. The optimization of DCARP aims to reschedule the remaining tasks with the minimal cost considering both outside and depot vehicles.

A DCARP solution $S = \{r_1, r_2, \dots, r_{N_{ov}}, \dots, r_K\}$ contains K routes, where the routes r_1 to $r_{N_{ov}}$ start from locations that outside vehicles located while routes $r_{N_{ov}+1}$ to r_K start from the depot. Each route can be represented by three components: starting vertex, an ordered list of tasks (arc IDs) and the final depot. Therefore, a given route r_k can be expressed as $r_k = (sp_k, t_{k,1}, t_{k,2}, \dots, t_{k,l_k}, v_0)$, where the vehicle starts from stop location sp_k and returns to the depot v_0 , whereas l_k denotes the number of tasks served by route r_k . For route r_k , where $k > N_{ov}$, sp_k equals to v_0 . This representation is very easy to be converted to an explicit

route by connecting two subsequent tasks using Dijkstra's algorithm so that the route cost can be calculated. In addition, a DCARP solution has to satisfy three constraints which are the same as constraints in the static CARP:

- Each route served by one vehicle must return to the depot.
- Each task has to be served once.
- The total demand for each route served by one vehicle cannot exceed the vehicle's capacity Q .

Due to the different remaining capacities for outside vehicles, the capacity constraint is required to be formulated for each outside vehicle separately. As a result, the objective function and the constraints for a DCARP instance are given as follows:

$$\mathbf{Min} \quad TC(S) = \sum_{k=1}^K RC_{r_k} \quad (2.1)$$

which subjects to the following constraints:

$$\mathbf{s.t.} \quad \sum_{k=1}^K l_k = N_T \quad (2.2a)$$

$$t_{k_1, i_1} \neq t_{k_2, i_2}, \text{ for all } (k_1, i_1) \neq (k_2, i_2) \quad (2.2b)$$

$$\sum_{i=1}^{l_k} dm(t_{k,i}) \leq q_k, \forall k \in \{1, 2, \dots, N_{ov}\} \quad (2.2c)$$

$$\sum_{i=1}^{l_k} dm(t_{k,i}) \leq Q, \forall k \in \{N_{ov} + 1, \dots, K\} \quad (2.2d)$$

where RC_{r_k} is the total cost of route r_k which is calculated by Eq. (2.3):

$$RC_{r_k} = mdc(sp_k, head_{t_{k,1}}) + mdc(tail_{t_{k,l_k}}, v_0) + \sum_{i=1}^{l_k-1} mdc(head_{t_{k,i}}, tail_{t_{k,i+1}}) + \sum_{i=1}^{l_k} sc(t_{k,i}) \quad (2.3)$$

where the $head_{t_{k,i}}$, $tail_{t_{k,i}}$ denote the beginning and ending vertices for serving the tasks. The minimal total traversing costs to traverse from node v_i to node v_j is denoted as $mdc(v_i, v_j)$, and $sc(t_{k,i})$ denotes the serving cost of the task $t_{k,i}$. The constraint (2.2a) guarantees that the number of tasks in the solution equals to the number of tasks of the instance. The constraint (2.2b) guarantees that any two tasks in the solution are different, i.e., the solution does not attempt to serve the same task more than once. Therefore, these two constraints guarantee that all tasks are served once and only once. Then, constraints (2.2c) and (2.2d) are formulated to satisfy the capacity constraint of outside vehicles and new vehicles, respectively. The capacity constraints for outside vehicles are the main difference between the conventional static CARP and the DCARP instance, requiring the algorithm to consider outside vehicles and new vehicles from the depot separately during the optimization process.

2.2 Literature Review

2.2.1 Solution Representation for CARP

The CARP aims to assign vehicles to serve a set of tasks. In order to obtain a feasible solution through algorithms for the given problem, the solution representation is the most fundamental issue requiring to be addressed first.

There are two representations commonly used in optimization algorithms for CARP in the

literature. The first type provides all explicit routes in the solution, separated by a dummy task [113], while the other type is an ordered list of tasks without separation. Suppose there are 10 tasks in an instance, two solutions represented by explicit-route representation and ordered-task-sequence representation, respectively, are presented as follows:

(a). Explicit-route representation: $\{0, t_1, t_2, t_3, 0, t_4, t_5, t_6, 0, t_7, t_8, t_9, 0, t_{10}, 0\}$

(b). Ordered-task-sequence representation: $\{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}\}$

Both solution representation types are effective for finding better solutions within the optimization algorithm. However, only the explicit-route represented solution can be directly deployed to vehicles in practice thereby enabling the calculation of the total costs for the solution. This is not applicable for the ordered-task-sequence represented solution, which does not explicitly show which tasks should be assigned to different vehicles. Nevertheless, the solution with ordered-task-sequence representation can be optimally converted into an explicit-route represented solution through the Ulusoy's split scheme [120], which builds an auxiliary graph according to the task sequences and splits the ordered task sequence to explicit routes. Consequently, solutions represented as ordered task sequence can be evaluated and deployed.

Almost all optimization algorithms for CARP are designed based on these two solution representation types. They are also employed to optimize the DCARP in this thesis.

2.2.2 Optimization Algorithms for CARP

Over the past decades, there have been a lot of research dedicated to solving the CARP, resulting in plenty of effective algorithms. Generally, these algorithms can be categorized into *constructive heuristic algorithms* and *meta-heuristic algorithms*.

2.2.2.1 Constructive Heuristic Algorithms

Constructive heuristic algorithms have attracted much attention since the very early stage. They are aimed at directly constructing effective, high-quality executable solutions for a given CARP instance from scratch. The path-scanning [36], augment-merge [36], and Ulusoy's split scheme [120] are three most classical and popular constructive heuristic algorithms in the literature. Their details are introduced as follows:

- The *path-scanning algorithm* constructs the route starting from the depot, and then, incrementally adds the nearest unserved, feasible task to the current route one by one. In cases where multiple nearest unserved tasks are available simultaneously, the algorithm employs five different tie-breaking rules to select the most promising task as the next task for the current route. Each route will complete when no additional task can be feasibly added due to the vehicle capacity constraint, and the route concludes with a depot. Then, if there are still remaining unserved tasks, a new route starting from the depot will be generated to continue the process. Finally, the algorithm terminates when all tasks are served, and then outputs an explicit-route represented solution.
- The *augment-merge algorithm* includes two different phases. In the augment phase, a complete route is constructed for each individual task. Each route starts from the depot and only serves one task before returning to the depot. Then, in the merge phase, these routes are merged to reduce total costs. For each pair of routes, the algorithm executes the merge with the largest cost saving among four possible merging strategies. Finally, the algorithm terminates when no further merging can be executed.
- The *Ulusoy's split scheme* is used to convert an ordered-task-sequence represented solution to a feasible executable explicit-route represented solution, splitting a giant tour into several capacity-feasible routes. Given an ordered-task-sequence represented solution, Ulusoy's split scheme builds an auxiliary graph based on the given solution. In the auxiliary

graph, the first node represents the depot, and each of the remaining nodes represents a task in the given solution. Each edge in the auxiliary graph represents a feasible sub-route of the CARP, and its cost is the weight of the corresponding edge. Then, Ulusoy's split uses Dijkstra's algorithm to find the shortest path from first node to the last node in the auxiliary graph, which indicates the optimal split for the assigned ordered-task-sequence represented solution.

On the basis of three basic constructive heuristic algorithms, various effective extensions were proposed recently. For example, Pearn [95] proposed a random path-scanning, which randomly chooses one tie-breaking rule to select the task for the current route rather than constructing a solution for each rule. Wøhlk [130] introduced a modified path-scanning, wherein the candidate tasks are not only limited to the adjacent edges of the current node, as in the original path-scanning [36]. Instead, all feasible tasks across the whole graph are applicable. This modified version of the path-scanning is more widely utilized than the original version currently. Santos et al. [107] proposed an ellipse rule-based path-scanning (PS-Ellipse), wherein they introduced an ellipse rule to restrict the selection of the next task within a specific ellipse close to the depot, when the loads of the vehicle is nearing the capacity. Arakaki and Usberti [1] further extended the PS-Ellipse by dynamically activating the rule and proposing a new PS-Efficiency algorithm with a new efficiency rule considering both the demand and the traversing costs of candidate tasks when choosing the next task of the route. Belenguer et al. [8] eliminated the augment phase in augment-merge algorithm resulting a merge algorithm, and the algorithm was also further improved by incorporating a rule that prefers the merge with the largest discrepancy in load when the maximal cost savings are tied. Prins et al. [100] extended Ulusoy's splitting scheme by improving the giant tour construction and the splitting process, respectively. Eventually, they proposed three methods for generating giant tour and four splitting approaches, resulting in 12 different splitting heuristics [100].

In the literature, besides improvements on basic algorithms, a few innovative constructive heuristics have been proposed. Wöhlk [130] proposed three new heuristics including Node Duplication Heuristic, Double Outer Scan, and A-ALG. The Node Duplication Heuristic constructs several Euler Tours in the graph assisted with the node duplicated network and subsequently splits these Euler Tours to capacity-feasible routes. The Double Outer Scan incorporates the ideas of both path-scanning and augment-merge algorithms that generates multiple tours in a path-scanning way and then merges pairs of tours to construct a feasible solution. Moreover, A-ALG [130] is also a splitting heuristic to split an RPP tour into a feasible CARP solution. It is similar to the heuristic named short optimal tour partitioning (SOTP) proposed by Jansen [48]. SOTP initially applies Frederickson's heuristic [34] to the RPP to create a giant tour, and then, employs the Ulusoy's split scheme to get a feasible CARP solution.

Constructive heuristics have been identified as an effective strategy for uncertain CARP in the literature. Instead of optimizing a robust solution, researchers focus on devising robust constructive heuristics for uncertain environments. The service schedules of vehicles can be updated according to the obtained constructive heuristic in response to the changing environment. Weise et al. [129] evolved robust constructive heuristics for uncertain CARP through symbolic regression using genetic programming (GP), employing trees to represent the heuristics. When constructing the solution, the heuristic function calculates a heuristic value for each candidate task and determines which task is to be served next until a complete feasible solution is constructed. In scenarios of uncertain CARP, the robust constructive heuristic can be used all the time during vehicle service to deal with the uncertain environment. Liu et al. [64] extended this groundwork by adapting the heuristic to be capable of tackling stochastic demands. Furthermore, to simplify the complexity of evolved heuristics, Wang et al. [125] incorporated a niching-based strategy to GP, which was demonstrated successful in yielding more compact yet effective heuristics. Ardeh et al. [3]

leveraged the knowledge, which consists of all evolved heuristics from related source problem instances, to assist in obtaining much superior heuristics for the target problem instance.

Even though constructive heuristic algorithms can directly construct an executable solution for a CARP instance, the quality of the generated solution is not guaranteed and often still required to be improved. To address this, the meta-heuristic algorithms are developed over the past two decades. In the next subsection, we will introduce meta-heuristic algorithms in details.

2.2.2.2 Meta-heuristic Optimization Algorithms

Unlike the constructive heuristic which only generates one single solution, meta-heuristic algorithms produce new solutions based on the existing ones and iteratively refine them to find much better solutions. For example, one of the earliest meta-heuristic algorithm, as far as we know, is the Simulated Annealing algorithm proposed by Eglese [26], wherein a solution is initially obtained by a “greedy” constructive heuristic, and subsequently, further improved through Simulated Annealing. Since then, many effective meta-heuristic algorithms have been proposed for optimizing the CARP. For example, the Tabu Search [14, 46, 78], Memetic Algorithms [52, 113], Genetic Algorithms [71], etc.

Within the meta-heuristic algorithms, solutions initialized by constructive heuristics are improved by various different search operators. In the literature, the search operators for refining the solution includes *local search* and *global search* operators.

- The *local search* is the most effective operators for exploiting the current solution and usually only applicable to an explicit-route represented solution. Neighborhood moves are the main components in the local search operators for CARP. The CARP solution is usually encoded as several explicit routes, each of which is a task sequence. The sequence

of tasks determines the order in which tasks are served and directly influences the total cost of a solution. The neighborhood moves switch tasks or part of the route in the same route or between different routes to search a better solution. Five most commonly used neighbourhood moves in the literature are described as follows, where the $\mathbf{0}$ is a dummy task to separate routes, t_i represents a single task, and $inv(t_i)$ represents the task t_i with inverse direction:

1. **Single Insertion:** One task is removed from one route and inserted into another position of the current or another route.

$$\{\mathbf{0}, t_1, \underline{t_2}, t_3, t_4, t_5, \mathbf{0}, t_6, t_7, t_8, t_9, \mathbf{0}\} \rightarrow \{\mathbf{0}, t_1, t_3, t_4, t_5, \mathbf{0}, t_6, t_7, \underline{t_2}, t_8, t_9, \mathbf{0}\}$$

2. **Double Insertion:** Two consecutive tasks are removed from one route and inserted into the current or another route.

$$\{\mathbf{0}, t_1, \underline{t_2}, \underline{t_3}, t_4, t_5, \mathbf{0}, t_6, t_7, t_8, t_9, \mathbf{0}\} \rightarrow \{\mathbf{0}, t_1, t_4, t_5, \mathbf{0}, t_6, t_7, \underline{t_2}, \underline{t_3}, t_8, t_9, \mathbf{0}\}$$

3. **Swap:** Two tasks belonging to the same route or different route are swapped.

$$\{\mathbf{0}, t_1, \underline{t_2}, t_3, t_4, t_5, \mathbf{0}, t_6, \underline{t_7}, t_8, t_9, \mathbf{0}\} \rightarrow \{\mathbf{0}, t_1, \underline{t_7}, t_3, t_4, t_5, \mathbf{0}, t_6, \underline{t_2}, t_8, t_9, \mathbf{0}\}$$

4. **Reverse:** A subsequence of a route is flipped, and the direction of each task is also reversed.

$$\{\mathbf{0}, t_1, \underline{t_2}, \underline{t_3}, \underline{t_4}, t_5, \mathbf{0}, t_6, t_7, t_8, t_9, \mathbf{0}\} \rightarrow \{\mathbf{0}, t_1, \underline{inv(t_4)}, \underline{inv(t_3)}, \underline{inv(t_2)}, t_5, \mathbf{0}, t_6, t_7, t_8, t_9, \mathbf{0}\}$$

5. **Cross:** Swap the end portion of two different routes.

$$\{\mathbf{0}, t_1, t_2, \underline{t_3}, \underline{t_4}, \underline{t_5}, \mathbf{0}, t_6, t_7, \underline{t_8}, \underline{t_9}, \mathbf{0}\} \rightarrow \{\mathbf{0}, t_1, t_2, \underline{t_8}, \underline{t_9}, \mathbf{0}, t_6, t_7, \underline{t_3}, \underline{t_4}, \underline{t_5}, \mathbf{0}\}$$

Moreover, the local search usually assesses all possible neighborhood moves for a given solution, thereby making the exploitation much more effective.

- The *global search* operators are always utilized to explore the current solution to increase the diversity of the search process. It is common to perform crossover on a pair of ordered-tasks-sequence solutions to obtain new solutions. The crossover operator often randomly splits each ordered-tasks-sequence solution into several parts, and merges new solutions using sub-routes from different ancestral solutions [71]. When designing the crossover operator, it is most important to avoid repetition and missing tasks in the offspring solution since each task should be served only once in the CARP. Three commonly used crossover operators for ordered-tasks-sequence solution are described as follows:

1. **Order Crossover (OX)** [52]: Each parent solution is divided into three segments in the same positions. The middle segment of the first parent is directly copied to the corresponding positions of the offspring, and these tasks are deleted from the second parent. Then, the remaining tasks from the second parent are inserted into the offspring, adhering their left-to-right sequence in the second parent.
2. **Random Key-based One Point Crossover (RK-OPX)** [60, 71]: A random key vector, which is a sequence of real numbers randomly sampled from interval $[0, 1)$ and can be decoded into a sequence of tasks, is generated to represent each parent solution. Then, a one point crossover is applied to two random key vectors to produce a new offspring key vector. The resulting key vector is then decoded to an ordered-task-sequence solution. In the one point crossover operator, each individual is divided into two segments, and the offspring individual is generated by inheriting one segment from each of the parents.
3. **Sequence-based Crossover (SBX)** [113]: Even though this operator is applied to an explicit-routed solution, for each parent solution, a single route is randomly selected from each parent solution for the crossover and the selected route is also

represented as an ordered-task-sequence. Each selected route is divided into two segments, and an offspring route is formed by combining the first segment from one parent route with the second segment from the other parent route. In the resulting offspring solution, duplicated tasks are removed, and each missing task is greedily inserted with the lowest additional cost.

Generally, the resulting ordered-task-sequence represented solution from the crossover will be split into an explicit-route solution to calculate its objective value. Typically, it is then provided to the local search operators for further improvements.

Overall, meta-heuristic algorithms in the literature can be classified into two categories: *individual-based algorithms* and *population-based algorithms*.

Generally, individual-based algorithms only work with one single solution initialized by the constructive heuristic algorithm, and iteratively improve it through local search operators. For example, Eglese [26] proposed the Simulated Annealing algorithm to refine the initially constructed solution in a multi-depot gridding problem. Hertz et al. [46] developed a tabu search heuristic called CARPET for the CARP, where a tabu list is used to store the movement of an edge between routes to avoid the reversal of the moves. Brandão and Eglese [14] proposed a deterministic tabu search algorithm (TSA) in which the tabu list is also used to prevent repetitive moves. In addition, they employed Frederickson's heuristic to refine the route, and thereby further improve the solution. Mei et al. [78] proposed a global repair operator to repair the infeasible solution within the optimization of the TSA, resulting in a new algorithm RTS. Besides, Hertz and Mittaz [47] proposed a variable neighborhood descent algorithm, which performs local search iteratively until no further better solutions can be found in the neighborhoods. Polacek et al. [98] proposed a variable neighborhood search which employs a shaking operation to generate a random solution for the subsequent local search at each step and the algorithm continues until the maximum steps is reached. Beullens et al. [11] proposed a guided local search, wherein the neighborhood moves are

guided though the neighbor lists and edge marking strategies during the local search to enhance computational efficiency.

In contrast to individual-based algorithms, the population-based algorithm works on a set of solutions. These solutions are initialized through both constructive heuristics and the splitting scheme on a set of randomly generated ordered-task-sequence solutions. It is generally insufficient to populate the initial solution set only using constructive heuristics since each constructive heuristic only produces one solution. On the other hand, solely relying on the random generation is likely to generate only poor-quality solutions, potentially causing an inefficient optimization. After the initialization, the solutions are subsequently iteratively evolved with both local search and global search operators. As a result, the inherent diversity of searching makes the population-based meta-heuristic algorithms more effective and robust than the individual-based meta-heuristic algorithms, and thereby more prevalent in the recent literature.

A plenty of population-based meta-heuristic algorithms have been proposed in the literature, among which memetic algorithms stand out as the particularly effective algorithm for solving CARP and its variants. Lacomme et al. [52] first applied the memetic algorithm to solve an extended CARP and proposed a Lacomme's memetic algorithm (LMA). It randomly selects two parent solutions for the ordered crossover operator to produce an offspring solution, which is then improved by the local search using five neighborhood moves based on a given probability. In addition, LMA uses a partial replacement procedure for the restart after a predefined number of productive crossovers to increase the search diversity. However, the ordered-tasks-sequences representation is used throughout the whole algorithm and the offspring after local search has to be converted to the explicit routes for fitness evaluation with the splitting scheme. The violation of the capacity constraint is not considered and the switch between ordered task sequence and explicit routes potentially degrades the quality of resultant offspring.

To overcome this weakness, Mei et al. [82] proposed an improved LMA (ILMA), which utilizes the explicit-route representation. Solutions in ILMA are converted to an ordered-task-sequence for the crossover and split after the crossover for subsequent local search. ILMA allows the violation of the capacity constraints, handled with the penalty strategies during the optimization. As a result, the performance of the ILMA is significantly better than the original LMA. Tang et al. [113] introduced a novel local search operator, known as Merge-Split, with a large step when performing the local search. They proposed a new memetic algorithm with extended neighborhood search (MAENS). The empirical studies demonstrate MAENS performs significantly better than all other state-of-the-art optimization algorithms [113].

Moreover, Feng et al. [31] proposed a formal probabilistic memetic algorithm in which the intensity of a solution undergoing the local search is determined by an upper bound of the local search intensity. Liu et al. [62] integrated an iterated local search into their memetic algorithm framework to further improve the quality of the solution from the local search when it is close enough to the best parent solution. They also proposed a longest common substring crossover employed in conjunction with the order crossover to trade off between the exploration and exploitation. Liu et al. [61] developed a similarity based parent selection scheme into the memetic algorithm framework, incorporating a hybrid recombination scheme crossover operation and the extended neighborhood search. Wang et al. [127] proposed a rank-based memetic algorithm in which a rank-based neighborhood search operator was designed to further improve the solution after the basic local search operators. The proposed operator could quantify the efficacy of a neighborhood move based on the rank information of the edges before its execution, and the move yielding a positive gain would finally be executed, directing a more effective and promising local search. Recently, Li et al. [57] proposed a memetic algorithm with non-smooth penalty, in which the concept of smooth degree was introduced to characterize a CARP solution. The smooth degree correlates positively with

the quality of solution. Therefore, they proposed partial reconstruction methods with non-smooth penalty and re-insertion methods to reduce the smooth degree of the solution for improving the quality of solution along with traditional local search operators.

Besides, there are also various population-based meta-heuristic algorithms in the literature. For example, Lacomme et al. [53] first introduced the genetic algorithm to solve the CARP. The ordered crossover operators were applied to parents selected by binary tournament for a new solution, which was then further improved through several neighborhood moves based on a given probability. Martinez et al. [71] introduced a biased random key genetic algorithm to optimize the CARP, in which solutions are encoded as random key vectors facilitating the use of operators within the genetic algorithm for generating new solutions. The local search operators and the restart operation were also applied to further improve the solution. Greistorfer [39] incorporated a tabu search procedure into the scatter search framework to solve the CPP, which combined elite solutions from tabu search procedure to produce new solutions to increase the search diversity. Chu et al. [18] and López-Santana et al. [66] also applied scatter search algorithm to optimize the periodic CARP and CARP with refill points, respectively. Furthermore, the ant colony optimization (ACO) algorithm is also effective for CARP. Lacomme et al. [55] applied the first ant colony scheme to solve the CARP, in which both elitist and non-elitist ants respectively controls the convergence and the diversity of the search. Local search operations were also applied to enhance the quality of solutions. The ACO framework was even applied to solve a real-world urban waste collection problem modeled as a mix CARP [6]. Santos et al. [106] introduced an improved ant colony meta-heuristic with more effective population initialization procedure, ant decision rules and local search operations, resulting significant performance improvements compared to state-of-the-art algorithms. Xing et al. [134] also proposed a hybrid ACO algorithm to tackle an extended CARP considering both service costs and fixed investment costs. Additionally, Shang et al. [109] proposed an immune clonal selection algorithm for

CARP with a variety of different strategies to discover high-quality solutions. Chen et al. [17] proposed a hybrid meta-heuristic approach with a new randomized tabu thresholding procedure. Its efficacy is enhanced through the use of a route-based crossover operator for producing new promising solutions and a quality-and-distance based pool updating strategy to escape the local optimum. Nagy et al. [88] adapted the artificial bee colony algorithm with a new sub-route plan operator to optimize the CARP instance.

Although it has already so many powerful algorithms proposed for solving CARP, they are all not applicable for DCARP due to the calculation of cost and capacity violation of the routes corresponding to the outside vehicles which have different locations and remaining capacities. It is more complicated to use an ordered list of tasks as the solution representation during the optimization because the Ulusoy's split [120] is not suitable anymore and specific split schemes [119] are required. Even though a new split scheme has been proposed [119], its high computational complexity limits its performance. Therefore, the existing algorithm for static CARP also needs to be adapted to solve DCARP instance.

2.2.3 Optimization Algorithms for DCARP

Even though the CARP has been extensively investigated in the literature with a plenty of effective algorithms, the studies focusing on DCARP remain comparatively limited so far, to the best of our knowledge. Dynamic CARP optimization focuses on the rescheduling of solutions that potentially become inferior when dynamic events occur. Current approaches for solution updates mostly involve re-optimizing the newly generated DCARP instance from scratch.

The DCARP was first investigated when considering the salting route optimization problem and solved by a new memetic algorithm [43]. A repair operator was incorporated to fix

infeasible solutions produced by an adapted edge assembly crossover, and three neighborhood moves were adopted within the algorithm. Tagmouti et al. [112] considered DCARP with time-dependent service costs motivated from winter gritting application, in which the dynamic events are known prior to solution deployment, and optimized with a proposed variable neighborhood descent heuristic. Liu et al. [59] designed a benchmark generator for DCARP considering a variety of general dynamic events including availability of vehicles, accessibility of roads, addition/deletion of tasks. Since then, the investigation and optimization for DCARP attracted increasing attentions.

Liu et al. [60] proposed a memetic algorithm with new split scheme for DCARP (MASDC). Since additional constraints of outside vehicles and different remaining capacities in the DCARP instance cause the Ulusoy's splitting scheme not to be applicable anymore, the proposed new distance-based splitting scheme converts ordered-tasks-sequence solutions to explicit-route solutions for fitness evaluation and local search operations. MASDC initialized a population of ordered-tasks-sequence solutions, which were evaluated through the distance-based splitting scheme. Subsequently, the memetic algorithm, consisting of a random key crossover and three local search operators including single insertion, double insertion and swap, was introduced to re-optimize the new DCARP instance. As the first algorithm capable of solving general DCARP instances, experimental results demonstrate that MASDC is effective for DCARP optimization.

Monroy-Licht et al. [84] considered disruption costs for optimizing the DCARP instance with broken-down vehicles. They reduced the size of the problem by fixing some variables and rescheduling only part of the vehicles in the original solution. The remaining tasks were divided into two sets: one consists of unserved tasks belonging to active vehicles, and another comprising tasks from failed vehicles. Subsequently, tasks from failed vehicles were reassigned to the closest active vehicle's task. Routes for those active vehicles that received new tasks initially assigned to failed vehicles were rescheduled, and remaining active vehicles

remained fixed.

Padungwech et al. [94] focused only on new task dynamic events and employed a random insertion operator to handle them. Initially, new tasks were arranged in a random order, and then they are inserted into the given solution using a greedy method. This method inserted each task into a feasible position, satisfying capacity constraints for all routes, within the solution that resulted in the smallest increase in costs. Following the random insertion process, Tabu Search was employed to improve the newly generated solution. Experimental studies showed that random insertion is particularly suitable for scenarios requiring more frequent update schedules.

Nagy et al. [87] proposed a novel re-routing algorithm handling new task dynamic events, demand increased events, and vehicle breakdown dynamic events. The added new tasks, tasks with increased demands that cause routes to exceed vehicle capacity, and tasks from failed vehicles are removed from the original solution to construct a new solution set. The proposed re-routing algorithm checked whether these new tasks can be inserted into one available route without violating capacity constraints. If the insertion is feasible, these tasks are greedily added to the route that results in the smallest increase in costs. Otherwise, these new tasks formed a new route which is then assigned to a new vehicle. Recently, they also proposed an artificial bee colony algorithm for optimizing static CARP and was also embedded into a DCARP framework² to optimize DCARP instances [88].

Additionally, it is worth noting that although VRP focuses on serving vertices instead of tasks/arcs, the research methodologies on solving dynamic VRP (DVRP) are also applicable for DCARP after transforming DCARP instances into DVRP instances. Nevertheless, such a transformation will increase the problem's dimension [33, 96]. For example, the number of vertices in the CVRP instance will increase if transformed from a CARP instance. The

²This DCARP framework is our proposed framework in Chapter 3.

number of vertices to be served will be greater than the number of arcs to be served. Furthermore, dynamic events in VRP and CARP are very different. In short, it is not a suitable approach to convert CARP into capacitated VRP (CVRP) and then solve CVRP. It is better to design CARP or DCARP specific algorithms as the research community has been doing for many years.

As reviewed above, existing studies focusing on DCARP in the literature are still limited. Most works only concentrated on one specific dynamic event. For example, Monroy-Licht et al. [84] only considered broken-down vehicles, while Padungwech et al. [94] focused only on new tasks. However, real-world applications always encounter a variety of unexpected dynamic events due to uncertain factors from the transportation environment or the changes in tasks [58], all of which impact the quality of currently deployed solution. Therefore, in this thesis, we focused on a much more general DCARP rather than any specific one to a particular dynamic event. The considered general DCARP instance can be generated by any type of dynamic event, such as additional arc requests, changes to the demand, changes to the arcs' costs, and so on. These dynamic events only affect the values of the problem instance without altering the objective or constraint formulas. For example, cost-changing events only modify the costs of arcs in the problem instance, and adding demands events only adjust the demands of existing tasks, and new tasks events only impact the total number of tasks in the problem instance. Consequently, this thesis primarily investigates such general DCARP instance, in which dynamic events solely affect values of the problem instance.

On the other hand, most existing optimization algorithms for DCARP are specifically designed to deal with outside vehicles or specific dynamic events. For example, the new distance-based splitting scheme proposed by Liu et al. [60] was an operator to handle outside vehicles. However, the random splitting process in their algorithm resulted in poor optimization performance [60]. Padungwech et al. [94] and Nagy et al. [87] focused solely on operators for managing new tasks. Therefore, the current literature still lacks of a generalized

optimization framework benefiting from the powerful algorithms that have already been proposed to CARP to handle DCARP with much more general dynamic events. In addition, each DCARP instance is related to its preceding instance and the executed solution, which could be utilized to enhance DCARP optimization. Unfortunately, these existing works never considered and utilized such optimization experience, wasting valuable knowledge that could be used to enhance the optimization process for DCARP instances. Therefore, there is a need for both the generalized optimization framework and an experience-assisted dynamic optimization framework for efficient DCARP optimization.

2.2.4 Overview of Other Dynamic Optimization Studies

The dynamic optimization problems are prevalent in both continuous and combinatorial problem domains, and numerous effective algorithms have been developed to handle the dynamic changes in these problems. Generally, the primary target of dynamic optimization is to track the changing of the optimal solution in a dynamic environment. Several comprehensive surveys have systematically reviewed the existing dynamic strategies in the literature [72, 89, 136, 138]. Overall, these strategies can be mainly classified into two categories according to whether or not they benefit from the historical information of former environments.

When dynamic changes happen, algorithms that do not incorporate historical knowledge typically increase the diversity through randomizing the population [90] or changing hyperparameters in the algorithm [102]. Since successive environments usually share some similarities [15], historical information of the previous environment is likely to promote the optimization for the problem in the new environment. Thus, the second category of algorithms exploits the historical information to facilitate dynamic optimization. In the literature, multiple-population and memory schemes are the most typical strategies. Multiple-population strategy typically maintains multiple populations during the optimization proce-

ture and allocates them to different areas of search space to keep a higher search diversity for the new environment [12, 137]. The memory schemes usually save a set of promising solutions for the old environment, and then utilize them for a better initialization in the new environment [16]. Besides, the learning model-based dynamic strategies have recently become popular, which utilize machine learning algorithms to model the changing environment. For example, Wang et al. [126] introduced a regression transfer learning method to generate better initial solutions in the dynamic multi-objective optimization problem. Liu et al. [63] applied neural networks to construct a transfer model that learns environmental changes and the solutions for the old environment are adapted to the new environment through the trained model. Zhao et al. [139] employed a denoising auto-encoder to extract the valuable knowledge from the previous search processes in the dynamic multi-objective optimization. Most algorithms for dynamic optimization problems were specifically designed for continuous optimization problems or different combinatorial problems, and are not applicable for DCARP optimization.

In the literature, the Dynamic Vehicle Routing Problem (DVRP) also targets the rescheduling of vehicles in dynamic environments. Many algorithms and strategies have been proposed for solving the DVRP, mainly including swarm intelligence and evolutionary algorithms [72, 97, 103]. For example, Hanshar and Ombuki-Berman [45] applied the genetic algorithm to solve DVRP and Sabar et al. [104] proposed a self-adaptive evolutionary algorithm to tackle DVRP. It is worth noting that most works for DVRP focus on the Ant Colony Optimization (ACO) algorithm [25, 49]. The ACO is naturally suitable for tackling dynamic problems since the pheromone model implements the memory strategy and inherit the past experience [74]. Mavrovouniotis et al. [74] investigated the behaviors of ACO in dynamic travelling salesperson problem (DTSP) and concluding that the performance of ACO highly relies on the pheromone update policy. Moreover, Eyckelhof and Snoek [29] inherited the graph's pheromone matrix and adjusted the pheromones of a few edges to increase the exploration

ability for cost-changing dynamic events. Most works in the literature focus on new-task dynamic events. For settings of pheromone when adding new tasks in the DVRP, the restart strategy is the most intuitive approach, which assigns an initial pheromone value to the new task [85]. In addition, according to the position of new tasks, η -strategy and τ -strategy are proposed to adjust the pheromone values of new tasks [41]. Recently, there are also special learning-based strategies for handling new tasks. For example, Xiang et al. [132] proposed a pairwise proximity-based ACO for DVRP, which learns the pair preference of tasks and applies such pair preferences when constructing the new solution. The pheromone information in the ACO is used in the new instance, which retains most search experience of the previous optimization. However, these strategies are only suitable for ACO algorithms instead of general meta-heuristic optimization algorithms.

Dynamic CARP optimization focuses on the rescheduling of solutions that potentially become inferior when dynamic events occur. As reviewed above, current literature on dynamic optimization for DCARP mostly involves re-optimizing the new DCARP instance from scratch, except for some strategies tailored to very specific dynamic events. However, for DCARP scenarios with general dynamic events, the significant knowledge contained in the solutions or search process of the optimization for the former DCARP instance has not been effectively utilized. In the literature, knowledge transfer strategies have also been applied to assist in solving static CARP [30, 32] and uncertain CARP in the literature [2, 3]. Nevertheless, they are also not suitable for DCARP due to either their heavy computational burden for learning the transfer model or the exclusive applicability to tree-based representations. Therefore, in this thesis, we will focus on dynamic optimization for the DCARP scenario, aiming to propose an effective DCARP dynamic optimization framework, which is capable of extracting valuable experience from former DCARP instances and leveraging it to enhance the optimization for new DCARP instances within the same DCARP scenario.

2.3 Simulation and Benchmarking System

In order to test optimization algorithms for DCARP, a simulation system that includes some common dynamic events is required. Even though a benchmark generator for DCARP has been proposed by Liu et al. [59], it has shortcomings, which prevent it to be used as research platform. Intuitively, a DCARP instance should be generated from the dynamic change of a previous DCARP instance during a solution's execution, such as road congestion or recovering from the congestion. However, these essential details are not considered in the existing benchmark generator [59]. Therefore, we have designed a simulation system which includes nine commonly occurring events and generates DCARP instances from the existing CARP benchmark³. Nine events and their corresponding changes in mathematical forms are listed in Table 2.1, and the architecture of the simulation system is presented in Figure 2.1.

Table 2.1: Types of dynamic events in DCARP.

Event types	Changes
1. Vehicle break down	$dm(u_k) : 0 \rightarrow Q - q_k$
2. Road closure	$dc(u) : \overline{dc(u)} \rightarrow \infty$
3. Congestion	$dc(u) : \overline{dc(u)} \rightarrow \overline{dc(u)} + c$
4. Recover from roads closure *	$dc(u) : \infty \rightarrow \overline{dc(u)}$
5. Recover from congestion *	$dc(u) : dc'(u) \rightarrow \overline{dc(u)}$
6. Congestion become worse *	$dc(u) : dc'(u) \rightarrow dc'(u) + c$
7. Congestion become better *	$dc(u) : dc'(u) \rightarrow dc'(u) - c$
8. Demand increases	$dm(u) : dm(u) \rightarrow dm(u) + d$
9. Added tasks	$dm(u) : 0 \rightarrow d$

$\overline{dc(u)}$ is the expected deadheading cost of arc u without congestion. $dc'(u)$ is the deadheading cost of arc u with congestion. c, d are the changing cost and demand, respectively. Event 5 is a special case of Event 7 where $c = dc'(u) - \overline{dc(u)}$ in Event 7.

To make our simulator closer to realistic events, we have added several dynamic events, which have not been considered in the literature. For example, the road can recover from a

³<https://github.com/HawkTom/Dynamic-CARP>

closure or a congestion, which has been marked with a star (*) in Table 2.1. If a vehicle breaks down, we can assume that this vehicle k has already served some tasks. Consequently, the demand of the arc u_k , where the vehicle k broke down, increases from 0 to $Q - q_k$ to include the already served loads in collection applications. For delivery applications, the broken down vehicle has no impact to the demand of the arc. As we mainly considered collection or gridding problems, we assumed in our work that the tasks/demands will not vanish until fully served, although this may be extended and addressed in the future. However, our proposed simulator and framework is still applicable and capable of taking these events into account in case they would be added in future research. Based on Table 2.1, we can easily observe that all dynamic events impact the cost or demand of arcs. Therefore, we designed the *cost changer* and *demand changer* in our simulation system to simulate these events (Figure 2.1).

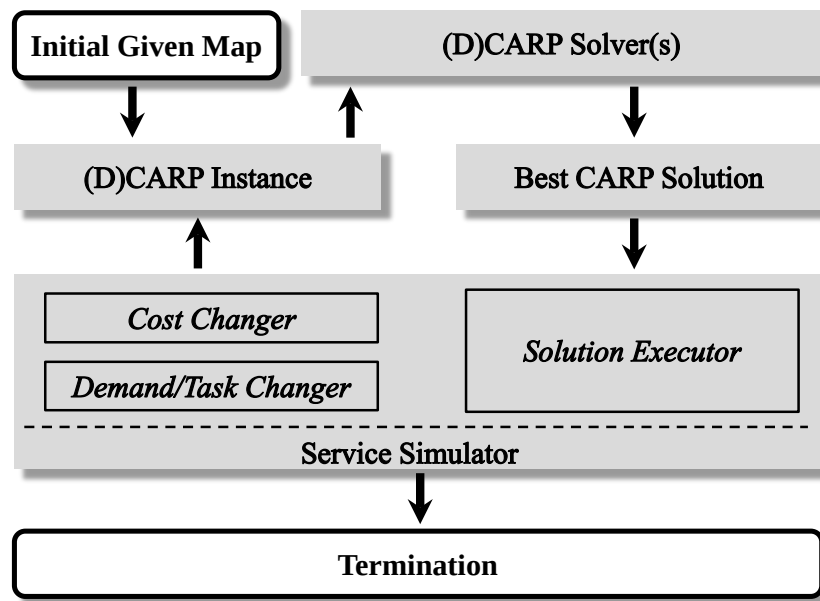


Figure 2.1: The architecture of our simulation system.

In Figure 2.1, the system starts from a provided initial graph, which could be taken from existing benchmarks for static CARP. Then, a CARP solver selected by the user, such as solvers based on memetic algorithms [113], can be used to obtain the initial solution, i.e. the initial schedule for the vehicles. The core part of the system is the *Service Simulator*

Algorithm 1: The pseudocode of the service simulator

Input: Executable solution S , Previous instance I_t

Output: The new instance I_{t+1}

```

1 Set probabilities of occurrence for all events:  $[p_{event}, p_{road}, p_{bdrv}, p_{crr}, p_{crbb}, p_{icd}, p_{add}]$ ;
2 Execute  $S$  on  $I_t$ ;
3 Select a uniformly random time to stop execution;
4 Remove all served tasks: make demand of all served tasks be 0;
5 Event 1 Randomly select  $n_{bdv}$  vehicles to break down.
6 /**** Cost Changer ****/
7 for each arc  $u_i \in A$  do
8     if  $rand() < p_{event}$  then
9         switch  $state(u_i)$  do
10             case 0 do
11                 if  $rand() < p_{road}$  then
12                     | Event 2  $dc(u_i) = \infty$ ,  $state(u_i) = 2$ 
13                 else
14                     | Event 3  $dc(u_i) = \overline{dc(u_i)} + c$ ,  $state(u_i) = 3$ 
15             case 2 do
16                 if  $rand() > p_{bdrv}$  then
17                     | Event 4  $dc(u_i) = \overline{dc(u_i)}$ ,  $state(u_i) = 0$ 
18             case 3 do
19                  $r = rand()$ 
20                 if  $r < p_{crr}$  then
21                     | Event 5  $dc(u_i) = \overline{dc(u_i)}$ ,  $state(u_i) = 0$ 
22                 else if  $r < p_{crbb}$  then
23                     | Event 6  $dc(u_i) = dc'(u_i) + c$ 
24                 else
25                     | Event 7  $dc(u_i) = dc'(u_i) - c$ 
26 /**** Demand Changer ****/
27 for each arc  $u_i \in A$  do
28     if  $dm(u_i) > 0$  and  $rand() < p_{icd}$  then
29         | Event 8  $dm(u_i) = dm(u_i) + d$ 
30     if  $dm(u_i) == 0$  and  $rand() < p_{add}$  then
31         | Event 9  $dm(u_i) = 0 + d$ 

```

in Figure 2.1, which is used to execute the CARP solution and update the graph. The pseudocode for the *Service Simulator* is presented in Algorithm 1. When the solution is executed, some dynamic events will happen according to a series of predefined parameters listed in Line 1 and influence the graph at a uniformly random time between the service start and completion time. After that, a new DCARP instance is generated based on the dispatched solution.

Once the dynamic change happens on the DCARP instance I_t , the *Service Simulator* stops the execution of the current solution. Then, the *cost changer* and *demand changer* will update the DCARP instance. First, as broken down vehicles influence only a specific arc, we simulate Event 1 separately from other dynamic events. The algorithm randomly selects n_{bdv} vehicles from all dispatched vehicles to break down, as shown in Line 5. Events 2 to 7 will influence the cost of several arcs so that the *cost changer* mainly simulates these five events, as shown in Lines 7-25. Each arc u_i has a traffic property, $state(u_i)$, recording whether it is currently in a changed state compared to its original state having expected cost without traffic events.

In the *cost changer*, the simulator firstly determines whether or not a change from the current state occurs according to the probability p_{event} for each arc. If a change occurs, a dynamic event is triggered according to the events' probabilities and arc's current changed state. If an arc keeps the original state, i.e. $state(u_i) == 0$, Event 2 or 3 happens in this arc depending on probability p_{road} . If $state(u_i) == 2$, the road has broken down before so that it recovers with a probability p_{bdr} . If $state(u_i) == 3$, the road is in congestion. It may either completely recover with a probability p_{crr} , or the traffic jam may ease or get worse (by a random cost) with a probability p_{crbb} or $1 - p_{crbb}$, respectively. Compared to Events 2 to 7, Events 8 and 9 in the *demand changer* are much easier to implement, because we assume that the tasks/demands do not vanish unless served completely and the demand of tasks can only increase under the practical scenarios considered in this thesis. Event 8 may

happen to a task with a probability p_{icd} , increasing the demand by a random amount. For arcs with no demand, Event 9 will happen with a probability p_{add} .

Finally, we will get a new DCARP instance I_{t+1} , and the solver generates a new DCARP solution. The system terminates after all tasks are served.

2.4 Summary

In this chapter, the first formal mathematical formulation for DCARP optimization is presented firstly. Then, we provided a detailed literature review about optimization algorithms on both basic CARP and DCARP, mainly including the constructive heuristic and meta-heuristics. Meta-heuristics are more prevalent in the literature, which employ various effective global search and local search operators to discover high-quality solutions in the search space. Nevertheless, current meta-heuristic algorithms for CARP are not capable of managing outside vehicles which locate at different points with various remaining capacities. The limited existing works on DCARP are also not effective for solving DCARP due to ineffective operations to handle these outside vehicles. Additionally, the existing work in the context of DCARP only addressed few specific dynamic events and does not cover more general dynamic events. Moreover, existing works for DCARP optimization only focused on re-optimization for new DCARP instances, without extracting potentially valuable knowledge from past optimization experience to promote dynamic optimization for DCARP scenarios. Hence, in this thesis, we mainly aim at developing effective meta-heuristic optimization algorithms for DCARP with general dynamic events, and designing strategies based on optimization experience to enhance dynamic optimization for DCARP scenarios. Finally, in this chapter, we also introduce a new simulation system facilitating the research on the DCARP.

Chapter Three

A Novel Generalized Framework for DCARP

This chapter is based on the following publication: *Tong, Hao, Leandro Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao. “A novel generalized meta-heuristic framework for dynamic capacitated arc routing problems.” IEEE Transactions on Evolutionary Computation, vol. 26, no. 6, pp. 1486-1500, Dec. 2022. <https://doi.org/10.1109/tevc.2022.3147509>.*

3.1 Introduction

DCARP aims to update the service schedule in response to the deterioration of the quality of the original schedule because of some dynamic events in an uncertain environment. When dynamic events happen, vehicles are usually located in different points and have served different tasks resulting in different remaining capacities. Therefore, besides CARP requiring the total demand being served by a vehicle not to exceed a vehicle's capacity, a DCARP instance has two additional constraints, i.e., (1) some vehicles are outside and have to start from these outside positions and (2) the remaining capacities of these outside vehicles are different from those of vehicles in the depot.

Due to additional constraints, the calculation of cost and capacity violation of the routes corresponding to the outside vehicles are required to be specifically considered in DCARP, causing the existing optimization algorithms for CARP not to be directly applicable to solve DCARP instances. However, since both static CARP and DCARP target to optimize the task sequence and schedule the routes for vehicles, the solution representations and search operators designed for static CARP are also likely to be adopted for solving DCARP. Nevertheless, these operators are under the assumption that all vehicles start at the depot and have the same capacities, which is not the case in DCARP. Moreover, it is more complicated to consider each outside vehicle, separately, during the optimization, because each outside vehicle has a different remaining capacity and stop location. There is still no existing work in the current literature that considers leveraging the many powerful optimization algorithms developed for static CARP to solve DCARP. To address these challenges, in this chapter, we propose a novel generalized optimization framework, which enables the adoption of all existing CARP algorithms to DCARP. It is also worthy to mention that the framework does not exclude future development of new dedicated dynamic algorithms.

Therefore, we are going to answer the *RQ1*: “How can we benefit from the powerful al-

gorithms existing for CARP in order to better solve DCARP?”. Our contributions are as follows:

- We propose a novel framework capable of generalizing almost all existing algorithms designed for static CARP to the DCARP context. The framework converts a DCARP instance into a “static” CARP instance by introducing the idea of “virtual tasks”, which enables outside vehicles (with potentially partial capacity) to be interpreted as vehicles located at the depot (with their full capacity). The DCARP instance can then be solved as if it was a static CARP instance by static CARP algorithms. After a solution is found, its corresponding DCARP route where the vehicles start at their outside positions is generated.
- As a dynamic scenario is composed of a series of DCARP instances, similarities between DCARP instances can and should be exploited. Therefore, we propose two strategies for generating initial solutions in our framework, namely a *sequence transfer strategy* and a *restart strategy*, to solve a new DCARP instance. The sequence transfer strategy generates a potentially good solution based on the previous optimization experience by transferring the sequence of remaining unserved tasks. The restart strategy starts from scratch without using any information and optimizes each DCARP instance independently of each other.
- We perform extensive experiments with a variety of DCARP instances, demonstrating the effectiveness of the proposed framework. We show that valuable research progress achieved by the static CARP literature can contribute towards optimization results that significantly outperform the existing algorithm [60] that was specifically designed for DCARP.

The remainder of this chapter is organized as follows. Section 3.2 introduces the main algorithm of our generalized optimization framework for DCARP. Section 3.3 presents our

experimental study on the proposed framework to evaluate its efficiency. Section 3.4 concludes this chapter.

3.2 A Generalized Optimization Framework

In this section, we propose a virtual-task strategy to change a DCARP instance to a ‘virtual static’ instance. After that, a generalized optimization framework based on a virtual-task strategy for DCARP with two different initialization strategies is proposed, which can make use of algorithms for static CARP for solving DCARP.

3.2.1 Virtual Task Strategy

As discussed in the previous section, the main challenge of scheduling vehicles for DCARP by using algorithms designed for static CARP is to take the outside vehicles with different locations and remaining capacities into account. We propose a virtual task strategy that forces all outside vehicles to virtually return to the depot for optimization purposes, such that all vehicles (some virtually) start at the depot during the optimization. As a result, algorithms for static CARP, which assume that all vehicles start at the depot, can be adopted. After the optimization, the obtained solution with routes starting from the depot will be converted to an executable solution according to the locations of outside vehicles. In other words, even though the outside vehicles will virtually return to the depot for running the optimization process, in the executable solutions themselves, the outside vehicles start their new routes from their outside locations. For this strategy to work, some adjustments need to be made so that the optimization problem with virtual tasks is equivalent to the actual DCARP instance being solved. Such adjustments will be explained next.

The pseudocode of constructing the virtual task is presented in Algorithm 2. Despite virtually returning to the depot, the outside vehicles are still required to start from the stop location when executing the new schedule after a change. So, these virtually returned vehicles have to first virtually move to their stop location in the new schedule. Therefore, the vehicles must serve some virtual paths in the new schedule to reach this stop location. The graph of the CARP instance is thus modified to include these virtual paths, which can be regarded as virtual tasks being optimized along with the normal tasks by a static CARP algorithm, as shown in Lines 2-4. We also need all vehicles in the depot to have the same full capacities to be able to use static CARP algorithms. Therefore, we assign the previous demands that have been served by an outside vehicle to the corresponding virtual task, as shown in Line 7. As a result, a DCARP instance is converted to a ‘static’ CARP instance, in which all vehicles are located at the depot with the same capacities. It is worthy to mention that the “virtual task” idea has also been used for large scale CARP [114], which is totally different from our idea here. They used the virtual task to represent the grouped neighboring tasks such that the problem’s dimensionality can be reduced.

Algorithm 2: Pseudocode of constructing virtual tasks

Input: Task set $R = \{t_1, t_2, \dots, t_{N_t}\}$,
Stop locations of outside vehicles: OV ,
Remaining capacity of outside vehicles: RQ .
 $V_{ov} = \{sp_1, sp_2, \dots, sp_{N_{ov}}\}$, $RQ = \{q_1, q_2, \dots, q_{N_{ov}}\}$

Output: The updated task set: E_R

```

1 for each outside vehicle  $k$  do
2   Construct an Arc with virtual task  $vt_k$ ;
3   Head:  $head_{vt_k} = v_0$ , where  $v_0$  is the depot;
4   Tail:  $tail_{vt_k} = sp_k$ ;
5   Deadheading cost:  $dc(vt_k) = \infty$ ;
6   Serving cost:  $sc(vt_k) = mdc(v_0, sp_k)$ ;
7   Demand:  $dm(vt_k) = Q - q_k$ , where  $Q$  is the original capacity of vehicles;
8   Add this virtual task into task set:  $E_R = E_R \cup vt_k$ .
```

A virtual task can also be interpreted as a representation of an outside vehicle’s previous

serving status, including the total cost, served demand and stop location before the occurrence of the dynamic events. During the optimization, the virtual tasks are regarded as arcs to be assigned to routes when being rescheduled. These arcs need to be served, so that some depot vehicles will actually correspond to the outside vehicles. Once a depot vehicle serves a virtual task, its remaining capacity will become the same as the remaining capacity of the corresponding outside vehicle, and so will its stop location. However, vehicles that are not serving these virtual arcs should not be able to traverse them, because these virtual arcs are not actual physical paths that can be used by vehicles. This is achieved by assigning a deadheading cost (traversing cost) of $dc(vt) = \infty$ to these arcs, as shown in Line 5. Note that this infinite traversing cost is not included as part of the serving cost.

The serving cost of a virtual task should be zero. This is because in reality, the outside vehicles are already in the stop locations and have already served some tasks. They should not incur any extra cost to stay where they were. However, in our strategy, a virtual task's serving cost is set as the minimal total deadheading cost between the depot and the vehicle stop location (Line 6) because some algorithms such as Path-Scanning use this cost as a denominator when deciding which task to assign to the current route [36]. To avoid this cost being counted towards the total cost in the objective function, the additional cost will be subtracted from the actual total cost after the optimization using the virtual task strategy. Besides, the demand is set as the amount corresponding to the demand already served by the vehicle, i.e. $Q - q_k$, to avoid the total demand of tasks in its new route exceeding the vehicle's remaining capacity q_k (Line 7).

In order to improve the understanding of how virtual tasks in DCARP instances are constructed, an example for tasks t_1, t_2 is provided in Figure 3.1. One vehicle traverses from v_0 (depot) and serves the task t_1 (Figure 3.1 left). When the dynamic change happens, a vehicle is located at v_2 . This vehicle is virtually placed into the depot and a virtual task vt_1 is constructed between v_0 and v_2 (Figure 3.1 right). This virtual task will enable the vehicle to

go back to its outside position to continue serving other tasks. The demand of vt_1 , $dm(vt_1)$, equals to $Q - dm(t_1)$, such that when the vehicle virtually returns to its outside position, its remaining capacity will be the same as the remaining capacity at the rescheduling point. The deadheading cost of vt_1 , $dc(vt_1)$ is set to ∞ to prevent other vehicles from traversing the virtual arc. This enables the outside vehicle to serve the virtual task. The serving cost of vt_1 , $sc(vt_1)$, equals to $mdc(v_0, v_2)$. This serving cost is incurred by the vehicle when it serves the virtual task, but is later on deducted from the objective function of the problem, so that the objective value with the virtual task strategy remains the same as the objective value without the virtual task strategy. After that, the new DCARP instance with the remaining task t_2 and the virtual task vt_1 will be optimized using one of the algorithms which are available for static CARP, in which the task t_1 is removed because it was already served before the change that triggered the rescheduling.

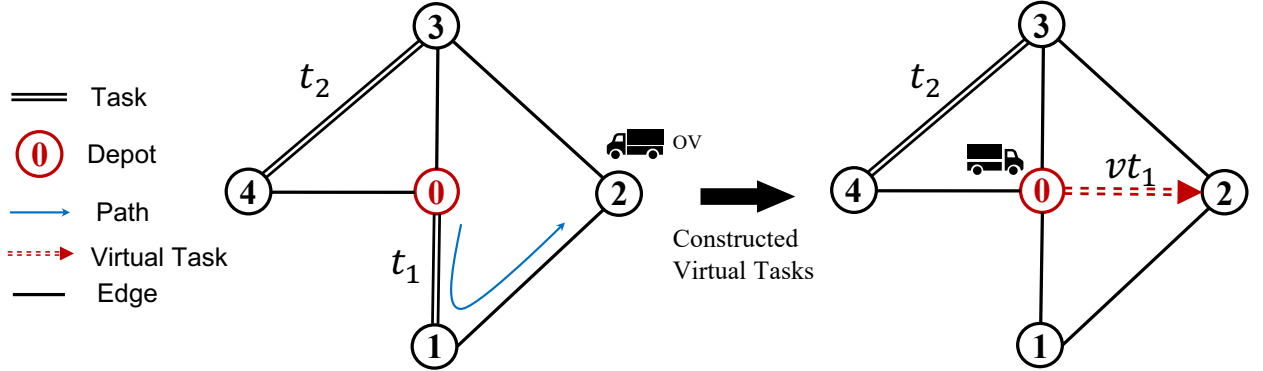


Figure 3.1: An example of constructing virtual tasks.

After applying the virtual-task strategy, the route formulation in a DCARP solution $S = \{r_1, r_2, \dots, r_{N_{ov}}, \dots, r_K\}$ becomes:

$$r_k = (v_0, vt_k, t_{k,2}, t_{k,3}, \dots, t_{k,l_k}, v_0), k = 1, 2, \dots, N_{ov}.$$

$$r_k = (v_0, t_{k,1}, t_{k,2}, \dots, t_{k,l_k}, v_0), k = N_{ov} + 1, \dots, K.$$

The new formulation of the optimization objective and the constraints for a DCARP

instance is given in Eq. 3.1:

$$\begin{aligned}
 \text{Min} \quad & TC(S) = \sum_{k=1}^K RC_{r_k} - \sum_{k=1}^{N_{ov}} mdc(v_0, sp_k). \\
 \text{s.t.} \quad & \sum_{k=1}^K l_k = N_T + N_{ov}. \\
 & t_{k_1, i_1} \neq t_{k_2, i_2}, \text{ for all } (k_1, i_1) \neq (k_2, i_2). \\
 & \sum_{i=1}^{l_k} dm(t_{k,i}) \leq Q, \forall k = 1, \dots, K.
 \end{aligned} \tag{3.1}$$

where for route $\{r_k | k = 1, 2, \dots, N_{ov}\}$, $t_{k,1} = vt_k$. The second term $\sum_{k=1}^{N_{ov}} mdc(v_0, sp_k)$ is to balance out Line 6 in Algorithm 2 so that we do not count the serving costs of virtual tasks.

The above adjustments enable a new schedule for the converted ‘static’ CARP instance to be obtained by directly using meta-heuristic algorithms for static CARP. An executable solution is obtained by removing the virtual tasks from the routes in the new schedule and assigning it to the corresponding outside vehicles. The virtual task is better to be the first task in a vehicle’s route. If a given virtual task is not the first task of a service route found by the static CARP algorithm, the tasks before this virtual task will be assigned to a new vehicle starting from the depot and the following tasks will be served by the corresponding outside vehicle. In such a case, we split a route that contains virtual tasks in the middle into multiple routes with one route served by a vehicle from the depot and others served by the corresponding outside vehicle. In this way, the total cost of the CARP solution will not be influenced by splitting because the head node of the virtual task is also the depot. An example (without relations to Figure 3.1) of converting an obtained new solution to an

executable service plan is provided below:

$$\begin{array}{c}
(v_0, vt_1, t_2, v_0), (v_0, t_3, t_4, vt_2, t_5, v_0) \\
\downarrow \\
(v_0, vt_1, t_2, v_0), (v_0, t_3, t_4, v_0), (v_0, vt_2, t_5, v_0) \\
\downarrow \\
(sp_1, t_2, v_0), (v_0, t_3, t_4, v_0), (sp_2, t_5, v_0).
\end{array}$$

In the above example, the solution (top) contains two routes and two outside vehicles (i.e. two virtual tasks). In the second route, the virtual task located in the middle of the task sequence. If the first task of a route is a virtual task, the remaining tasks of the route are served by the outside vehicle. Otherwise, all tasks of the route are served by a new empty vehicle. Therefore, after conversion (middle), tasks t_3, t_4 are assigned to a new vehicle starting from the depot, but task t_5 should actually be served by the outside vehicle. The final executable routes (bottom), are obtained by removing the virtual tasks and starting from the corresponding stop locations (The sp_1 and sp_2 are the stop locations of outside vehicles corresponding to vt_1 and vt_2).

3.2.2 Proposed Framework based on Virtual-Task Strategy

There are generally two commonly used strategies in dynamic optimization. One is to re-start the optimization, which can also include some additional diversity enhancing techniques [72]. The other is to migrate some good solutions for the old environment to the new environment and initialize the starting individuals with them [73, 89].

These two strategies could also be used in our optimization framework. The restart strategy is straightforward to apply after a change has occurred. However, the current strategies for re-using good solutions for the old environment may not be suitable for our DCARP

scenarios because the dynamic events may influence the problem's dimension and the previous solution's feasibility in a DCARP scenario. For example, the served tasks and added tasks change the total number of tasks, and the potential road closure event makes previous solutions infeasible in the new DCARP instance.

Algorithm 3: Pseudocode of knowledge (sequence)

```

1 transfer strategy Input: Previous best solution  $S_{t-1}$ 
   Output: A newly transferred solution  $S_t$ 
2 Convert  $S_{t-1}$  into an ordered list of tasks  $P_{t-1}$ ;
3 Remove all served tasks from  $P_{t-1}$ ;
4  $P_t = P_{t-1}$ ,  $l_P = |P_t|$ ;
5 The newly added tasks set  $NE_R = \{nt_1, nt_2, \dots, nt_{|NE_R|}\}$ ;
6 for each  $nt_i \in NE_R$  do
7   for each position in  $P_t$  do
8     Calculate the increased cost after insertion;
9   Obtain the position  $p$  with the smallest increased cost;
10  Insertion:  $P_m = [t_1, \dots, t_p, nt_i, t_{p+1}, \dots, t_{l_P}]$ ;
11   $l_P = l_P + 1$ ;
12   $NE_R = NE_R \setminus \{nt_i\}$ .
13 Use split scheme in  $P_t$ .

```

Therefore, we propose a new knowledge (especially sequence) transfer strategy for the DCARP scenario (Algorithm 1), which can benefit from previous DCARP instance's best solutions. Given an instance I_t , all scheduled routes in the best solution S_{t-1} of the previous instance I_{t-1} are concatenated to construct an ordered list of tasks P_{t-1} . All tasks that have been served are then removed from the list in P_{t-1} , and the remaining tasks keep their orders. After that, the newly added tasks are inserted into P_{t-1} greedily, i.e., they are inserted into the positions with the smallest increased cost. Finally, a corresponding transferred solution is generated by using the split scheme to convert an ordered list to an explicit-route solution. The newly transferred solution is used as one of the initial solutions when optimizing the new DCARP instance using the population-based algorithm.

On the basis of the virtual-task strategy and two initialization strategies, i.e. the restart

strategy and the sequence transfer strategy, we propose a generalized optimization framework with virtual tasks (GOFVT) to generalize static CARP algorithms to dynamic scenarios. Our framework comprises four main steps:

1. Construct virtual tasks;
2. Apply the restart strategy (randomly generate initial solutions) or the sequence transfer strategy (generate one transferred solution for individual-based algorithms and additionally generated random solutions for population-based algorithms);
3. Apply the meta-heuristic algorithm to optimize the converted “static” CARP instance;
4. Convert the obtained solution with virtual tasks to an executable solution without virtual tasks.

For a DCARP instance, the framework will first construct the virtual tasks to convert the dynamic instance to a ‘static’ instance. Then, one of the two initialization strategies explained above can be adopted to assist the optimization for the DCARP instance. In the computational studies in Section 3.3, we will compare the effectiveness of these two strategies. Then, meta-heuristic algorithms with the initialization strategy are applied to optimize the DCARP instance which is a ‘static’ instance with virtual tasks. Finally, the solution obtained for the ‘static’ instance is converted into an executable solution, in which the routes with virtual tasks are assigned to the corresponding outside vehicles and the routes without virtual tasks are assigned to vehicles located at the depot.

3.3 Experimental Studies

In order to evaluate the efficiency of our proposed framework (GOFVT), three sets of experimental studies have been conducted by embedding a selection of meta-heuristic algorithms

in the GOFVT in this section. After providing the experimental set-up (Section 3.3.1), in the first experiment, the virtual-task strategy is compared with a simple rescheduling strategy (Section 3.3.2). After that, the virtual-task strategy's efficiency is investigated by comparing it with an existing algorithm for DCARP in the second experiment (Section 3.3.3). Finally, in the last experiment, GOFVT is combined with several classical meta-heuristic algorithms originally designed for static CARP, and its performance is analyzed by running the newly generated algorithms in DCARP scenarios (Section 3.3.4).

3.3.1 Experimental Settings

All experiments are conducted on a series of DCARP instances or scenarios generated by the simulation system presented in Section 2.3, based on a static CARP benchmark, namely the *egl* set [56]. The *egl* set contains 24 CARP instances. In each experiment, the DCARP instances or scenarios are generated independently of static CARP instances. For our first and second set of experiments, 3 DCARP instances are generated for each static CARP instance. These DCARP instances are not used to compose a DCARP scenario, as the algorithms just optimize the current instance and do not use any knowledge transfer in these two experiments. For the third experiment, one DCARP scenario including 5 DCARP instances is generated based on each CARP instance. For a fair comparison, each DCARP instance is generated from the best solution among all obtained solutions by all compared algorithms in the previous instance. When the simulator executes the selected solution according to the deployment policy, the time for serving all tasks will be calculated first, and the simulator will uniformly randomly select a stop point (for dynamic events) within the longest time. As we only require a set of DCARP instances to test the effectiveness of proposed strategies and framework, we have arbitrarily chosen parameters according to real world situations. For example, a road is more likely to become congested than being closed,

hence we set $p_{road} = 0.1$. In our experiments, the parameters of the simulator are chosen as $p_{event} = 0.5$, $p_{road} = 0.1$, $p_{bdrv} = 0.5$, $p_{crr} = 0.3$, $p_{crbb} = 0.6$, $p_{cid} = 0.35$, $p_{add} = 0.35$. In the future, we will carry out a more comprehensive study of the characteristics of simulator in relation to its parameter values. Eight different types of dynamic changes (Table 2.1) are simulated in our simulator excluding the case of “vehicles broken down” because its formulation is the same as the event of added tasks.

Because a key optimization requirement when dynamic changes happen in the real world is to obtain a new solution quickly, we limit the maximum optimization time to 60s for the small problems (E1 $\sim$ E4) and 180s for larger maps (S1 $\sim$ S4) in *egl* for all algorithms. All programs are implemented in C language and run using a PC with an Intel Core i7-8700 3.2GHZ. The source code of our experiments has been available on github¹.

3.3.2 Evaluating the Necessity of Rescheduling in DCARP

Although many algorithms have been proposed to solve DCARP in the literature, a simple baseline strategy, named the return-first strategy, has been ignored. The return-first strategy schedules all outside vehicles back to the depot first in order to convert a DCARP instance to a static one, and then reschedules all vehicles for the new static instance after all vehicles are located at the depot. If the return-first strategy is efficient enough, the direct optimization of a DCARP instance would not be necessary anymore. However, since this has not been shown in the literature so far, in this subsection, we use the proposed virtual-task strategy to solve DCARP and compare it with the return-first strategy to show the importance of optimizing DCARP instances directly instead of ignoring the outside vehicles and assigning new vehicles to all remaining tasks.

In our experiment, the simulation system generates three different DCARP instances for

¹<https://github.com/HawkTom/Dynamic-CARP>

Table 3.1: Results of virtual-task strategy (VT) and return-first strategy (RF) on DCARP instances with different settings of remaining capacities from the *egl* dataset. The value in each cell represents “Mean $\pm$ Std” over 25 independent runs and the bold ones denote the better result on the DCARP instance based on the Wilcoxon signed-rank test with a significance level of 0.05. The penultimate row summaries the number of win-draw-lose of RF strategy versus VT strategy and the last row provides the p-values of the Wilcoxon signed-rank test with a significance level of 0.05 on instances with the same settings of all maps.

Static Map	Instance 1 ($q \in [0, 0.33Q]$)		Instance 2 ($q \in [0.34Q, 0.66Q]$)		Instance 3 ($q \in [0.67Q, Q]$)	
	RF	VT	RF	VT	RF	VT
E1-A	8386 $\pm$ 22	8095$\pm$31	10397 $\pm$ 34	9925$\pm$22	8630 $\pm$ 8	8566$\pm$10
E1-B	9830$\pm$29	9839$\pm$39	9718 $\pm$ 22	8437$\pm$28	9675 $\pm$ 16	9208$\pm$29
E1-C	12181$\pm$52	12109$\pm$52	14726 $\pm$ 31	14106$\pm$16	11286 $\pm$ 11	10565$\pm$25
E2-A	10255$\pm$6	10480 $\pm$ 6	13740 $\pm$ 27	12278$\pm$29	11692 $\pm$ 51	10231$\pm$34
E2-B	13162$\pm$15	13317 $\pm$ 23	16435 $\pm$ 29	15207$\pm$18	15667 $\pm$ 35	14321$\pm$55
E2-C	17425$\pm$41	17566 $\pm$ 35	17444 $\pm$ 32	16881$\pm$25	14186 $\pm$ 16	13871$\pm$46
E3-A	11343 $\pm$ 42	11167$\pm$17	12877 $\pm$ 21	12030$\pm$26	10369 $\pm$ 19	8869$\pm$12
E3-B	15517$\pm$84	15776 $\pm$ 31	15150 $\pm$ 31	12797$\pm$68	20743 $\pm$ 59	20081$\pm$39
E3-C	20549$\pm$69	20880 $\pm$ 63	20012 $\pm$ 144	18078$\pm$88	27521 $\pm$ 81	25289$\pm$82
E4-A	11216$\pm$31	11387 $\pm$ 27	12489 $\pm$ 26	12152$\pm$32	16213 $\pm$ 29	14858$\pm$135
E4-B	15522 $\pm$ 40	15242$\pm$62	14933 $\pm$ 65	13791$\pm$63	17477 $\pm$ 71	15869$\pm$29
E4-C	20928 $\pm$ 63	20691$\pm$41	21998 $\pm$ 28	20174$\pm$86	17869 $\pm$ 79	15798$\pm$59
S1-A	16097 $\pm$ 49	16012$\pm$40	15643 $\pm$ 31	14917$\pm$40	13684 $\pm$ 53	13159$\pm$18
S1-B	20462$\pm$69	21167 $\pm$ 58	16184 $\pm$ 35	15697$\pm$35	15151 $\pm$ 19	13428$\pm$82
S1-C	27944$\pm$71	28624 $\pm$ 116	21040 $\pm$ 36	18616$\pm$87	26470 $\pm$ 103	25789$\pm$76
S2-A	19610 $\pm$ 85	19479$\pm$107	22314 $\pm$ 40	20016$\pm$48	22499 $\pm$ 67	19212$\pm$82
S2-B	27970$\pm$75	28408 $\pm$ 59	26334 $\pm$ 130	25911$\pm$118	25723 $\pm$ 67	25024$\pm$94
S2-C	31859$\pm$108	32101 $\pm$ 125	40170 $\pm$ 132	38575$\pm$186	35053 $\pm$ 143	31815$\pm$133
S3-A	19489 $\pm$ 64	19211$\pm$107	23630 $\pm$ 118	21329$\pm$98	28237 $\pm$ 76	24494$\pm$71
S3-B	27518$\pm$72	27521$\pm$98	25651 $\pm$ 64	24311$\pm$93	32286 $\pm$ 50	30646$\pm$96
S3-C	32468 $\pm$ 93	32049$\pm$107	35435 $\pm$ 67	33927$\pm$68	43151 $\pm$ 98	40801$\pm$97
S4-A	27256$\pm$87	28082 $\pm$ 142	23780 $\pm$ 74	22788$\pm$95	29895 $\pm$ 102	27576$\pm$105
S4-B	34872 $\pm$ 122	34709$\pm$149	28283 $\pm$ 121	28023$\pm$93	31770 $\pm$ 87	30652$\pm$125
S4-C	43746$\pm$168	44165 $\pm$ 157	36808 $\pm$ 117	36706$\pm$123	44006 $\pm$ 145	42827$\pm$178
#of ‘w-d-l’	12-3-9		0-0-24		0-0-24	
p-value	0.22		1.82e-5		1.82e-5	

each test map (i.e., a static instance in the *egl* benchmark set) with different sets of remaining capacities. As this experiment aims to demonstrate the necessity and efficiency of directly optimizing DCARP instances, the setting of remaining capacities is divided into three intervals, i.e., $[0, 0.33Q]$, $[0.34Q, 0.66Q]$ and $[0.67Q, Q]$. Then, an optimization algorithm, Memetic Algorithm with Extended Neighborhood Search (MAENS) [113], assisted with the return-first strategy and virtual-task strategy are applied to optimize each DCARP instance, respectively. Two algorithm instantiations using MAENS follow the same setting during the optimization such that the return-first strategy and virtual-task strategy is the only difference between them. The comparison results in terms of mean and standard deviation over 25 independent runs (mean $\pm$ std), of the return-first (RF) strategy and virtual-task (VT) strategy on DCARP instances with different remaining capacities are presented in Table 3.1. The bold values with gray background for each DCARP instance are the better results between return-first strategy and virtual-task strategy based on the Wilcoxon signed-rank test with a significance level of 0.05. The second last row of Table 3.1 summarizes the number of win-draw-lose of the RF versus VT strategies. We have calculated the Wilcoxon signed-rank test with a significance level of 0.05 for the mean total cost of RF and VT strategies on the instances with the same range of remaining capacities, and the p-values are listed in the last row of Table 3.1.

Table 3.1 shows that the RF and VT strategies are significantly different for the instances where the remaining capacities are in the range of $[0.34Q, Q]$ (instances 2 and 3 in Table 3.1), with the VT strategy outperforming the RF strategy on all DCARP instances. In contrast, for the scenarios with remaining capacities in $[0, 0.33Q]$, there are 12 out of 24 DCARP instances where the RF strategy outperforms the VT strategy. When comparing the results using the Wilcoxon test across maps, we confirm that none of these strategies is a consistent winner when analyzed across maps where the remaining capacities are smaller than $0.33Q$. This is understandable because when the vehicles are mostly fully loaded, i.e.,

when the remaining capacities are smaller than $0.33Q$, there is limited space for serving more tasks no matter what strategy is used.

Overall, we can conclude that it is necessary and much more effective to optimize the DCARP instance directly rather than using the RF strategy when outside vehicles have enough remaining capacities.

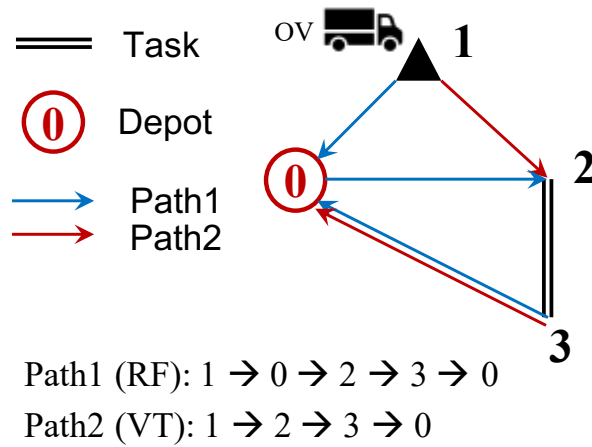


Figure 3.2: An example of demonstrating why the RF strategy is not efficient enough when outside vehicles have enough remaining capacities.

The reason why the RF strategy is not always helpful when outside vehicles have enough remaining capacities can be explained using a simple example in Figure 3.2, where an outside vehicle stops at vertex 1. If its remaining capacity is sufficient to serve task t_{23} , it can directly traverse from vertex 1 to vertex 2, presented as ‘Path 2’ in Figure 3.2, and the final total cost will be $RC = mdc(v_1, v_2) + mdc(v_2, v_3) + mdc(v_3, v_0)$. But if we apply the RF strategy, the total cost will change to $RC = mdc(v_1, v_0) + mdc(v_0, v_2) + mdc(v_2, v_3) + mdc(v_3, v_0)$, presented as ‘Path 1’ in Figure 3.2. It is obvious that the total cost for the RF strategy is greater than that of the VT strategy because $mdc(v_1, v_0) + mdc(v_0, v_2) \geq mdc(v_1, v_2)$ according to triangle inequality. The RF strategy increases the final cost because vehicles take a detour in such cases.

3.3.3 Analysis of the Effects of the Virtual-Task Strategy

Memetic algorithm with new split scheme (MASDC) [60] is the only meta-heuristic algorithm for DCARP in the literature that considers a general DCARP scenario including several dynamic events, such as road closure and added tasks. It comprises a distance-based split scheme to assist the DCARP solution being used in the crossover and local search. Our virtual-task strategy can convert a DCARP instance to a ‘static’ CARP instance so that the operator used in the static CARP can be used in the DCARP instance directly. In this subsection, we analyse the effects of our VT strategy by embedding it to MASDC, referred to as VT-MASDC, and comparing it to the original MASDC. The advantage of embedding our strategy into MASDC is that this enables us to isolate and analyse the effect of the virtual tasks compared to a state-of-the-art DCARP algorithm. In particular, all components in MASDC and VT-MASDC are the same except for the use of virtual tasks and the distance-based split scheme. The latter needs to be replaced by Ulusoy’s split scheme in VT-MASDC because the distance-based split scheme is specifically designed for DCARP instances with outside vehicles at different stop locations. When we apply the virtual task strategy, the DCARP instance is converted to the ‘static’ instance where all vehicles are located at the depot. Therefore, the distance-based split scheme is not suitable anymore and the Ulusoy’s split scheme is used instead. The use of virtual tasks is thus inherently linked to this split scheme, and any advantages provided by the virtual tasks are also linked to the fact that they enable this split scheme to be adopted.

In our experiment, we generate three independent DCARP instances for each map in the *egl* benchmark set, ensuring that outside vehicles has enough remaining capacities, i.e. $q \geq 0.5Q$, in all generated instances. However, we need to generate DCARP instances rather than DCARP scenarios because the aim of DCARP is to minimize the total cost for each DCARP instance separately (Eq. 2.3). The optimization results are presented in Table

Table 3.2: Results of MASDC and VT-MASDC on DCARP instances from the *egl* dataset. The value in each cell represents “Mean $\pm$ Std” over 25 independent runs and the bold ones denote the better result on the DCARP instance based on the Wilcoxon signed-rank test with the level of significance 0.05. The last row summaries the number of win-draw-lose of MASDC versus VT-MASDC.

Map	Ins	MASDC	VT-MASDC	Map	Ins	MASDC	VT-MASDC
E1-A	1	19354 $\pm$ 748	16180$\pm$1716	S1-A	1	35122 $\pm$ 1318	28122$\pm$2707
	2	15074 $\pm$ 529	12551$\pm$1584		2	37189 $\pm$ 1362	32033$\pm$2467
	3	18528 $\pm$ 724	15927$\pm$1946		3	40972 $\pm$ 1185	34348$\pm$2529
E1-B	1	20644 $\pm$ 783	17229$\pm$1644	S1-B	1	38612 $\pm$ 1196	33192$\pm$2775
	2	18052 $\pm$ 716	15889$\pm$1394		2	36846 $\pm$ 1292	32335$\pm$2240
	3	18947 $\pm$ 564	15691$\pm$1664		3	37521 $\pm$ 1153	32815$\pm$2734
E1-C	1	22591 $\pm$ 754	19104$\pm$1379	S1-C	1	42269 $\pm$ 1540	37935$\pm$2671
	2	18778 $\pm$ 556	16192$\pm$1444		2	36263 $\pm$ 971	31978$\pm$2606
	3	23001 $\pm$ 1039	18119$\pm$1755		3	48458 $\pm$ 1389	43726$\pm$2405
E2-A	1	23673 $\pm$ 917	20981$\pm$1881	S2-A	1	57452 $\pm$ 1384	51503$\pm$4217
	2	19463 $\pm$ 762	16235$\pm$2155		2	58765 $\pm$ 1622	52707$\pm$3584
	3	18319 $\pm$ 697	15843$\pm$1800		3	45582 $\pm$ 1390	40030$\pm$2801
E2-B	1	25745 $\pm$ 869	21650$\pm$2151	S2-B	1	64514 $\pm$ 2001	54667$\pm$3991
	2	23281 $\pm$ 816	20146$\pm$1744		2	58803 $\pm$ 1833	53834$\pm$2675
	3	23517 $\pm$ 941	19071$\pm$2264		3	56169 $\pm$ 1253	48964$\pm$2281
E2-C	1	29112 $\pm$ 1014	24493$\pm$2035	S2-C	1	65181 $\pm$ 1799	58049$\pm$2561
	2	23363 $\pm$ 682	20451$\pm$1624		2	72010 $\pm$ 1674	64127$\pm$3558
	3	27457 $\pm$ 896	24353$\pm$1883		3	75819 $\pm$ 2091	69322$\pm$2828
E3-A	1	24838 $\pm$ 1030	20851$\pm$1867	S3-A	1	49937 $\pm$ 1061	43826$\pm$3168
	2	25274 $\pm$ 1011	21138$\pm$2332		2	45992 $\pm$ 1571	39608$\pm$3019
	3	26588 $\pm$ 1254	22012$\pm$2441		3	43552 $\pm$ 981	36280$\pm$2977
E3-B	1	27026 $\pm$ 893	22399$\pm$2364	S3-B	1	56644 $\pm$ 1736	51214$\pm$2533
	2	27106 $\pm$ 812	21950$\pm$2649		2	61856 $\pm$ 1440	53962$\pm$3334
	3	23861 $\pm$ 833	20398$\pm$2059		3	66870 $\pm$ 1877	57878$\pm$3772
E3-C	1	31574 $\pm$ 942	27110$\pm$2354	S3-C	1	62193 $\pm$ 1778	56232$\pm$3060
	2	32775 $\pm$ 1018	28566$\pm$1542		2	69501 $\pm$ 1601	62797$\pm$3424
	3	31589 $\pm$ 679	28282$\pm$1855		3	79699 $\pm$ 1891	71613$\pm$3912
E4-A	1	20835 $\pm$ 854	17411$\pm$1440	S4-A	1	58456 $\pm$ 1621	51512$\pm$3298
	2	23469 $\pm$ 939	20619$\pm$1781		2	55394 $\pm$ 1746	49422$\pm$3517
	3	25219 $\pm$ 896	22005$\pm$2083		3	64795 $\pm$ 1546	58676$\pm$3517
E4-B	1	23648 $\pm$ 946	20050$\pm$1708	S4-B	1	64343 $\pm$ 1886	57017$\pm$3622
	2	26950 $\pm$ 779	23664$\pm$2491		2	65027 $\pm$ 1712	57413$\pm$3033
	3	25755 $\pm$ 844	21767$\pm$2165		3	60143 $\pm$ 1612	53342$\pm$3323
E4-C	1	28338 $\pm$ 812	25212$\pm$1430	S4-C	1	69224 $\pm$ 1475	62760$\pm$2929
	2	30586 $\pm$ 708	26925$\pm$1560		2	68070 $\pm$ 1424	62304$\pm$2465
	3	28152 $\pm$ 762	24482$\pm$1979		3	78134 $\pm$ 1893	69348$\pm$3451
				# of w-d-l		0-0-72	
				p-value		1.67e-13	

3.2, in which the values in each cell represent the mean and standard deviation over 25 independent runs (mean $\pm$ std). For each DCARP instance, the better result, based on the Wilcoxon signed-rank test with a significance level of 0.05, is highlighted using a bold font and gray background in Table 3.2. The summary of win-draw-lose of MASDC versus VT-MASDC, which is presented in the last row, shows that VT-MASDC outperforms MASDC on all generated DCARP instances. The Wilcoxon signed-rank test with a significance level of 0.05 was conducted for the average total cost of VT-MASDC and MASDC in all instances, and the p-value was 1.67e-13. Overall, we can conclude that VT-MASDC performs much better than the original MASDC.

MASDC uses a distance-based split scheme (DSS) to evaluate DCARP solution's fitness because the split scheme designed for static CARP is not suitable for DCARP [119]. The DSS operator randomly splits the sequence of tasks to an executable CARP solution with explicit routes, and the random splitting process is repeated three times to obtain a better schedule. After embedding with the VT strategy, the DSS operator is replaced by Ulusoy's split [120] as explained in the beginning of this subsection, and the evaluation of a sequence of tasks only requires to apply the Ulusoy's split once. As a result, the randomness brought by the DSS's split scheme is removed.

Moreover, the DSS operator never considers new vehicles starting from the depot during the optimization, whereas our VT strategy enables both outside and new vehicles to be used. We provide an example in Figure 3.3 to show the advantages of considering new vehicles during the optimization of DCARP. An outside vehicle is located at vertex 1, and its remaining capacity can only serve task t_{23} . We assume that $mdc(v_0, v_1) + mdc(v_0, v_2) \approx mdc(v_1, v_2)$ and $mdc(v_3, v_0) + mdc(v_0, v_4) \gg mdc(v_3, v_4)$ and the total cost of 'Path1' and

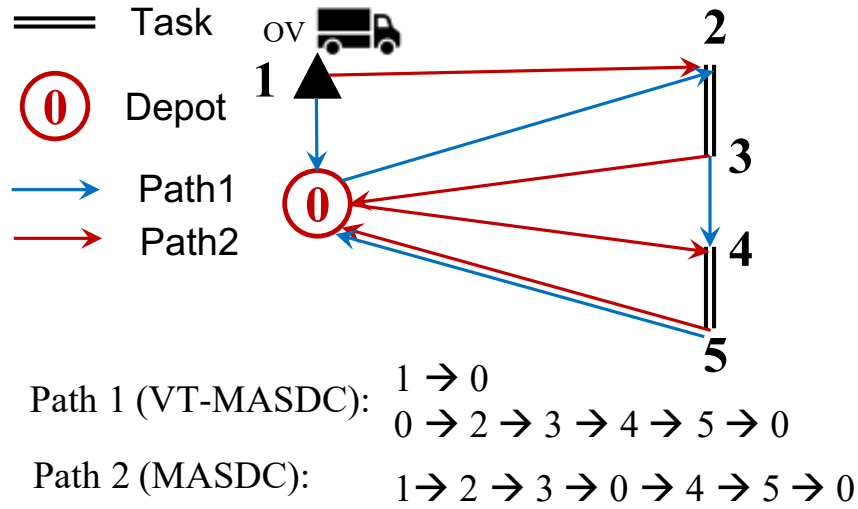


Figure 3.3: An example of demonstrating the advantages of considering new vehicles.

‘Path2’ can be calculated as:

$$\begin{aligned}
 TC_1 &= \underline{mdc}(v_1, v_0) + \underline{mdc}(v_0, v_2) + sc(t_{23}) + \underline{mdc}(v_3, v_4) + sc(t_{45}) + \underline{mdc}(v_5, v_0), \\
 TC_2 &= \underline{mdc}(v_1, v_2) + sc(t_{23}) + \underline{mdc}(v_3, v_0) + \underline{mdc}(v_0, v_4) + sc(t_{45}) + \underline{mdc}(v_5, v_0).
 \end{aligned}$$

For a sequence of tasks $[0, t_{23}, t_{45}, 0]$, if applied with the DSS operator, the only obtained path will be ‘Path 2’ as shown in Figure 3.3. In contrast, if we use the VT strategy, a better path, i.e. ‘Path 1’ in Figure 3.3, can be obtained, which avoids traversing the longer returning path.

3.3.4 Analysis of the Proposed Framework

The proposed optimization framework GOFVT is capable of generalizing almost all algorithms for static CARPs to optimize DCARP. To demonstrate its effectiveness and efficiency, we have selected three classical meta-heuristic algorithms for static CARPs, namely RTS [78], ILMA [82] and MAENS [113], and embedded them into the GOFVT in our experiments to evaluate whether it is advantageous to make use of existing static CARP algorithms within

the GOFVT framework.

A brief description of each algorithm is presented below.

- **RTS** [78]: A global repair operator which is embedded in a tabu search algorithm (TSA [14]). Source code is available online².
- **ILMA** [82]: An improved version of Lacomme’s memetic algorithm (LMA) [54]. For our experiments, we implemented ILMA³ ourselves according to the details given in [82].
- **MAENS** [113]: A memetic algorithm with a merge-split operator. Source code is available online⁴.

The newly generated algorithms for optimizing DCARP are denoted as VT-RTS, VT-ILMA and VT-MAENS. We use these acronyms to denote GOFVT with the restart strategy. When using the sequence transfer strategy, they are denoted as VTtr-RTS, VTtr-ILMA, VTtr-MAENS.

To demonstrate the efficiency and robustness of our proposed framework, we use parameters as given in their original papers [78, 82, 113]. In order to compare the restart and sequence transfer strategies, a DCARP scenario consisting of 5 DCARP instances has been generated for each static map in *egl* benchmark set in our experiments. A new DCARP instance is generated by executing the obtained best solution of the previous DCARP instance in our simulation system.

We have also embedded MASDC into GOFVT and generated two algorithms as VT-MASDC and VTtr-MASDC, respectively. The results of above eight algorithms are presented

²<https://meiyi1986.github.io/publication/mei-2009-global/code.zip>

³<https://github.com/HawkTom/Dynamic-CARP>

⁴<https://meiyi1986.github.io/publication/tang-2009-memetic/code.zip>

in Table 3.3. The values in each cell represent the ‘mean $\pm$ std’ and the average ranking (in the brackets) among 8 algorithms in terms of the average total cost over 25 independent runs. The bold values represent the better result between the restart strategy and the sequence transfer strategy for each algorithm on a DCARP instance under the Wilcoxon signed-rank test with a significance level of 0.05. Rows where both the restart strategy and sequence transfer strategy are in bold imply that these two strategies are not significantly different under the hypothesis test. The summaries of win-draw-lose of the restart strategy versus the sequence transfer strategy on all DCARP instances are listed in the penultimate row. We have also conducted the Wilcoxon signed-rank test with a significance level of 0.05 for the average total cost of the restart strategy and sequence transfer strategy of each meta-heuristic algorithm on all instances, and the p-values associated with each meta-heuristic algorithm are listed in the last row of Table 3.3.

We conclude that the restart strategy is significantly different from the sequence transfer strategy when embedded in RTS ($0.01 < 0.05$), but both strategies obtain a similar performance when embedded in ILMA ($0.15 > 0.05$), MAENS ($0.93 > 0.05$) and MASDC ($0.95 > 0.05$). This is mainly because RTS is an individual-based meta-heuristic algorithm while ILMA, MAENS and MASDC are population-based algorithms. An individual-based algorithm only uses one solution during optimization. The solution generated by the sequence transfer strategy will be the only initial solution in the individual-based algorithm and therefore significantly influences the optimization result. In contrast, the population-based algorithm contains a population during the optimization so that it depends much less on a single transferred solution.

From the statistical test result and the number of ‘win-draw-lose’ of all DCARP instances for the individual-based algorithm, i.e. RTS, we can conclude that the performance of the sequence transfer strategy is better than the restart strategy. The efficiency depends on how much information is transferred from the previous best solution. For a CARP solution, the

Table 3.3: Results of VT-RTS, VTtr-RTS, VTtr-ILMA, VTtr-MAENS, VTtr-MAENS, VT-MASDC and VTtr-MASDC on the *egl* dataset. The values in each cell represent the “MEAN $\pm$ STD” with the average ranking (in the brackets) w.r.t the average total cost over 25 independent runs. The bold values are the better results under the Wilcoxon signed-rank test with a significance level of 0.05 between restart and inheriting strategies in an algorithm for a DCARP instance. The last three rows summarize the number of win-draw-lose of restart strategy versus inheriting strategy in each algorithm and the p-values of the Wilcoxon signed-rank test with a significance level of 0.05 on all instances.

Map	Ins.Index	VT-RTS	VTtr-RTS	VT-ILMA	VTtr-ILMA	VT-MAENS	VTtr-MAENS	VT-MASDC	VTtr-MASDC
S3-A	1	22989$\pm$156(3.6)	22971$\pm$190(3.4)	24100$\pm$307(5.5)	24073$\pm$289(5.5)	22411$\pm$61(1.5)	22401$\pm$70(1.5)	52327$\pm$3849(7.5)	52327$\pm$3849(7.5)
	2	25249$\pm$348(2.5)	26012 $\pm$ 192(4.0)	27134$\pm$338(5.3)	27234$\pm$405(5.6)	25013$\pm$105(1.8)	25017$\pm$103(1.8)	57870$\pm$3931(7.5)	57870$\pm$3931(7.5)
	3	25907$\pm$350(3.3)	26117 $\pm$ 160(3.7)	27011$\pm$250(5.5)	27042$\pm$320(5.5)	25285$\pm$145(1.4)	25349$\pm$113(1.6)	59652$\pm$3792(7.5)	59652$\pm$3792(7.5)
	4	41928 $\pm$ 177(6)	33347$\pm$287(9)	34800$\pm$398(4.5)	34840$\pm$420(4.5)	32374$\pm$119(1.4)	32423$\pm$100(1.6)	72252$\pm$4788(7.5)	72250$\pm$4790(7.5)
	5	31319 $\pm$ 4(6)	24329$\pm$316(2.7)	25614$\pm$322(4.6)	25632$\pm$334(4.4)	23989$\pm$74(1.7)	24006$\pm$85(1.6)	62362$\pm$4014(7.5)	62362$\pm$4014(7.5)
S3-B	1	30945 $\pm$ 70(6)	29429$\pm$141(5)	26486$\pm$282(3.4)	26493$\pm$230(3.6)	25161$\pm$123(1.5)	25165$\pm$106(1.6)	46491$\pm$3247(7.5)	46491$\pm$3247(7.5)
	2	28494$\pm$1307(3.5)	29198 $\pm$ 1867(4.1)	29318$\pm$356(5.2)	29374$\pm$347(5.2)	27445$\pm$121(1.8)	27356$\pm$140(1.2)	55642$\pm$2518(7.5)	55642$\pm$2518(7.5)
	3	36665$\pm$2093(5.3)	35529$\pm$3448(3.8)	34378$\pm$330(4.2)	34315$\pm$347(4.7)	32584$\pm$78(1.5)	32613$\pm$112(1.5)	65559$\pm$3265(7.5)	65559$\pm$3265(7.5)
	4	36424 $\pm$ 10(6)	29498$\pm$240(3.0)	30262$\pm$280(4.6)	30306$\pm$339(4.4)	28983$\pm$133(1.6)	28930$\pm$131(1.4)	59740$\pm$4565(7.5)	59740$\pm$4565(7.5)
	5	32715$\pm$2377(4.0)	31524$\pm$348(3.7)	32271$\pm$945(5.2)	32185$\pm$417(5.1)	30444$\pm$167(1.4)	30437$\pm$160(1.6)	61165$\pm$3506(7.5)	61165$\pm$3506(7.5)
S3-C	1	45906$\pm$0(5.0)	47919 $\pm$ 1147(6.0)	40932$\pm$333(3.4)	40930$\pm$425(3.6)	39026$\pm$146(1.4)	39049$\pm$154(1.6)	69270$\pm$3741(7.5)	69270$\pm$3741(7.5)
	2	58082$\pm$263(5.0)	58770 $\pm$ 195(6.0)	51538$\pm$346(3.6)	51445$\pm$366(3.4)	49131$\pm$169(1.5)	49157$\pm$207(1.5)	78938$\pm$4233(7.5)	78938$\pm$4233(7.5)
	3	45802$\pm$74(5.2)	45868 $\pm$ 21(5.8)	38727$\pm$341(3.4)	38925$\pm$241(3.6)	36791$\pm$118(1.6)	36730$\pm$94(1.4)	67602$\pm$3362(7.5)	67602$\pm$3362(7.5)
	4	41011 $\pm$ 1(6)	40879$\pm$115(5)	35457$\pm$919(3.5)	35339$\pm$338(3.5)	33663$\pm$89(1.7)	33633$\pm$91(1.3)	61607$\pm$2949(7.5)	61607$\pm$2949(7.5)
	5	49309$\pm$114(5.5)	46793$\pm$3817(4.8)	42843$\pm$488(3.9)	42764$\pm$568(3.7)	40153$\pm$107(1.6)	40143$\pm$132(1.4)	72621$\pm$3858(7.5)	72621$\pm$3858(7.5)
S4-A	1	24849 $\pm$ 1470(6)	21105$\pm$170(2.9)	22336$\pm$255(4.5)	22385$\pm$332(4.5)	20730$\pm$131(1.6)	20699$\pm$85(1.5)	53513$\pm$3603(7.5)	53513$\pm$3603(7.5)
	2	24693$\pm$1073(3.1)	24947 $\pm$ 322(4.0)	26177$\pm$477(5.4)	26175$\pm$352(5.5)	23967$\pm$134(1.6)	23920$\pm$96(1.4)	59023$\pm$3019(7.5)	59020$\pm$3023(7.5)
	3	28262 $\pm$ 12(6)	22203$\pm$286(9)	23181$\pm$956(4.4)	23091$\pm$319(4.5)	21661$\pm$67(1.6)	21638$\pm$68(1.4)	49301$\pm$3606(7.5)	49301$\pm$3606(7.5)
	4	28294 $\pm$ 261(4.6)	27786$\pm$653(3.1)	28551$\pm$420(5.0)	28636$\pm$379(5.4)	26885$\pm$82(1.7)	26856$\pm$103(1.3)	62334$\pm$3972(7.5)	62334$\pm$3972(7.5)
	5	28727$\pm$249(3.0)	28989 $\pm$ 389(3.6)	30571$\pm$479(5.5)	30693$\pm$441(5.5)	28421$\pm$139(1.6)	28440$\pm$125(1.8)	64242$\pm$5138(7.5)	64242$\pm$5138(7.5)
S4-B	1	47887 $\pm$ 173(6)	40827$\pm$2573(3.3)	41301$\pm$476(4.4)	41242$\pm$464(4.3)	38564$\pm$117(1.6)	38549$\pm$139(1.4)	70139$\pm$4037(7.5)	70139$\pm$4037(7.5)
	2	43677 $\pm$ 0(5.6)	43024$\pm$2961(4.7)	40667$\pm$448(3.7)	40804$\pm$492(4.0)	38055$\pm$128(1.5)	38061$\pm$172(1.5)	70219$\pm$3947(7.5)	70219$\pm$3947(7.5)
	3	50458$\pm$193(3.4)	50440$\pm$239(3.2)	52701$\pm$547(5.6)	52632$\pm$519(5.4)	50091$\pm$210(1.6)	50103$\pm$151(1.8)	83768$\pm$3444(7.5)	83768$\pm$3444(7.5)
	4	34946 $\pm$ 288(5.6)	33974$\pm$1347(3.2)	34526$\pm$317(4.4)	34478$\pm$318(4.7)	32820$\pm$95(1.5)	32864$\pm$148(1.5)	62765$\pm$3949(7.5)	62765$\pm$3949(7.5)
	5	38698$\pm$4308(4.5)	38901$\pm$3107(4.8)	36166$\pm$349(4.4)	36195$\pm$289(4.4)	34049$\pm$163(1.6)	34016$\pm$143(1.4)	65365$\pm$4152(7.5)	65365$\pm$4152(7.5)
S4-C	1	42365 $\pm$ 10(5.7)	42128$\pm$314(5.3)	35903$\pm$393(3.5)	35922$\pm$456(3.5)	33866$\pm$168(1.5)	33892$\pm$143(1.4)	61633$\pm$3417(7.5)	61633$\pm$3417(7.5)
	2	49614$\pm$26(5)	49797 $\pm$ 74(6)	44052$\pm$394(3.4)	44087$\pm$340(3.6)	41842$\pm$170(1.4)	41878$\pm$204(1.6)	71240$\pm$3779(7.5)	71240$\pm$3779(7.5)
	3	50103 $\pm$ 0(6)	49263$\pm$282(5)	41426$\pm$369(3.6)	41299$\pm$442(3.4)	39367$\pm$173(1.4)	39386$\pm$133(1.6)	71625$\pm$2776(7.5)	71625$\pm$2776(7.5)
	4	51311 $\pm$ 0(6)	50781$\pm$48(5)	44940$\pm$369(3.4)	45099$\pm$410(3.6)	42895$\pm$168(1.4)	42939$\pm$179(1.6)	72347$\pm$3396(7.5)	72347$\pm$3396(7.5)
	5	46067 $\pm$ 54(6)	45495$\pm$105(5)	40747$\pm$390(3.6)	40751$\pm$340(3.4)	38984$\pm$120(1.4)	38994$\pm$79(1.6)	66381$\pm$3802(7.5)	66381$\pm$3802(7.5)
Statistical results over 120 instances below									
Avr.Ranking		5.25	4.78	3.95	4.00	1.51	1.51	7.50	7.5
#of 'win-draw-lose'		43-18-59	4-116-0	4-116-0	4-116-0	1-118-1	0-120-0	0-120-0	0-120-0
p-value		0.01	0.15	0.15	0.15	0.93	0.95	0.95	0.95

The data of E1-A $\sim$ S2-C are omitted in the table due to the page limitation. The complete data has been attached as supplementary material.

most critical information is the sequence of tasks of each route. Therefore, if each route has many tasks left, and these remaining tasks can also maintain the tasks' sequence of the best solution in the previous instance, the transferred solution will be of high quality. The sequence transfer strategy fixes the remaining tasks' position to maintain the sequence information. Then, new tasks are inserted into the sequence to construct a new initial solution. If an outside vehicle has only a few remaining tasks, most tasks in the new schedule will be the new tasks so that the new schedule is unlikely to benefit much from the previous best schedule.

In contrast, if there are many remaining tasks for an outside vehicle, the order of remaining tasks will be maintained in the new sequence of tasks, and the Ulusoy's split will still assign them to an outside vehicle. Consider the following two remaining task sequences as an example:

$$S_1 : (v_0, vt_1, t_1, v_0, vt_2, t_2, vt_3, t_3, v_0),$$

$$S_2 : (v_0, vt_1, t_1, t_2, t_3, t_4, t_5, v_0, vt_2, t_6, t_7, t_8, t_9, t_{10}, t_{11}, v_0).$$

S_1 has three outside vehicles with one remaining task for each vehicle, and S_2 has two outside vehicles with 5 and 6 remaining tasks, respectively. The task sequences in both remaining task sequences are the same as those of the previous instance's best schedule. After greedy insertion of a set of new tasks, the final task sequence served by outside vehicles in S_2 will be more similar to the schedule in the previous instance's best solution. As a result, the second scenario is more likely to obtain a high-quality transferred solution.

We have calculated the average remaining tasks for outside vehicles on all DCARP instances in our experiment in Figure 3.4. We can conclude that as the average remaining tasks for each outside vehicle increases, the sequence transfer strategy benefits more from the optimization experience in the previous instance and outperforms the restart strategy with a higher probability.

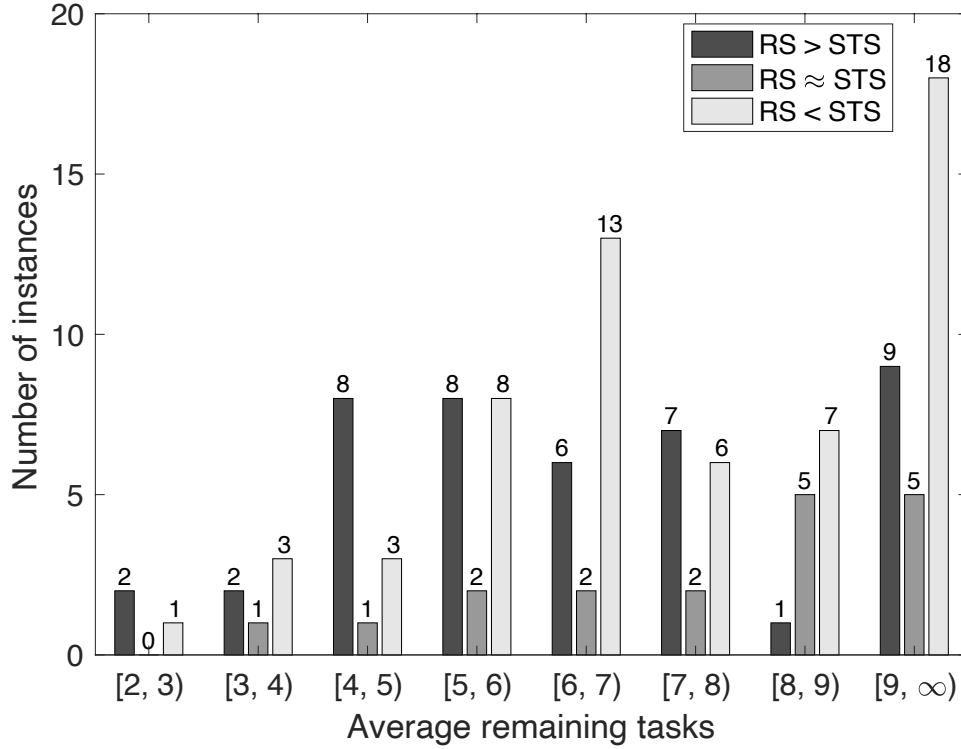


Figure 3.4: The number of instances for the restart strategy (RS) wins over ($\succ$), draws against ($\approx$) and loses to ($\prec$) the sequence transfer strategy (STS) for different ranges of average remaining tasks.

We also compared the rankings of each algorithm over 25 independent runs on each DCARP instance, which is presented in each cell's bracket in Table 3.3. The overall average rankings of each algorithm instance over 120 DCARP instances are summarized in the row of 'Avr. Ranking' in Table 3.3. The Friedman test with a significance level of 0.05 was carried out to compare the ranking of eight algorithms across problem instances, leading to a p-value of 8.69e-159. This indicates that at least one pair of algorithms are not equivalent to each other. We then perform Nemenyi post-hoc tests to identify which algorithms perform significantly different. The critical difference diagram is presented in Figure 3.5 where the value of critical difference is 0.96 [23]. We can conclude from these results that the restart and sequence transfer strategies obtained almost the same performance for population-based algorithms (MAENS and ILMA) while the sequence transfer strategy slightly outperformed the restart strategy in the individual-based algorithm (RTS) in our experiments.

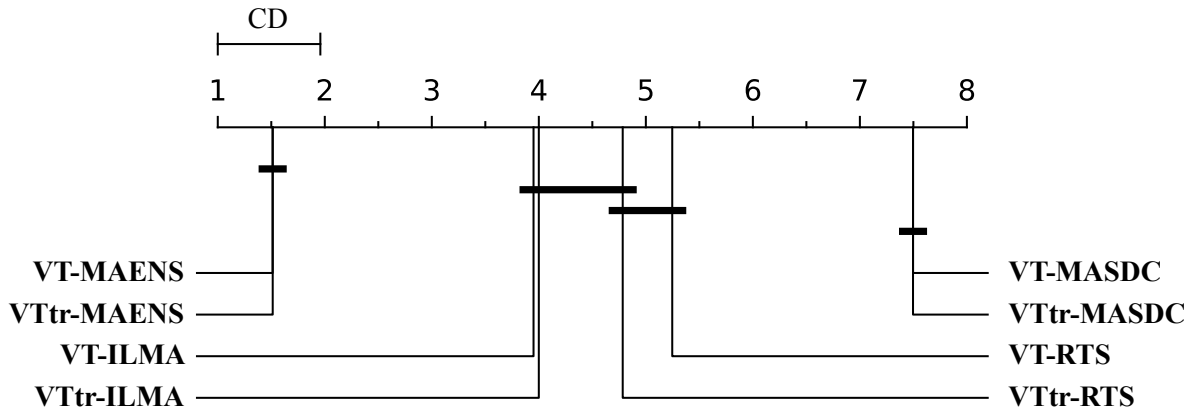


Figure 3.5: Critical difference diagram for the comparison of 8 algorithms against each other on *egl* with Friedman test and Nemenyi test. Groups of algorithms which are not significantly different at the level of 0.05 are connected.

Furthermore, MAENS obtains the best overall result, and RTS is the worst among the three meta-heuristic algorithms which are originally designed for static CARP. However, all of them significantly outperform the state-of-the-art dynamic algorithm, i.e. VT-MASDC and VTtr-MASDC. Recall that VT-MASDC was shown to outperform MASDC in Section 3.3.3. Therefore, we can conclude that not only the proposed framework generalizes the existing algorithms for static CARPs to solve DCARPs but also that the constructed algorithm maintains its superior performance when optimizing a DCARP instance.

3.4 Summary

In this chapter, we proposed a generalized optimization framework for DCARP which can generalize algorithms for static CARPs to optimize DCARPs. In our framework, we proposed a virtual-task strategy that constructs a virtual task between the stop location and the depot, to make all outside vehicles virtually return to the depot, which tremendously simplified the optimization for DCARP. As a result, the DCARP instance was converted into a virtual ‘static’ instance so that algorithms for static CARP can solve it.

Two initialization strategies were adopted in our generalized optimization framework: the restart and sequence transfer strategies. The restart strategy completely restarts the optimization algorithm at random when there is a new DCARP instance. The sequence transfer strategy maintains the sequence of remaining tasks to the new DCARP instance and greedily inserts the new tasks into the remaining sequence, to transfer the information of task sequence from the previous optimization experience.

In our computational study, the necessity of directly optimizing DCARP together with outside vehicles was first demonstrated by comparing the virtual-task strategy with a return-first strategy. The results indicated that it would be more efficient to optimize the DCARP instance by using the virtual-task strategy when the outside vehicles' remaining capacities were sufficiently large to serve more tasks. Then, the efficiency of the virtual-task strategy was demonstrated by embedding it into an existing algorithm and comparing it to the original version of the existing algorithm. Finally, our generalized optimization framework's efficiency was analyzed by integrating a set of optimization algorithms that were designed for static CARPs, and the constructed algorithms performed significantly better than state-of-the-art algorithms for DCARP.

Therefore, the work in this chapter provides a general optimization framework for solving DCARP instances by adapting existing meta-heuristic algorithms in the context of static CARP and answers the *RQ1*: “How can we benefit from the powerful algorithms existing for CARP in order to better solve DCARP?”. However, our limited understanding of the characteristics of DCARP instances which hinders us to further improve the optimization algorithm. Therefore, in the next chapter, our focus is to investigate the fitness landscape of DCARP instances and answer the *RQ2*: “What makes the DCARP instance hard to solve for optimization algorithms?”.

Chapter Four

Fitness Landscape Analysis for DCARP

This chapter is based on following publications: Tong, Hao, Leandro L. Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao. “A hybrid local search framework for the dynamic capacitated arc routing problem.” In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 139-140. 2021. <https://doi.org/10.1145/3449726.3459450> and Tong, Hao, Leandro L. Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao. “What makes the dynamic capacitated arc routing problem hard to solve: insights from fitness landscape analysis.” In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 305-313. 2022. <https://doi.org/10.1145/3512290.3528756>.

4.1 Introduction

Even though there are powerful algorithms for solving general DCARP instances, existing literature still lacks information about which dynamic factors are likely to cause DCARP instances to be harder to solve. This seriously hinders the ability to improve the performance of DCARP algorithms. Therefore, in this chapter, we focus on investigating the effect of different dynamic factors on the characteristics of DCARP instances by fitness landscape analysis. Our main goal is to analyze the basic fitness landscape of a DCARP instance and compare the landscape's features between different DCARP instances produced by various dynamic factors.

Therefore, we are going to answer the *RQ2*: “*What makes the DCARP instance hard to solve for optimization algorithms?*”. We provide the following novel contributions:

- The first systematic fitness landscape analysis for DCARP is performed in this chapter, including several local and global optima related features. The analysis is based on the hybrid local search algorithm (HyLS) [118]. Multiple landscape properties are compared between instances generated with different dynamic events to investigate what factors make DCARP instances easier or harder to solve.
- We show that new tasks are the main factor influencing the difficulty of the instances, and instances requiring additional vehicles for the remaining tasks are harder to solve than instances with other features. These insights on the fitness landscape can potentially contribute towards improving the performance of algorithms in the future.

The remainder of this chapter is organized as follows. Section 4.2 introduces the background on fitness landscape analysis. Section 4.3 provides details on the local search algorithm used in our analysis, here, HyLS. Section 4.4 presents the empirical results and the

analysis of fitness landscape from several different perspectives. Section 4.5 concludes this chapter and points out future work.

4.2 Fitness Landscape Analysis Approaches

Fitness landscape analysis is a popular approach for a better understanding of the characteristics of problem instances and an improved selection of suitable algorithms for the specific problem [67]. A fitness landscape is defined by three elements, including a solution set X , a neighborhood structure V , and a fitness function f . A fitness landscape (X, V, f) is determined by the fitness of the solutions and their corresponding neighbors [50].

Fitness landscape analysis targets to investigate features of optimization problems, such as the fitness distribution in the search space and the structure of optima [67]. In the literature, there are two kinds of empirical techniques proposed for the characterization of these features, namely the random sampling based technique [67, 68] and the local optimal network (LON) based technique [92]. The random sampling based technique performs a search on random solutions to obtain a set of representative solutions, such as potential local optima, in the search space. Then, analysis metrics are computed based on the representative solutions to characterize the target problem. The LON technique usually enumerates all solutions in small problems and formulates the fitness landscape as a network [91]. Each node represents a local optimum in the landscape, and each edge represents the basin-transition [123] or escape edges [122] between two local optima. Then, the properties of LON can directly reflect some characteristics of the problem, such as the number of vertices and density of edges in LON. Moreover, network analysis and visualization tools can also be used to analyze the fitness landscape [92], helping people to better understand the connectivity between local optima. For example, community detection can be used to detect the clustering of local optima [92].

The above two kinds of fitness landscape analysis techniques have been successfully applied to some combinatorial problems. Tavarani-N et al. [115, 116] carried out fitness landscape analysis on four combinatorial optimization problems, based on randomly sampled solutions from the search space and analyzed their local optima-related features and global optima-related features. For example, the correlation length technique operates a random walk in the search space and calculates a correlation coefficient based on the sampled solutions to measure the ruggedness of the fitness landscape [128]. The LON technique was also widely used for analyzing the characteristics of combinatorial problems, such as the flow-shop problem [22], quadratic assignment problem [117] and traveling thief problem [135]. Some studies also proposed sampled LON algorithms, such as Markov-Chain sampling and snowball sampling [117], enabling LON to be adopted for larger problems.

In this thesis, we focused on sampling-based techniques to analyze DCARP instances, as they are simple yet effective techniques to conduct fitness landscape analysis [115].

4.3 Hybrid Local Search Algorithm

The fitness landscape is defined specifically for a neighborhood structure such that one problem instance can have different fitness landscapes for different neighborhood structures. In this thesis, we focused on a hybrid local search algorithm (HyLS) proposed in [118] for landscape analysis. Therefore, the local optima related landscape features analyzed in this thesis are highly related to HyLS, which goes beyond a general analysis based on simple neighborhood moves.

Our proposed HyLS framework (Algorithm 4) can find an improved solution for DCARP with a fast response time by removing the crossover operators existing in meta-heuristic algorithms and promoting the neighborhood of the current solutions. It adopts three effec-

tive strategies including parallel neighborhood moves, an archive and exhaustive one-step moves. The *parallel neighborhood moves strategy* allows for an efficient use of multiple parallel processes where each slave process computes one neighborhood move on one solution independently. Five neighborhood moves (Single Insertion, Double Insertion, Swap, Reverse and Cross [52]) are adopted. The master process collects the improved solutions potentially leading to the optimal solution and maintains an archive to store them (*archive strategy*). For each solution in the archive, the *exhaustive one-step move strategy* is utilized, which optimizes a solution until no further neighborhood move improves the solution quality, to find the best neighbor for the solution, thus focusing on the local area. The framework terminates once the optimization exceeds the time limitation or all solutions in the archive have been optimized.

Algorithm 4: The hybrid local search framework

Input: The updated graph data; Stop locations of outside vehicles; Remaining capacities of outside vehicles;

```

1 Re-construct the solution  $S_0$  with explicit routes;
2 Initialize the solution archive (set)  $SA = \{S_0\}$ ;
3 Set global best solution  $S_{gb} \leftarrow S_0$ ;
4 for each solution  $S_i$  in  $SA$  do
5   Local best solution  $S_{lb} \leftarrow S_i$ ;
6   while true do
7     // Parallel for loop
8     for each neighborhood move  $Move_j$  do
9       Solution  $S_{mj} = OneStepMove_j(S_{lb})$ ;
10      if improve and archive is not full then
11         $SA \leftarrow SA \cup S_{mj}$ ;
12      Update local best solution  $S_{lb}$  based on  $S_{mj}$ ;
13      if not improve or exceed time limitation then
14        break;
15  if  $S_{lb}.cost < S_{gb}.cost$  then
16     $S_{gb} \leftarrow S_{lb}$ ;
17  if exceed time limitation then
18    break;

```

Output: The global best solution S_{gb}

Therefore, we used it to find the local optimum for each sample in the landscape, and neighborhood moves in the HyLS represent neighborhood functions. The size of solution archive (SA) in our work is set to 400, which is determined by our preliminary investigations on tested instances.

4.4 Fitness Landscape Analysis on DCARP instances

4.4.1 Instance Generation

The deployed CARP solution is likely to be affected by a series of dynamic events, and these dynamic events will generate DCARP instances with different characteristics. To compare the influence of different dynamic events, all DCARP instances are generated from the same initial static CARP instance and the same corresponding initial solution obtained by using a powerful meta-heuristic algorithm [113]. The *egl-E2-A* instance was selected as the basic static instance because it is the easiest instance among all static instances in the *egl* dataset, which has more tasks than other datasets [113]. An easier initial instance is likely to make the effect of different settings of dynamic events more obvious.

Four dynamic events which potentially affect the current CARP solution's quality are considered in this work including *cost increase*(CI), *cost decrease*(CD), *new demand*(ND) and *new tasks*(NT). The influence of these dynamic events on the CARP solution is as follows:

- *Cost increase*: The cost of arcs belonging to the solution's path increases deteriorating the current solution.
- *Cost decrease*: The current solution's quality can potentially be improved when the

cost of arcs which are not in the solution's path decreases.

- *New demand*: New demand in the current tasks potentially makes the current solution infeasible due to limited vehicles' capacities.
- *New tasks*: When there are new tasks, the current plan needs to be rescheduled to accommodate the new tasks.

Each dynamic event corresponds to one or multiple factors. We investigated two levels for each factor in this paper, which are presented in Table 4.1. When a dynamic event happens, a few outside vehicles are located at various positions with different remaining capacities. Thus, each DCARP instance will have two state factors, namely *number of outside vehicles* and *value of remaining capacities for each outside vehicle*, which are also investigated in our work. Their levels are presented at the bottom of Table 4.1.

For cost increase/decrease events, the two levels denote that we increase/decrease the costs of 20% (few) and 80% (many) of arcs in/not in the solution's path change. The arcs to suffer the change are selected uniformly at random and their cost is doubled/halved. In the new demand event, we selected uniformly at random 20% (few) and 80% (many) tasks among all remaining tasks to be changed. For the new task event, we also selected uniformly at random 20% (few) and 80% (many) of all available arcs which are not the remaining tasks as the new tasks. The value of new demand and the demand of new tasks are determined by the number of new vehicles required. If the value of new demand is set as small/large, we set that the total demands added to this instance one/four times of the vehicle's full capacity, with the new demand added to each task being split equally among all tasks. For new task events, the positions of new tasks are also considered including *far from* and *close to* the depot. In the following subsections, we will use '0' and '1' to represent the first and second level for each factor.

Table 4.1: Dynamic factors and state factors investigated in this work

Dynamic Events	Factors	Acronyms	Levels	#Settings
Cost Increase	Number of arcs	CI-N	[few, many]	2
Cost Decrease	Number of arcs	CD-N	[few, many]	2
New Demand	Number of changed tasks	ND-N	[few, many]	4
	Value of new demand	ND-V	[small, large]	
New Tasks	Postion of new tasks	NT-P	[close, far]	8
	Number of new tasks	NT-N	[few, many]	
	Demand of new tasks	NT-D	[small, large]	
State Factors		Acronyms	Levels	#Settings
Number of outside vehicles		OV	[few, many]	4
Value of remaining capacities		RQ	[small, large]	

For simplicity, each dynamic event was investigated independently in our experiment. Each DCARP instance was generated based on one dynamic event. Given a dynamic event, a DCARP instance configuration was created for each possible combination of levels for that dynamic event and levels for the state factors. Due to the page limitation, we put the pseudo-code of the instance generator in the supplementary file. The column “#Settings” in Table 4.1 presents the number of settings. Therefore, we generated DCARP instances with 64 (i.e., $(2 + 2 + 4 + 8) * 4$) different configurations of dynamic events and state factors. Moreover, 10 different DCARP instances were generated for each configuration in our experiment. The generated instances and the source code of the presented work is available at Github¹.

4.4.2 Auto-Correlation

The auto-correlation measures the local ruggedness of the fitness landscape based on random-walk sampling [128]. It calculates the expected correlation between the first $T - \tau$ and

¹<https://github.com/HawkTom/DCARP-FLA/>

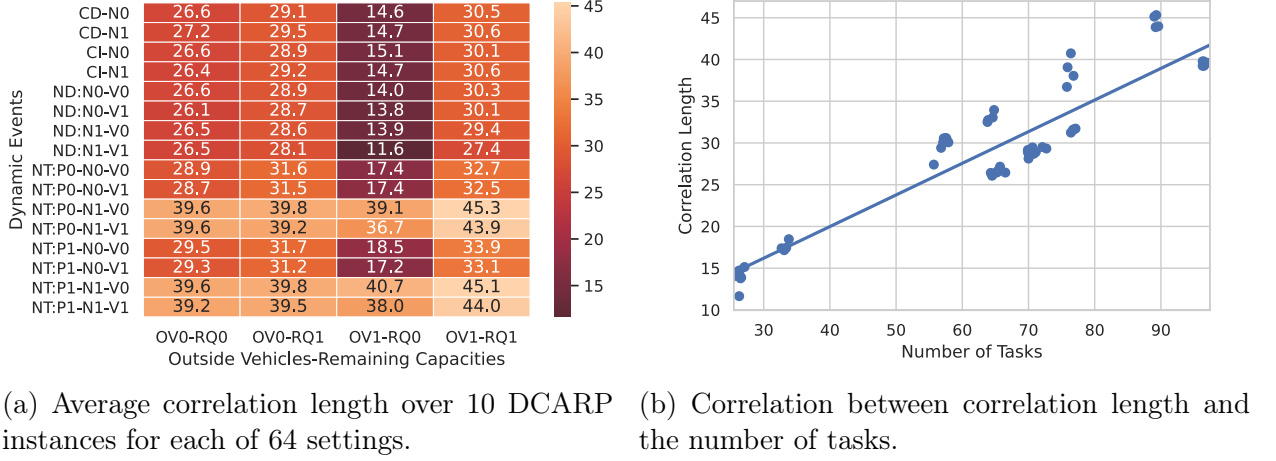


Figure 4.1: Landscape analysis results in terms of correlation length.

last $T - \tau$ subsequences of a fitness sequence $\{c_1, c_2, \dots, c_T\}$ corresponding to the solutions obtained by the random walk. The auto-correlation is estimated according to:

$$R(\tau) = \frac{\sum_{t=1}^{T-\tau} (c_t - \bar{c})(c_{t+\tau} - \bar{c})}{\sum_{t=1}^T (c_t - \bar{c})^2}$$

where $\bar{c}$ is the mean fitness of the whole sequence. The correlation length l is a single value reflecting the ruggedness of the fitness landscape measuring the auto-correlation when $\tau = 1$ [111] and is calculated by:

$$l = -\frac{1}{\ln R(1)}$$

Due to space limits, here, we present the average correlation length over 10 DCARP instances for each setting (Figure 4.1a), where a smaller value indicates a much more rugged landscape. We performed 10e9 steps of random walk on each instance in our experiment.

From Figure 4.1a, for state factors, instances with more outside vehicles and fewer remaining capacities (OV1-RQ0) have the smallest correlation length, while the other three have similar correlation lengths to each other. For dynamic events, instances with new tasks have a larger correlation length, especially for settings with more new tasks (NT:P*-N1-V*), while the other three events have similar correlation lengths to each other. The main reason is the

number of tasks in each instance. States with more outside vehicles and fewer remaining capacities indicate that many tasks have been served and only a few tasks remain to be served. In contrast, new task events obviously will cause the instance to have a large number of tasks. For instances with more tasks, the total cost is a relatively large value so that the cost change of one-step random walk has less effect on the cost of solutions. Thus, instances with more tasks will have a large correlation length. To further confirm the relationship between correlation length and number of tasks, we plot the correlation length and the number of tasks of all instances in Figure 4.1b and applied linear regression. It clearly demonstrates that they are approximately linearly correlated. Therefore, the conclusion motivates us that the algorithm designed for DCARP should focus much more on the number of tasks in the DCARP instances.

It is worth noting that even though the landscape becomes smoother for instances with more tasks, there is no indication that these instances are much easier because the ruggedness measured here just reflects the relative fitness of neighbors. Therefore, an instance with smoother landscape at solution level may also be hard to solve. For example, although an instance has a smooth landscape, it may have some attractive local optimum in the search space causing the instance to become difficult. Therefore, more analysis for DCARP instances are presented in the following subsections.

4.4.3 Local Optima

In this subsection, we randomly sampled 100,000 solutions for each DCARP instance and performed the HyLS on each sampled solution to reach a local optimum. The cost and the solution of all local optima were recorded. The analysis for these local optima is presented in the following.

4.4.3.1 Number of Local Optima:

Larger numbers of local optima can make the combinatorial optimization problem difficult because the optimization algorithm may be easily trapped in local optima [115]. In a landscape, if two solutions can converge to the same local optimum, they must be in the same local optimum's basin. Therefore, the number of local optima is related to the basin size of the local optima. Thus, we calculate the average number of local optima for each setting (Figure 4.2).

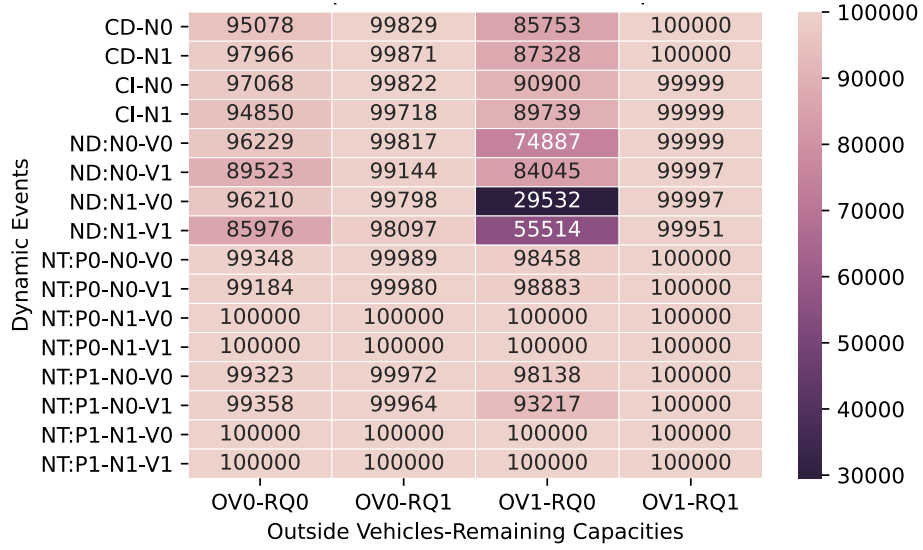


Figure 4.2: Number of local optima over 10 DCARP instances for each of 64 settings.

From Figure 4.2, instances with new tasks (NT) have more local optima than the other three dynamic events. In addition, the influence of the state factors is highlighted when the number of tasks becomes smaller as shown in instances without new tasks (CI, CD, ND) in Figure 4.2 ($N_T < 70$ in our results). Instances with many outside vehicles and lower remaining capacities (OV1-RQ0) have the fewest local optima, indicating that the basin size of these instances might become large. In Figure 4.2, instances that only add demand to many existing tasks when there are many outside vehicles and lower remaining capacities (ND:N1-V*, OV1-RQ0) have a significantly smaller number of local optima. Therefore, from the perspective of the number of local optima, the new tasks are also the main factor which

make instances much harder to solve. If there is no new task, the adding demand event can make an instance even easier to solve.

4.4.3.2 Time to Reach the Local Optimum:

The time for a solution to reach the local optimum reflects the depth of basin of attraction² of the corresponding local optimum. If the solution takes a long time to reach the local optimum, it indicates that this local optimum has a deep basin of attraction. A deep basin of attraction usually makes the optimization problem more difficult because the algorithm will take more time to reach the local optimum. Therefore, we recorded the average time to reach the local optimum by HyLS for all 100,000 solutions. The mean results over 10 DCARP instances for each setting are presented in Figure 4.3.

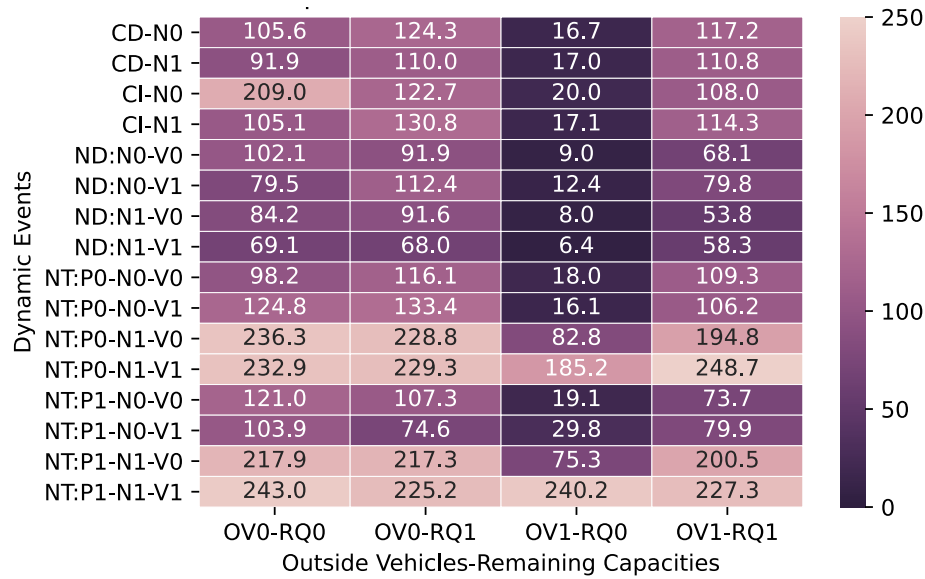


Figure 4.3: Average time over all 100000 solutions to reach the corresponding local optima (unit: s). The presented result for each configuration is the mean result over 10 DCARP instances.

Instances with a much deeper basin of attraction require algorithms to spend more compu-

²The basin of a local optimum corresponds to the set of solutions from which the local optimum can be attained using basic moves/local search techniques.

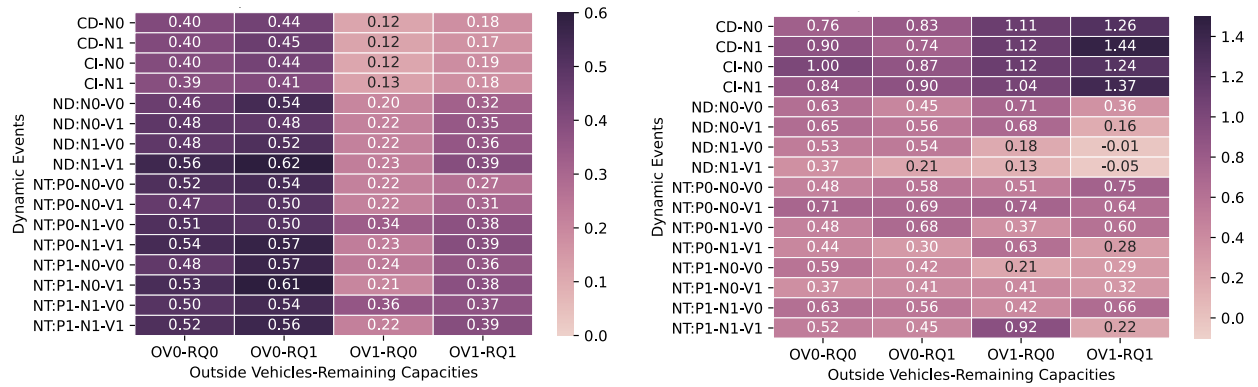
tational resources for exploitation. For these instances, the number of local optima searched within limited computational resources will be very small, as the algorithm is likely to take a longer time to exploit one local optimum. Therefore, such problem instances are more likely hard to solve. From Figure 4.3, the new task event is still the main event that influences the difficulty of the problem. Specifically, instances with more new tasks take a longer time to reach the local optima. The positions and demand value of new tasks have no significant influence on the results.

4.4.3.3 Cost Distribution:

The probability of visiting a local optimum is an important metric to analyze the difficulty of a problem instance. For an easy problem instance, the cost distribution will be more biased to a high fitness so that the probability of finding a high-quality local optimum will be much higher. The fitness distribution is capable of well reflecting such characteristics. Therefore, we analyzed the cost distribution based on all recorded local optima. For a much fairer comparison, we normalized the cost of each DCARP instance by its expected cost, which is estimated by sampling a large number of random solutions from the search space.

Two statistical measures are applied to compare the cost distribution between DCARP instances with different configurations, including the mean cost and skewness. The mean cost reflects the expected quality of a local optimum obtained by HyLS, and the skewness reflects the shape of the distribution compared with a normal distribution. A positive skewness value indicates that the bulk of costs lies to the left of the mean cost. Therefore, a lower mean cost and a higher skewness indicates that the probability of finding a high-quality local optimum is higher. The average values of these two statistics over 10 DCARP instances for each of the 64 settings are presented in Figure 4.4a and Figure 4.4b.

From Figure 4.4a, the values in the left two columns are larger than the ones in the



(a) Averaged mean value of all reached local optima for each DCARP setting over 10 DCARP instances

(b) Averaged skewness value of all reached local optima for each DCARP setting over 10 DCARP instances.

Figure 4.4: Landscape analysis results in terms of cost distribution.

right two columns. We conclude that instances with more outside vehicles (OV1) are much easier to solve. For the same dynamic event, instances with many outside vehicles and small remaining capacities (OV1-RQ0) are the easiest because the number of tasks is the smallest. Moreover, for the same state factor, adding new tasks makes instances harder than the other three events. The events of cost increase and decrease are similar to each other indicating that instances with only these two events will not become very difficult. The skewness presented in Figure 4.4b is able to reflect the probability of finding a local optimum in an instance. A much larger value indicates that it has a high probability for finding a high-quality solution. From Figure 4.4b, almost all instances have a positive skewness value, indicating that the quality of most reached local optima is much better than a random solution. This is also helpful for the effectiveness of HyLS. Specifically, the distribution of instances only with cost change events is more biased to the area with high-quality solutions meaning that an algorithm may find a good solution with fewer computing resources.

In summary, according to the mean cost and the skewness of cost distribution, we can conclude that cost change events will not make DCARP instances harder to solve compared with new demand and new tasks events. A higher number of outside vehicles will also result

in a higher mean quality of local optima to the DCARP instances.

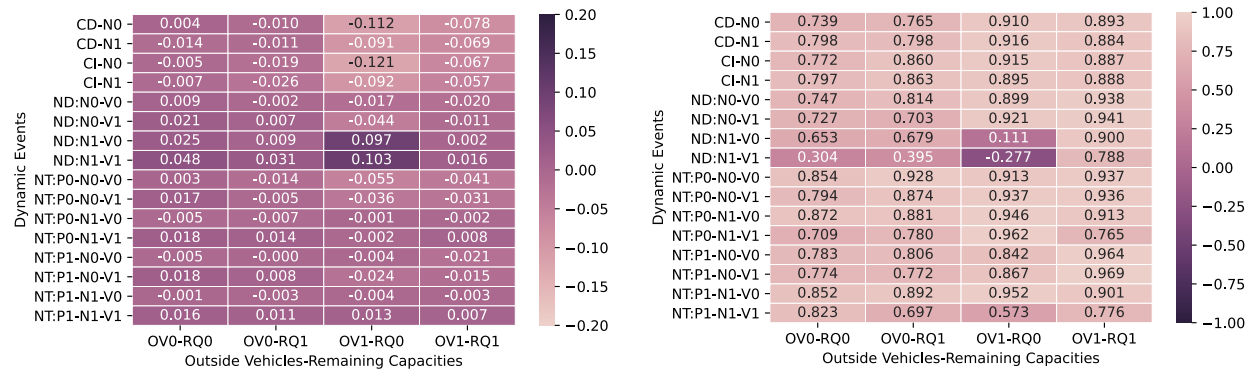
4.4.3.4 Distance Between Local Optima:

The distance between local optima is also an important property of the fitness landscape. If a local optimum with a low cost is far from other local optima, it will be hard for the algorithm to explore from a local optimum to a much better local optimum which makes the problem hard to optimize. Therefore, we analyzed the following two distance properties:

1. The distance between local optima versus expected distance between two random solutions.
2. The correlation between the cost of local optima and their distance.

Unlike for continuous optimization problems, the Euclidean distance is not suitable for the CARP space. We applied Hamming distance to measure the distance between two CARP solutions. Suppose V_R is a set of vertices including the depot and all vertices incident with at least one task, a complete graph $G_R = (V_R, A_R)$ can be constructed, where A_R is the set of arcs between each pair of vertices in V_R . If one vertex v_i incidents with $d(v_i)$ tasks ($d(v_i) > 0$), we will duplicate $d(v_i)$ copies of this vertex in V_R and assign a unique id to each copy. The cost of an arc in A_R depends on the original (D)CARP instance. Therefore, all traversed paths in a CARP solution are arcs in G_R , and no arc in G_R will be used twice in a CARP solution. Consequently, a CARP solution can be represented by a binary vector in the space of arcs in G_R , where each bit denotes one arc in G_R and “1” denotes that the corresponding arc is used in the CARP solution. Thus, solutions will be converted to the binary representation first before calculating the Hamming distance.

The ratios of the expected distance between two local optima to that of two random solutions for all 64 settings are presented in Figure 4.5a, where each result is the averaged



(a) Ratio of the expected distance between two local optima to that of two random solutions. The presented result for each configuration is the averaged value over 10 DCARP instances. (b) Correlation between the cost of local optima and their distance. The presented results for each configuration are the averaged values over 10 DCARP instances.

Figure 4.5: Landscape analysis results in terms of distance between local optima.

value over 10 DCARP instances. A negative value indicates that the distance between two local optima is smaller than two random solutions. In Figure 4.5a, almost all ratios are very close to 0, reflecting that local optima in all DCARP instances are uniformly distributed instead of gathering in a specific area or far from each other. We conclude that all dynamic factors will not impact the distribution of the local optima, not further influencing the difficulty of each instance.

The correlation between the cost of local optima and their distance for all 64 settings are presented in Figure 4.5b, where each result is also the averaged value over 10 DCARP instances. From Figure 4.5b, almost all correlations between the cost of local optima and their distance are close to 1. This indicates that better local optima are much closer to each other than the less fit local optima, which is beneficial to the optimization. Moreover, instances with more outside vehicles (OV1) have larger correlation values according to Figure 4.5b whereas the correlations for different dynamic factors under the same state factor are similar to each other. Therefore, instances with more outside vehicles might be slightly easier for optimization. Instances with adding demands to many tasks (ND:N1-V*) have the smallest correlation. This indicates that the distribution of local optima is not clustered

based on their cost. Thus, the algorithm might require a bigger step for the exploration.

4.4.4 Approximate Global Optimum

In this subsection, we focus on the properties of the global optimum of the problem instance. As we cannot enumerate all samples in the search space in our DCARP instances, the real global optimum is unknown. Therefore, we applied a powerful meta-heuristic, i.e., Memetic Algorithm with Extended Neighborhood Search (MAENS) [113, 119], to optimize all DCARP instances with sufficient computational resources, i.e., with the same maximum iteration number as used in the static optimization [113]. The obtained best solution is then considered as the approximate global optimum because the real global optimum is unknown.

4.4.4.1 Fitness Distance Correlation:

Fitness Distance Correlation (FDC) is also a useful measurement to characterize the problem difficulty, measuring the correlation of a solution's fitness and its distance to a global optimum [50, 51]. In minimization problems, a positive correlation indicates that the closer a solution distance is to the global optimum, the higher quality (i.e., lower cost) the solution has. The algorithm can benefit from such a fitness landscape and be more easily guided to the global optimum. In contrast, an instance with a negative correlation will potentially guide the algorithm towards the opposite direction of the global optimum. For DCARP instances, suppose $F = \{c_1, c_2, \dots, c_n\}$ and $D = \{d_1, d_2, \dots, d_n\}$ are the fitnesses of n solutions and the distances to the global optimum of these solutions. FDC is calculated by the following equation [50]:

$$FDC = \frac{\frac{1}{n} \sum_{i=1}^n (c_i - \bar{c})(d_i - \bar{d})}{\sigma_F \sigma_D}$$

where $\sigma_F, \sigma_D, \bar{c}, \bar{d}$ represent the standard deviations and means of F and D . The distance between two solutions are also measured by the Hamming distance as introduced in Section 4.4.3.4. The mean FDCs over 10 DCARP of each configuration is presented in Figure 4.6. For each instance, all found local optima were used to calculate the FDC in our analysis.

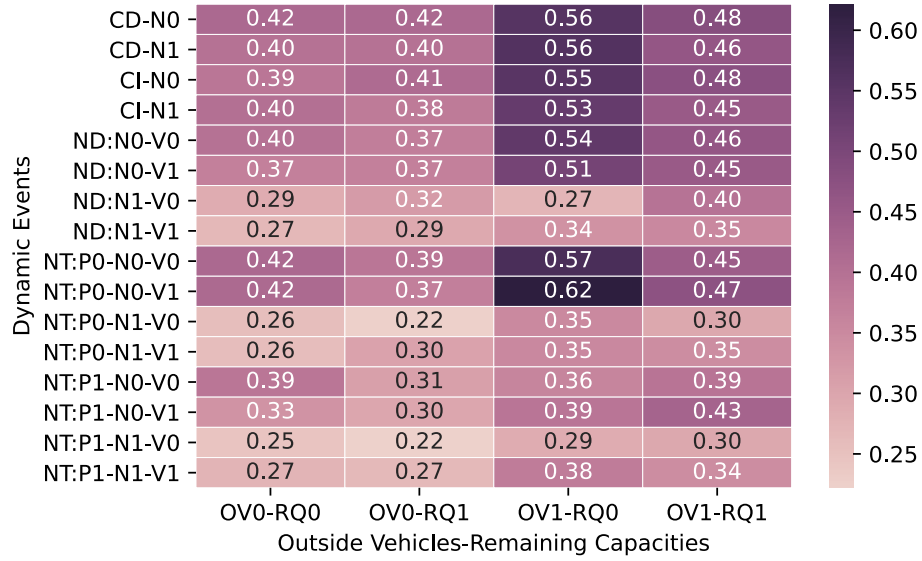


Figure 4.6: Fitness distance correlation of global optimum for all 64 settings. The presented result for each configuration is the mean result over 10 DCARP instances.

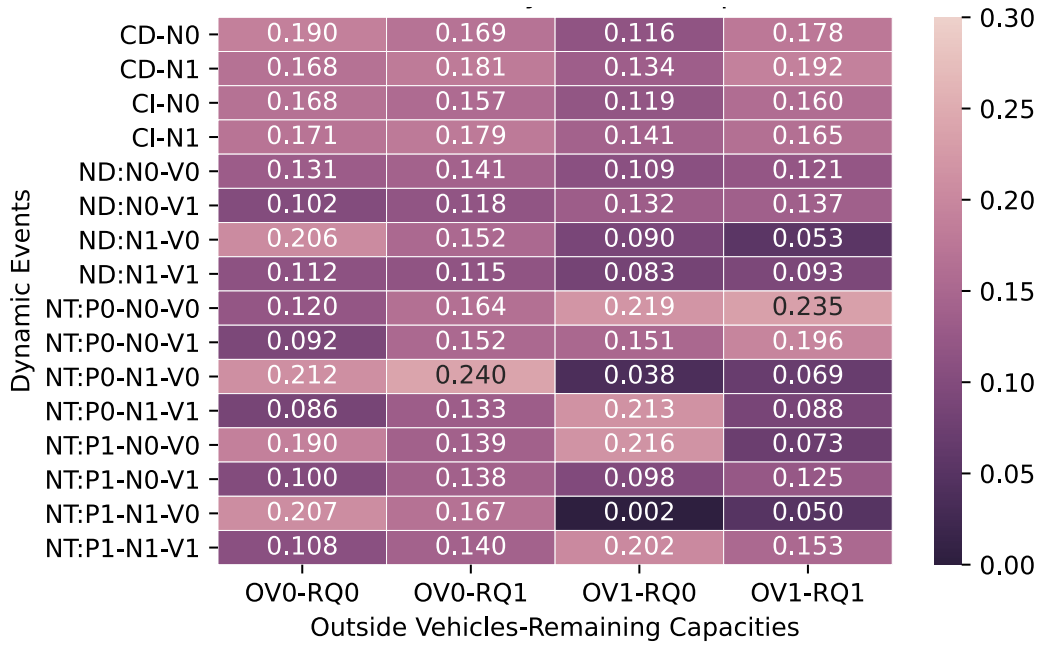
A high value in Figure 4.6 indicates that local optima that are closer to the global optima have higher qualities, meaning that the instance is potentially easier to solve. In Figure 4.6, instances with new tasks (NT) have a relatively small FDC value compared with the other three events. For state factors, FDC values of instances with fewer outside vehicles (OV0) are also smaller than those with many outside vehicles (OV1). We also observe that instances with adding demand to many tasks (ND:N1-V*) also have small FDC values. These instances all require more vehicles to serve the remaining tasks. Therefore, a potential conclusion is that when remaining tasks require more new vehicles, the instance will have a smaller FDC value, and the problem instance might become harder to solve.

4.4.4.2 Return Probability:

The final characteristic we investigate is the return probability of the global optimum. The return probability empirically measures the basin of attraction of the global optimum by calculating the probability of returning to the global optimum starting from a solution with a fixed distance to it [115]. If the global optimum has a high return probability even though the starting solution is far from the global optimum, it may be easier for the algorithm to find the global optimum.

In our experiment, the return probability is measured by starting from solutions which are obtained by applying a given number of *Single Insertion (SI)* operations. It is the most basic move for the CARP solution that move one task from one route to another position of this or another route in the solution. The return probability is estimated by starting from 1000 solutions generated by applying the same number of steps of *SI* from the global optimum. From our results, the return probability in all DCARP instances drops to 0 after about 6 steps of *SI*. For simplicity, we only present the return probability of the global optimum after one random step of *SI* averaged over 10 instances in Figure 4.7.

From Figure 4.7, the return probability for one-step neighborhood move was very small for all DCARP instances, which indicates that the global optimum's basin size is very small for the DCARP problem. Therefore, the DCARP instances are hard to solve, and a good algorithm should have a good exploration ability to find a high-quality solution. In addition, the bottom right part of Figure 4.7 has a relatively smaller return probability compared with other parts. In this area, instances have more remaining demands to be served and many outside vehicles. Therefore, the dynamic events causing more demands and the state with more outside vehicles are likely more challenging to solve.

Figure 4.7: Return probability of global optimum with one random step of *Single Insertion*.

4.5 Summary

In this chapter, we focused on investigating which factor(s) potentially make the DCARP instances hard to solve. We considered four common dynamic events: cost increase, cost decrease, new demand, and new tasks as well as two inherent state factors of the DCARP instance, i.e., the number of outside vehicles and the remaining capacities. Fitness landscape analysis techniques based on a local search algorithm, i.e., HyLS, were performed on a set of generated DCARP instances according to different configurations of dynamic and state factors. We investigated the correlation length, the number of local optima, the time to reach a local optimum, the mean cost of the local optima, the distance properties of local optima, the fitness distance correlation, and the return probability on all generated DCARP instances.

Based on our empirical results from the fitness landscape analysis, a list of valuable conclusions are obtained as follows:

- DCARP instances with many remaining tasks and new tasks caused by the new task event had a smoother fitness landscape at solution level.
- The task's number was also the main factor affecting the number of local optima and the averaged time to reach the local optima. Instances with more tasks caused more local optima and consequently a longer time to reach the local optima.
- The local optimum's mean cost of DCARP instances was mainly affected by the number of outside vehicles and the number of tasks. Instances with more outside vehicles and fewer tasks had a relatively lower mean cost.
- The local optima in all DCARP instances were almost uniformly distributed. Their distances were similar to that of two random solutions.
- Local optima with higher quality were closer to each other than to less fit local optima for all DCARP instances.
- In terms of the global optimum, instances which require more vehicles for the remaining tasks had a smaller fitness distance correlation as well as a smaller return probability.

In summary, the new task event had much more influence on the difficulty of the DCARP instances than the other three dynamic events. The number of remaining tasks was the main factor making DCARP instances hard to solve, with a larger number being harder. There was no evidence that the cost increase and cost decrease significantly impacted the characteristics of the DCARP instances. In addition, our fitness distance correlation analysis found that instances requiring more new vehicles are likely more difficult.

Therefore, our insights from the fitness landscape analysis address the *RQ2*: “*What makes the DCARP instance hard to solve for optimization algorithms?*”. Based on our understandings of the fitness landscape analysis for DCARP instances, a potential direction about designing more effective algorithms for optimizing DCARP is to pay more attention to the

new tasks. For example, for a new DCARP instance, we can focus on designing strategies to allocate new tasks. Therefore, in the next chapter, we consider DCARP scenarios with different settings based on whether they include new tasks or not and design effective algorithms in conjunction with optimization experience, separately. It answers the *RQ3*: “How to extract optimization experience from a DCARP instance and integrate into a meta-heuristic framework for enhanced performance in different DCARP scenarios?”.

Chapter Five

Benefiting from Experience through Building Blocks

This chapter is based on following publication: *Tong, Hao, Leandro L. Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao. “Building blocks as experiences in dynamic capacitated arc routing problems.” In submission to CAAI Transactions on Intelligence Technology.*

5.1 Introduction

A DCARP scenario is composed of a series of discrete DCARP instances. A new DCARP instance is generated based on the preceding DCARP instance and a partially executed solution. As a result, a new DCARP instance in a DCARP scenario shares similarities with its preceding DCARP instance. Intuitively, the optimization experience and solution information of the former DCARP instance could be extracted to promote the optimization for the new DCARP instance in a DCARP scenario. However, current existing works do not consider and utilize such optimization experience [43, 60, 87, 88, 118], wasting valuable knowledge that could be used to enhance the optimization process for DCARP instances. When new DCARP instances generate once dynamic events happen, the solutions from the previous optimization process become feasible for the new DCARP instance. Consequently, these existing works always optimize the new DCARP instance from scratch to obtain new feasible solutions to improve the current service schedule. Therefore, to extract useful optimization experience from the former DCARP instances and enhance the optimization for new DCARP instances, in this chapter, we propose a solution building block adaptation strategy and incorporate it into a dynamic optimization framework able to transfer valuable optimization experience for the DCARP scenario.

Therefore, we are going to answer the *RQ3*: “How to extract optimization experience from a DCARP instance and integrate into a meta-heuristic framework for enhanced performance in different DCARP scenarios?”. Our contributions are as follows:

- We propose a new DCARP optimization framework that benefits from experience through knowledge transfer in the DCARP scenario to promote optimization of DCARP instances.
- We propose the new concept of *building block* for extracting optimization experience

contained in historical solutions. A building-block-based constructive heuristic is proposed to handle DCARP scenarios with cost-changing and task-changing dynamic events. Additionally, our solution building block adaptation strategy includes an insertion heuristic to deal with new tasks.

- We evaluate the proposed Solution Building Block Adaptation (SBBA) strategy progressively considering dynamic events from only cost-changing dynamic events (OC) to both cost-changing and task-changing dynamic events (CT). In particular, we consider two different types of DCARP scenarios, i.e., DCARP-OC and DCARP-CT scenarios. Our analysis shows that the adapted historical solutions are significantly better than the solutions generated by conventional constructive heuristics when dynamic events are small and the high-quality solutions for the new environment are similar to those of the previous environment.
- We embed two state-of-the-art DCARP optimization algorithms into the proposed SBBA based dynamic optimization framework (DO-SBBA), and evaluate them on the generated DCARP scenarios. The empirical results demonstrate that our DO-SBBA framework significantly improves the performance of the original algorithms on these DCARP-OC and DCARP-CT scenarios, and is particularly beneficial on the DCARP-OC scenarios.
- The robustness of the proposed DO-SBBA framework was also investigated on DCARP instances with varying severity of dynamic changes. The empirical results show that our DO-SBBA framework was particularly effective in DCARP-OC scenarios with various severity of dynamic changes and DCARP-CT scenarios without a large change in the number of tasks.

The remainder of this chapter is organized as follows. Section 5.2 introduces the proposed DO-SBBA framework and the details of SBBA strategy. Section 5.3 presents experimental

studies of evaluating the performance of DO-SBBA. Summaries are provided in Section 5.4.

5.2 Solution Building Block Strategy based Dynamic Optimization Framework

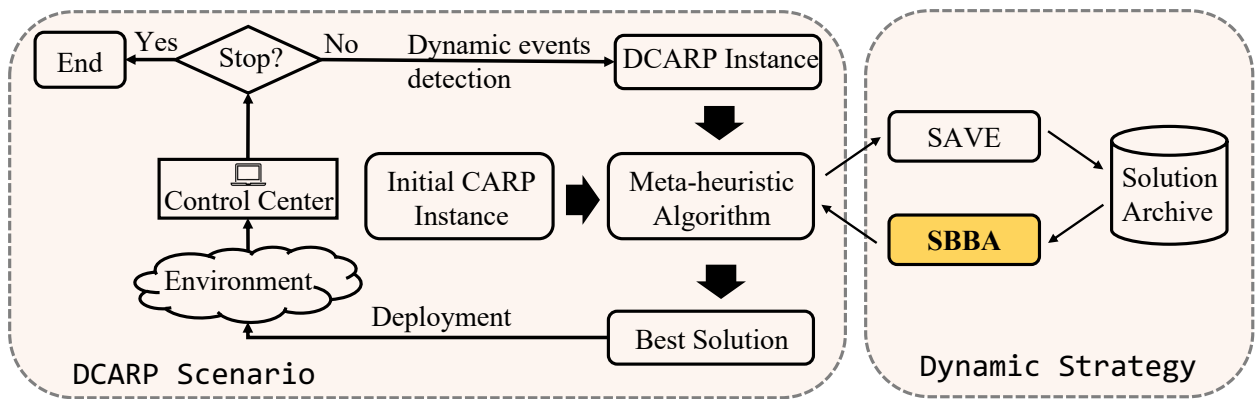


Figure 5.1: The structure of the SBBA-based dynamic optimization framework for the DCARP scenario. The left part is an illustration of the DCARP scenario, where dynamic optimization process starts with an initial static CARP instance and the control center will detect changes in the environment and determine to generate DCARP instances or not. The dynamic strategy at the right is used to assist the optimization for DCARP instances.

A dynamic optimization framework based on a novel solution building block adaptive (SBBA) strategy is proposed in this section. Its structure is presented in Figure 5.1.

The dynamic optimization process starts with an initial static CARP instance. A meta-heuristic optimization algorithm can be employed to obtain the best solution for the static CARP instance, which will be used to schedule vehicles to serve the pre-defined tasks. Subsequently, a control center will detect changes in the environment [85]. Whenever any dynamic events occur while vehicles are in service, the control center will report the type of dynamic events and generate a new DCARP instance according to the occurred dynamic events. The meta-heuristic algorithm will then dynamically optimize the newly generated DCARP instance. After the dynamic optimization process for the new DCARP instance, an updated

solution will be obtained and deployed to the environment to update the service plan of all vehicles. The dynamic optimization will stop only when all tasks have been served.

In our framework, the SBBA strategy adapts the historical solutions and uses them as initial solutions for the meta-heuristic algorithm. Since different types of dynamic events can result in DCARP instances with varying degrees of difficulty to solve as concluded in Chapter 4, we consider two different types of DCARP instances in our work:

1. DCARP instances with only cost-changing events (DCARP-OC). Cost-changing dynamic events include the increase and decrease of road deadheading costs.
2. DCARP instances with both cost-changing and task-changing dynamic events (DCARP-CT). Task-changing dynamic events include added and deleted tasks.

To handle these different types of DCARP instances, specific strategies in SBBA are applied to adapt the historical solutions for promoting dynamic optimization. All these strategies are based on the new concept of *building block*. Therefore, in the following subsections, we will introduce the concept of the building block first and then provide details on the specific strategies.

5.2.1 Building Blocks as Experiences

The new DCARP instance is generated based on the former DCARP instance and the deployed solution. However, except for the best solution obtained in the optimization process for the former DCARP instance, historical high-quality solutions cannot be used directly to schedule the vehicles' service plans in the new DCARP instance. Nevertheless, the information contained in these solutions can be extracted and reused. For instance, it is highly likely that two adjacent tasks in high-quality historical solutions will still be neighbors in the

best solution of the new DCARP instance. Therefore, such sequence patterns contained in historical solutions can be extracted to assist in constructing feasible high-quality solutions for the new DCARP instance.

Algorithm 5: The pseudo-code of constructing building blocks.

Input: The historical solution: S ; The task set E_R

```

1 Function ConstructBuildingBlocks( $S, E_R$ )
2   Initialize an empty set for building blocks  $BLOCK = \emptyset$ .
3   Initialize a building block:  $bb$ , where  $bb.seq = []$ .
4   for each route  $r_k \in S$  do
5     for  $i$  from 1 to  $l_k$  do
6       if  $t_{k,i} \notin E_R$  then
7         if  $bb.seq$  is not empty then
8           Set  $bb.head = head_{bb.seq[0]}$ .
9           Set  $bb.tail = tail_{bb.seq[-1]}$ .
10          Set  $bb.cost$  equals to the total serving costs of  $bb.seq$ .
11          Set  $bb.demand$  equals to the total demands of all tasks in  $bb.seq$ .
12           $BLOCK = BLOCK \cup bb$ .
13          Re-initialize  $bb$  that  $bb.seq = []$ .
14        continue;
15      else
16         $bb.seq.append(t_{k,i})$ .
17  return  $BLOCK$ ;

```

Output: The set of building blocks: $BLOCK$

In this work, we propose the concept of the “building block” to describe such sequence patterns. Since the CARP solution is a set of routes, and the route is composed of a sequence of tasks, if a sub-sequence of tasks in the previous CARP solution is still required to be served in the new DCARP instance, we combine this sub-sequence of tasks into a building block. We assume that such a task sequence that exists in the high-quality solutions of the former DCARP instance will still remain in the high-quality solutions of the new DCARP instance.

The pseudo-code for constructing building blocks from a historical solution is presented in Algorithm 5. It receives as input a historical solution and the set of tasks to be served in the new DCARP instance. Each building block has five properties, including the block’s task

sequence, head node, tail node, total serving costs, and total demands. In each historical solution, consecutive sequences of tasks still required to be served in the current DCARP instance are detected and combined into building blocks. The head node and tail node of this building block are the head and tail nodes of this task sequence. The cost and demand of the building block are equal to the total costs of serving this task sequence and the total demands of all tasks in this task sequence. If a task in a route has both its predecessor and successor tasks served or deleted in the new DCARP instance, we also make this independent task a building task. Finally, all detected building blocks form a set of building blocks for the solution construction.

5.2.2 SBBA for DCARP-OC Instances

For DCARP-OC instances, the SBBA strategy generates a new population of solutions for the new DCARP instances from an archive of historical solutions. The pseudo-code of SBBA for DCARP-OC instances is presented in Algorithm 6.

Algorithm 6: SBBA for DCARP-OC instances

Input: The set of archived solutions: AS ; The task set: E_R .

- 1 Initialize an empty solution set $POP = \emptyset$.
- 2 **for** each solution $S_i \in AS$ **do**
- 3 $BLOCK = \text{ConstructBuildingBlocks}(S_i, E_R)$.
- 4 Apply Path-Scanning algorithm to $BLOCK$ and obtain a new solution NS_i .
- 5 **if** $NS_i \notin POP$ **then**
- 6 $POP = POP \cup NS_i$.

Output: Adaptive solution set POP

The archived solutions AS are a set of high-quality solutions saved from the optimization process for the former DCARP instance. Different definitions for “high-quality” could be adopted. In our experiments later on, we consider the top l_{AS} solutions as being high-quality solutions, where l_{AS} is a pre-defined parameter. For each historical solution S_i , we first obtain a set of building blocks using Function $\text{ConstructBuildingBlocks}(S, E_R)$

in Algorithm 5, where E_R is the task set for the new DCARP instance. Then, we apply the path-scanning constructive heuristic to the obtained building blocks to generate a new solution NS_i . If the current adaptive solution set does not contain NS_i , then it is added to the adaptive solution set as one of the adaptive solutions. Finally, the new adaptive solution set generated from archived solutions which belong to the previous DCARP instance is returned, and used to help the solution initialization for new DCARP instance.

5.2.3 SBBA for DCARP-CT Instances

Different from the DCARP-OC instances, DCARP-CT instances require considering changing tasks, especially newly added tasks and how to assign them to existing or new routes. Therefore, a building block based constructive strategy and an insertion strategy are proposed to handle changing tasks in SBBA. The pseudo-codes of SBBA for DCARP-CT instances are presented in Algorithm 7.

First, the newly added tasks are filtered out from all tasks E_R of the current DCARP instance to form a new task set $NE_R \subset E_R$. Then, the building block based constructive and insertion strategies are used to adapt historical solutions to the new environment from different perspectives.

The building block based constructive strategy is similar to SBBA for DCARP-OC instances, which considers building blocks in historical solutions. However, SBBA for DCARP-CT instances also considers each new task as an independent building block. Therefore, both the sequence pattern in historical solutions and the new tasks are considered for generating new solutions. Finally, the path-scanning algorithm is used to generate new solutions.

The insertion strategy considers the complete sequence pattern of the historical solution and inserts the new task into the historical solution to obtain the new solution. For each

Algorithm 7: SBBA for DCARP-CT instances

Input: The set of archived solutions: AS ; The task set: E_R ; The capacity Q .

```

1 Filter the new task set  $NE_R \subset E_R$ .
2 /* Building block based construction */
3 Initialize an empty set of building blocks  $BLOCK = \emptyset$ .
4 Initialize a building block:  $bb$ , where  $bb.seq = []$ .
5 for each task  $tk \in NE_R$  do
6     Set  $bb.seq.append(tk)$ .
7     Set  $bb.head = head_{tk}$ .
8     Set  $bb.tail = tail_{tk}$ .
9     Set  $bb.cost = sc(tk)$ .
10    Set  $bb.demand = dm(tk)$ .
11     $BLOCK = BLOCK \cup bb$ .
12 Initialize an empty solution set  $POP = \emptyset$ .
13 for each solution  $S_i \in AS$  do
14      $BLOCK' = BLOCK \cup \text{ConstructBuildingBlocks}(S_i, E_R)$ .
15     Apply Path-Scanning algorithm to  $BLOCK'$  and obtain a new solution  $NS_i$ .
16     if  $NS_i \notin POP$  then
17          $POP = POP \cup NS_i$ .
18 /* Insertion */
19 Set  $W_T$  as the total demands of all tasks in  $E_R$ .
20 Set  $k_0 = \lceil \frac{W_T}{Q} \rceil$ .
21 for each solution  $S_i \in AS$  do
22     Set a new solution  $NS_i = S_i$ .
23     Remove all served tasks and deleted tasks in  $NS_i$ .
24     Set  $K$  as the number of routes in  $NS_i$ .
25     if  $K < k_0$  then
26         Add  $k_0 - K$  empty routes into  $NS_i$ .
27     Set a copy of new task set  $CNE_R = NE_R$ .
28     while  $|CNE_R| > 0$  do
29          $t, r =$  task and route with the smallest increasing cost for the insertion.
30         if  $t, r = NULL$  then
31             // No route is feasible for any tasks in  $CNE_R$ 
32             Add a new empty route into  $NS_i$ .
33         else
34             Insert  $t$  into  $r$  in  $NS_i$ .
35             Remove  $t$  from  $CNE_R$ .
36     if  $NS_i \notin POP$  then
37          $POP = POP \cup NS_i$ .
```

Output: Adaptive solution set POP

historical solution, tasks that have been served or deleted are first deleted from the solution's task sequence. This results in an incomplete solution that contains information about the historical solution. Since the minimal number of vehicles k_0 can be determined based on the instance's information (i.e., $k_0 = \left\lceil \frac{W_T}{Q} \right\rceil$ where W_T is the total demands of all tasks), we add several empty routes to the current incomplete solution if the number of routes in the current incomplete solution is smaller than the minimum required number of vehicles. Then, the new tasks are inserted into the current incomplete solution to obtain a new feasible solution in a greedy way, such that the task resulting in the smallest increase in solution cost is added first. If no route is feasible for any of the remaining tasks to be inserted, a new empty route is inserted into the current solution, and the greedy insertion process continues.

After using the two different strategies, a new adaptive solution set is obtained to assist the solution initialization for the new DCARP instance.

5.2.4 Discussion

Our work shows that a relatively simple building blocks strategy can be very effective to update solutions to dynamic events under limited optimization time, as will be shown in section 5.3. In particular, it is able to explicitly and efficiently extract valuable information from historical solutions of the former environment, particularly in DCARP-OC scenarios where no new tasks are inserted. However, for DCARP-CT scenarios involving the insertion of new tasks, our approach currently has no optimization experience based mechanism for assisting insert these new tasks to proper positions. As a result, the proposed strategies might be less effective if there are a large number of new tasks. Therefore, it is very valuable to investigate how to extract useful knowledge from past optimization to enhance the insertion of new tasks. The efficacy of proposed strategies will be demonstrated and analyzed in the following experimental section.

5.3 Experimental Studies

In this section, the effectiveness of proposed SBBA strategy are evaluated in a series of different DCARP scenarios. Then, we embed two state-of-the-art meta-heuristic algorithms into our proposed DO-SBBA framework and evaluate their performance in a series of different DCARP scenarios as well. The empirical results are presented and analyzed in this section.

5.3.1 Experimental Settings

All experiments were conducted on a series of DCARP scenarios generated by the simulation system designed in Section 2.3 based on a static CARP benchmark, namely the *EGL* set [56]. The set consists of 22 different static CARP instances, including 10 *EGL-G* instances with 200 initial tasks and 12 *EGL-S* instances with 100 initial tasks. For simplicity, we simplified the proposed simulation system, focusing only on the changes of edges' costs and tasks to benchmark the DCARP-OC and DCARP-CT scenarios.

To simulate cost-changing dynamic events, the expected deadheading cost ($\overline{dc}$) is assigned to each edge in the map to indicate the minimal cost of the edge. When simulating the cost-changing dynamic events, the deadheading cost (dc) of an edge is determined by the following equation:

$$dc = \begin{cases} dc, & p < 0.25 \\ \overline{dc} \times r, & 0.25 \leq p \leq 0.75 \\ \overline{dc}, & p > 0.75 \end{cases} \quad (5.1)$$

where $p \in U(0, 1)$ is a random number drawn from a standard uniform distribution. For each edge in the map, if $p < 0.25$, its cost remains the same, indicating there is no dynamic event on the edge. If $p > 0.75$, the edge's cost becomes $\overline{dc}$. Otherwise, the edge's cost becomes

r times the base cost, where $r \in U(1, C)$ is also a random number drawn from a uniform distribution with an upper bound value of C . In our experiments, C is set to 5.

Then, to simulate task-changing dynamic events, a vanishing probability of $p_v = 0.2$ is assigned to each remaining task when generating new DCARP instances. To simulate newly added tasks, a predefined set of available arcs is used, and N_{NE_R} arcs are randomly selected from the set to become the new tasks for the new environment. The value of N_{NE_R} is determined by the following equation:

$$N_{NE_R} = p_t \times N_{RT}, \quad (5.2)$$

where N_{RT} denotes the number of remaining tasks when dynamic events occur, and p_t is a factor used to control the size of the new task dynamic events. In our experiments, p_t is set to 0.2.

We generated 10 different DCARP scenarios independently for each static CARP instance. In our simulation, after optimization for a DCARP instance with limited time, the best-obtained solution will be used to update the schedule of vehicles. Then, the next DCARP instance will be generated after the current solution has been executed for a random duration. We generated up to 5 DCARP instances in each DCARP scenario. If the number of tasks in a new DCARP instance is less than 20, we also stopped generating new instances.

Besides, to investigate the generalization ability of the proposed SBBA, we also generated a series of DCARP scenarios with varying severity of dynamic changes. Specifically, in the DCARP-OC scenarios, the severity of cost changes is controlled by the parameter C , and we investigate six different settings with $C \in \{1, 3, 5, 10, 15, 20\}$. For the DCARP-CT scenarios, we focused on investigating different numbers of added tasks in our analysis. Thus, we generate six different severity of added tasks with $p_t \in \{0.1, 0.2, 0.4, 0.5, 0.6, 0.8\}$. For simplicity, we only used the static CARP instance *egl - g1 - A* for this analysis. In the

DCARP-CT scenarios, the parameter C for cost changes is set to 5.

In our experiment, two optimization algorithms are embedded into the proposed DO-SBBA framework, including the memetic algorithm with extended neighbor search (MAENS) [113] and the global repair operator-based tabu search algorithm (RTS) [78] embedded with the proposed virtual task strategy. MAENS is a population-based meta-heuristic algorithm and its population size is set as $p_{size} = 30$ which is same as its original paper [113]. The number of archived top solutions l_{AS} is also set as 30 in our experiments. For MAENS, if the number of adaptive solutions is higher than the predefined population size p_{size} , the best p_{size} solutions are selected as the initial solutions. Otherwise, if the number of adaptive solutions is smaller than the predefined population size due to removing repeated solutions, the necessary number of additional solutions is generated by the optimization algorithm's initialization scheme to form a complete population. Conversely, RTS is an individual-based meta-heuristic algorithm and only the best among all adaptive solutions is used, as $p_{size} = 1$.

For each DCARP instance optimized by a meta-heuristic algorithm, the maximum optimization time was set to 1 minute, as it is desirable for optimization results to be provided quickly in real world DCARP problems. We executed 25 independent runs in our experiments. All programs were implemented in C language and ran on a Linux server with an AMD Ryzen Threadripper PRO 3995WX 64-Cores 2.7GHZ processor. All static CARP instances, generated DCARP scenarios, and algorithms' source codes are available online¹.

Upon completion of the optimization process, the best-obtained solution will be employed to update the vehicle schedules. In our simulation, a new DCARP instance is generated after the current solution has been executed for a specified duration.

¹<https://github.com/HawkTom/Solution-Building-Block-Strategy-for-DCARP>

5.3.2 Performance Measurement

Since the generated DCARP instances in a DCARP scenario can vary significantly in scale, we defined a normalized cost (NC) for each DCARP instance using the following equation:

$$NC = \frac{TC - TC_{min}}{TC_{max} - TC_{min}} \quad (5.3)$$

where TC is the cost of the solution that needs to be normalized, and TC_{max} and TC_{min} are the maximum and minimum costs, respectively, used for normalization in the corresponding DCARP instance. The specific values for TC_{max} and TC_{min} depend on the set of experiments, which will be discussed in the next section.

Furthermore, since a DCARP scenario comprises a series of DCARP instances, we have employed an average normalized cost (ANC) for dynamic optimization [89] to compare the performance of different algorithms/strategies for a DCARP scenario. The ANC of an algorithm/strategy for a DCARP scenario is calculated using the following equation:

$$ANC = \frac{1}{M} \sum_{m=1}^M NC_m \quad (5.4)$$

where M is the number of instances, and NC_m is the normalized cost of the corresponding algorithm/strategy in the m^{th} instance of the DCARP scenario. In our experiments, smaller values of NC and ANC indicate better performance.

5.3.3 Can SBBA Extract Optimization Experience

Since the SBBA was proposed to adapt historical solutions and obtain better initial solutions for the new environment, we compare the quality of solutions obtained by the SBBA strategy with two conventional constructive heuristics: path-scanning (PS) [36] and greedy generation

Table 5.1: Results of generated solutions' quality for SBBA, PS and GRD strategies on DCARP-OC (left part) and DCARP-CT (right part) scenarios from the EGL dataset. The value in each cell represents "MEAN $\pm$ STD" of NC for all DCARP instances in 10 generated DCARP scenarios for each map. The highlighted values denote the best results among three strategies under the Friedman test with a significance level of 0.05.

MapName	DCARP-OC Scenarios			DCARP-CT Scenarios		
	SBBA	PS	GRD	SBBA	PS	GRD
egl-g1-A	0.2259 $\pm$ 0.1001	0.5797 $\pm$ 0.0935	0.5075 $\pm$ 0.0883	0.3502 $\pm$ 0.0723	0.5369 $\pm$ 0.0907	0.4729 $\pm$ 0.0732
egl-g1-B	0.2588 $\pm$ 0.1347	0.6445 $\pm$ 0.0942	0.5637 $\pm$ 0.1060	0.3989 $\pm$ 0.0926	0.5551 $\pm$ 0.0964	0.4678 $\pm$ 0.0758
egl-g1-C	0.2213 $\pm$ 0.1254	0.6213 $\pm$ 0.0979	0.5504 $\pm$ 0.0973	0.4288 $\pm$ 0.1067	0.5369 $\pm$ 0.0972	0.4995 $\pm$ 0.0729
egl-g1-D	0.2330 $\pm$ 0.1159	0.6184 $\pm$ 0.1062	0.5431 $\pm$ 0.1057	0.4728 $\pm$ 0.1023	0.5826 $\pm$ 0.0900	0.4981 $\pm$ 0.0805
egl-g1-E	0.2664 $\pm$ 0.1331	0.6432 $\pm$ 0.1166	0.5801 $\pm$ 0.0984	0.4750 $\pm$ 0.1143	0.5898 $\pm$ 0.1067	0.4904 $\pm$ 0.0815
egl-g2-A	0.2257 $\pm$ 0.1249	0.5861 $\pm$ 0.1128	0.5222 $\pm$ 0.1005	0.3698 $\pm$ 0.0907	0.5140 $\pm$ 0.1039	0.4166 $\pm$ 0.0879
egl-g2-B	0.2560 $\pm$ 0.1212	0.6065 $\pm$ 0.1014	0.5569 $\pm$ 0.0968	0.4093 $\pm$ 0.0944	0.5634 $\pm$ 0.0964	0.4869 $\pm$ 0.0785
egl-g2-C	0.2248 $\pm$ 0.1133	0.6027 $\pm$ 0.0982	0.5434 $\pm$ 0.0903	0.4201 $\pm$ 0.1069	0.5728 $\pm$ 0.1129	0.5017 $\pm$ 0.0802
egl-g2-D	0.2296 $\pm$ 0.1102	0.6038 $\pm$ 0.1137	0.5296 $\pm$ 0.1146	0.4620 $\pm$ 0.1076	0.5708 $\pm$ 0.0929	0.5190 $\pm$ 0.0838
egl-g2-E	0.2498 $\pm$ 0.1356	0.6574 $\pm$ 0.1159	0.5820 $\pm$ 0.1266	0.4586 $\pm$ 0.1039	0.6288 $\pm$ 0.0926	0.5305 $\pm$ 0.0852
egl-s1-A	0.1876 $\pm$ 0.1350	0.6020 $\pm$ 0.1979	0.5710 $\pm$ 0.1940	0.3935 $\pm$ 0.1028	0.5755 $\pm$ 0.1401	0.5164 $\pm$ 0.1372
egl-s1-B	0.1396 $\pm$ 0.0892	0.6955 $\pm$ 0.1437	0.6202 $\pm$ 0.1565	0.4228 $\pm$ 0.1111	0.6352 $\pm$ 0.1431	0.5743 $\pm$ 0.1405
egl-s1-C	0.0921 $\pm$ 0.0759	0.6481 $\pm$ 0.2181	0.5618 $\pm$ 0.2241	0.4655 $\pm$ 0.1347	0.6461 $\pm$ 0.1355	0.5924 $\pm$ 0.1440
egl-s2-A	0.1966 $\pm$ 0.1380	0.6167 $\pm$ 0.1843	0.5646 $\pm$ 0.1790	0.3654 $\pm$ 0.1100	0.5616 $\pm$ 0.1372	0.5008 $\pm$ 0.1132
egl-s2-B	0.1487 $\pm$ 0.1321	0.6189 $\pm$ 0.2048	0.5579 $\pm$ 0.2036	0.4060 $\pm$ 0.1278	0.5935 $\pm$ 0.1428	0.5301 $\pm$ 0.1454
egl-s2-C	0.1601 $\pm$ 0.1022	0.6343 $\pm$ 0.1593	0.5651 $\pm$ 0.1835	0.4640 $\pm$ 0.1375	0.6672 $\pm$ 0.1520	0.6081 $\pm$ 0.1705
egl-s3-A	0.2017 $\pm$ 0.1371	0.6428 $\pm$ 0.1519	0.5684 $\pm$ 0.1482	0.3589 $\pm$ 0.1214	0.5513 $\pm$ 0.1353	0.5015 $\pm$ 0.1304
egl-s3-B	0.1258 $\pm$ 0.0913	0.6505 $\pm$ 0.1458	0.6086 $\pm$ 0.1483	0.4109 $\pm$ 0.0976	0.6257 $\pm$ 0.1278	0.5732 $\pm$ 0.1303
egl-s3-C	0.1436 $\pm$ 0.1086	0.6656 $\pm$ 0.1634	0.5180 $\pm$ 0.1762	0.4311 $\pm$ 0.1289	0.6248 $\pm$ 0.1520	0.5546 $\pm$ 0.1680
egl-s4-A	0.1530 $\pm$ 0.1169	0.5596 $\pm$ 0.1678	0.4838 $\pm$ 0.1606	0.4026 $\pm$ 0.1514	0.5943 $\pm$ 0.1513	0.5179 $\pm$ 0.1665
egl-s4-B	0.1592 $\pm$ 0.1081	0.5946 $\pm$ 0.1903	0.5573 $\pm$ 0.1841	0.4126 $\pm$ 0.1370	0.6030 $\pm$ 0.1286	0.5199 $\pm$ 0.1317
egl-s4-C	0.1468 $\pm$ 0.1089	0.6061 $\pm$ 0.1597	0.5444 $\pm$ 0.1613	0.4320 $\pm$ 0.1157	0.6205 $\pm$ 0.1341	0.5411 $\pm$ 0.1133

(GRD) [113] algorithms. These two strategies are still the most popular and widely-adopted initialization mechanisms in the meta-heuristic algorithms for optimizing (D)CARP in the literature. The MAENS algorithm was used to optimize each DCARP instance, and the l_{AS} best solutions of the former DCARP instance were saved for the new DCARP instance. Thus, for a new DCARP instance, we could obtain at most l_{AS} solutions from the SBBA strategy, one solution from the PS strategy, and p_{size} solutions from the GRD strategy. The performance of the three strategies was evaluated by comparing their best-generated solutions.

NC performance measurement was used to compare the performance of three strategies. For each DCARP instance, in Eq.(5.3), TC represents the cost of the best solution obtained by the corresponding strategy, TC_{min} represents the cost of the best solution obtained by MAENS after optimization, and TC_{max} represents the cost of the worst solution among all solutions obtained by the three strategies. We generated 20 different DCARP scenarios for each map, including 10 DCARP-OC scenarios and 10 DCARP-CT scenarios. For each type of DCARP scenario for each map, the mean and standard deviation (mean $\pm$ std) of all NC values for all DCARP instances of the three different strategies are presented in Table 5.1. For each map, we applied the Friedman test with a significance level of 0.05 to all NC values corresponding to the obtained solutions of the three strategies in all DCARP instances of the 10 generated DCARP scenarios. The best results under the hypothesis test are highlighted in bold values with a gray background. From Table 5.1, it is clear that the quality of solutions obtained by SBBA is significantly better than the other two conventional constructive strategies, both in DCARP-OC and DCARP-CT scenarios.

We also calculate the difference in solution's quality between the SBBA and each of the other two strategies in all DCARP instances for DCARP-OC and DCARP-CT scenarios (Figure 5.2). A higher difference implies a higher improvement in NC using the SBBA strategy. As we can see, the improvements were present in both scenarios, but were of

larger magnitude for the DCARP-OC scenarios. In other words, the building blocks were particularly helpful when coping with cost-related dynamic events.

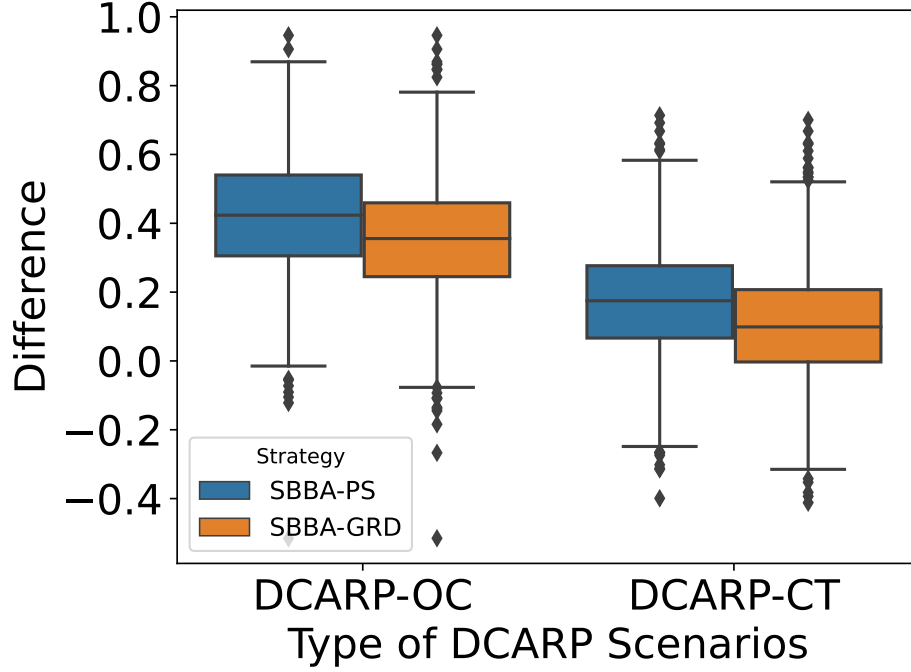


Figure 5.2: The difference of the generated solution's quality between the SBBA and other two strategies in all DCARP instances for DCARP-OC and DCARP-CT scenarios.

We hypothesized that SBBA outperforms the other two strategies because SBBA managed to obtain solutions that are more similar to the best solution for a DCARP instance. To test this hypothesis, a type of “similarity” (*Sim*) between one solution and the “best” solution was defined. The best solution is set as the best among all solutions obtained when running MAENS both with SBBA and with GRD as the initialization strategy. The similarity was calculated using the following equation:

$$Sim = \frac{\#SameLinks}{\#TotalLinks} \quad (5.5)$$

where $\#TotalLinks$ is the number of links between two consecutive tasks in the best solution, and $\#SameLinks$ is the number of links that exist in both the given solution and the best solution. If more links in the best solution also exist in the given solution, the *Sim* value

will be higher, indicating that the given solution is more similar to the best solution.

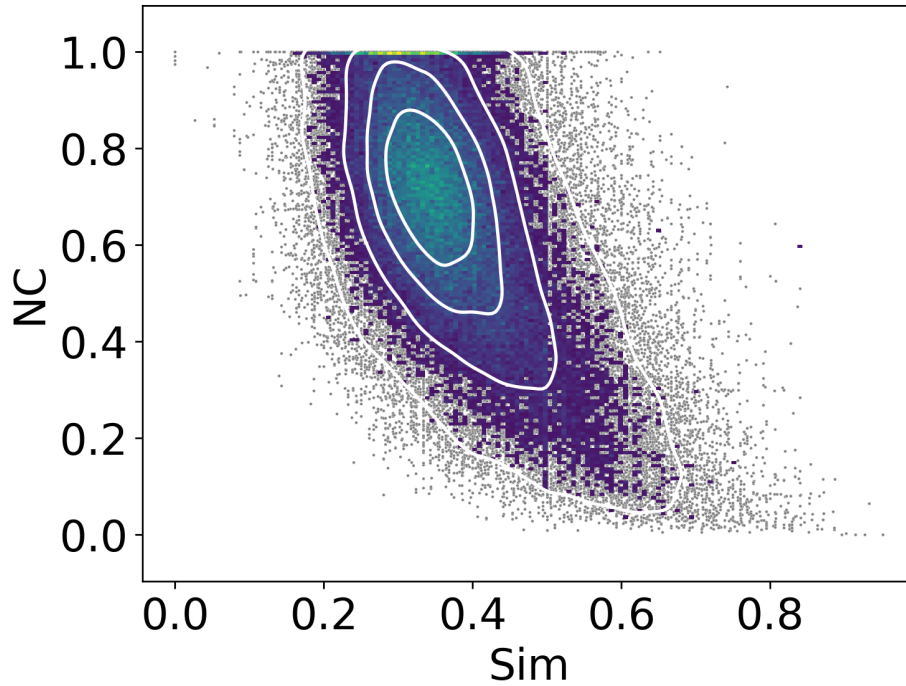


Figure 5.3: Results of the NC and Sim for all solutions generated by three strategies in all DCARP scenarios of all maps. White contours represent the estimated distribution for all data.

We calculated the NC and Sim values for all solutions generated by the three strategies in all DCARP scenarios and maps, and the results are presented in Figure 5.3. The grey points in the figure represent the data of all solutions, while purple distribution and white contours represent the estimated distribution. From the figure, it is evident that the value of NC decreases as the Sim value increases. Therefore, a strategy that can generate solutions that are more similar to the best solution will have higher-quality solutions.

Then, we calculated the NC and Sim values of the best solutions obtained by each strategy in different types of DCARP scenarios (Figure 5.4). Figure 5.4a and Figure 5.4b show the distributions of solutions for DCARP-OC and DCARP-CT scenarios, respectively. The colored areas represent the estimated distribution of solutions obtained by different strategies. It is evident that the solutions obtained by SBBA are located in the bottom right part for

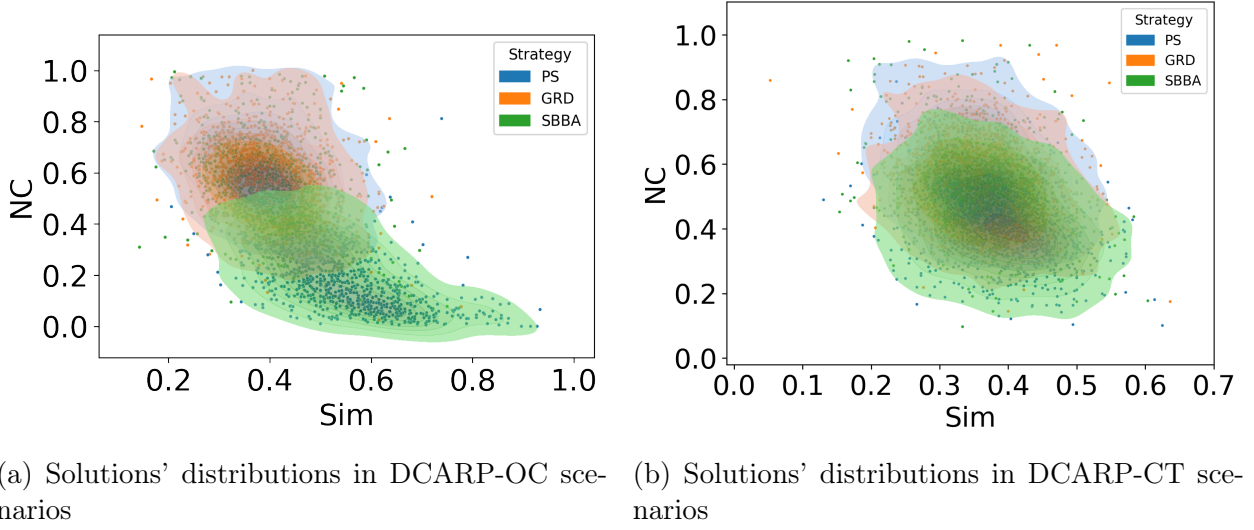


Figure 5.4: The distribution of solutions' Sim and NC for path-scanning (PS), greedy generation (GRD), and SBBA strategies.

both types of DCARP scenarios, and especially for DCARP-OC. This indicates that the solutions generated by SBBA are overall more similar to the best solution and have a higher quality than the other two strategies in both DCARP-OC and DCARP-CT scenarios, and especially for DCARP-OC. This is in line with the improvements in NC , which were present for both DCARP-OC and DCARP-CT, but were relatively larger for DCARP-OC (Figure 5.2) as discussed earlier in this section.

Additionally, our motivation for the SBBA is to retain the high-quality solutions' characteristics of the former DCARP instance by the *building block* so that the new DCARP instance can benefit from such characteristics and generate better solutions. The above results between the solution's similarity to the best solution and the solution's quality also demonstrated that our proposed SBBA indeed retains the valuable characteristics of the high-quality solutions of the former DCARP instance. SBBA is effective for generating high-quality solutions, especially in DCARP-OC scenarios (Figure 5.2).

5.3.4 Can SBBA Promote Dynamic Optimization

The solutions generated by SBBA have been demonstrated to be of better quality than the solutions generated by conventional constructive strategies in the above experiments. In this subsection, the generated solutions by SBBA strategy were used as the initial solutions for the dynamic optimization. As mentioned in Section 5.3.1, two meta-heuristic algorithms, i.e., MAENS (population-based) [113], and RTS (individual-based) [78], are employed to evaluate the performance of proposed DO-SBBA framework.

The meta-heuristic algorithm with SBBA strategy was compared to the same algorithm with GRD strategy in both DCARP-OC and DCARP-CT scenarios. For each type of DCARP scenario, we generated 10 different DCARP scenarios for each map, and each optimization algorithm optimized each DCARP instance for 25 independent runs. The *ANC* was used for the performance measurement. We calculated the *ANC* value in each independent run for meta-heuristic algorithms with different strategies. Then, for each DCARP scenario, the Wilcoxon signed-rank test with a significance level of 0.05 was applied to all *ANC* values from 25 independent runs for two strategies. The results of MAENS and RTS algorithms with the SBBA and GRD strategies in DCARP-OC and DCARP-CT scenarios are presented in Table 5.2. The values in each cell represent the number of “win-draw-lose” of the SBBA strategy versus the GRD strategy under the hypothesis test over 10 different DCARP scenarios for each map. Besides, we also plotted the convergence curves of meta-heuristic algorithms with different strategies in DCARP scenarios, and some of them are presented in Figure 5.5. Each plot represents a complete DCARP scenario including all DCARP instances. Each segment in the plot represent the algorithm’s convergence curve within the maximum optimization time in the corresponding DCARP instance.

Based on the results presented in Table 5.2 and Figure 5.5, it can be observed that the SBBA strategy improves the optimization performance of both MAENS and RTS algorithms

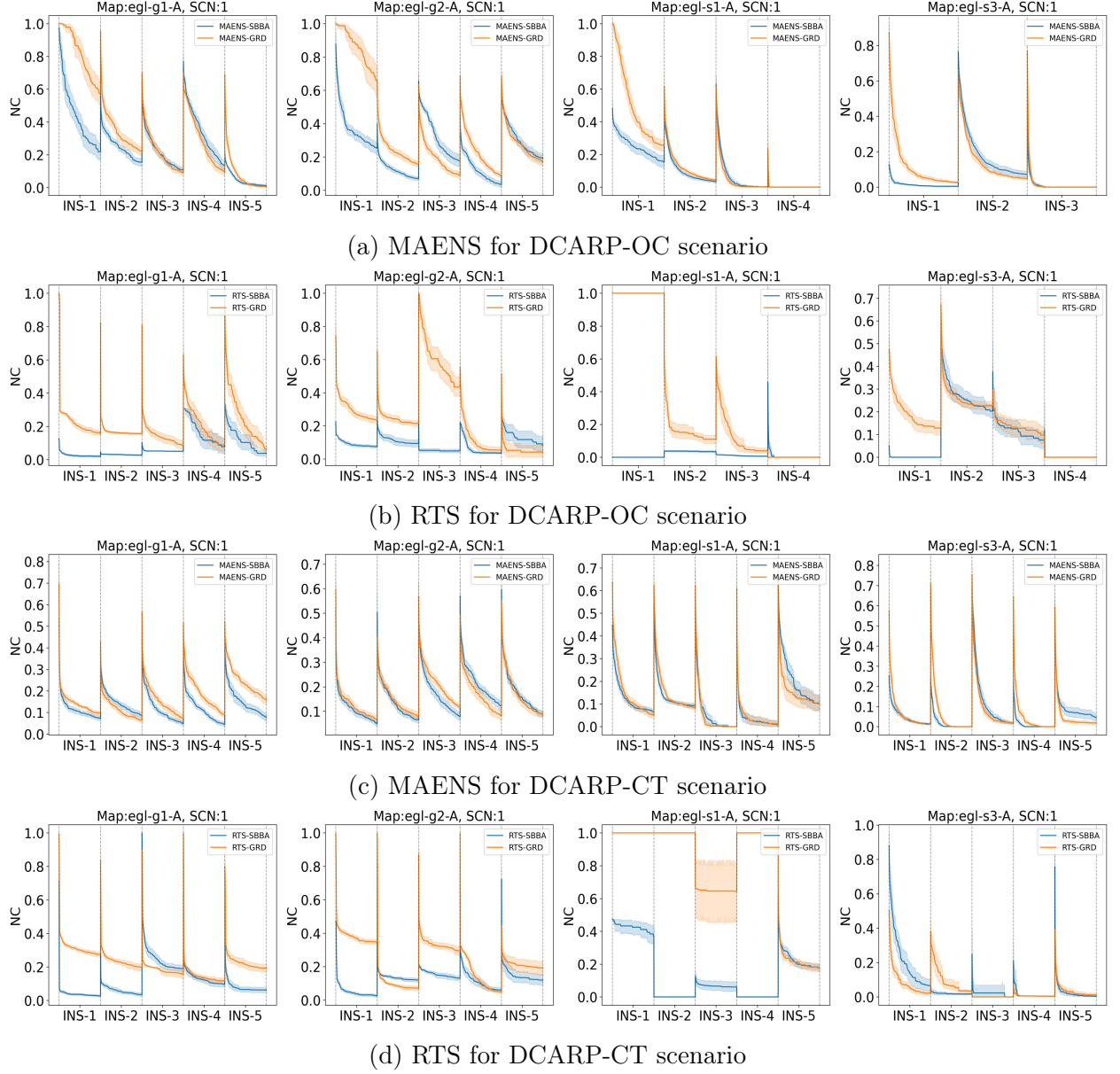


Figure 5.5: The convergence curves of MAENS [113] and RTS [78] with SBBA (*blue lines*) and GRD (*orange lines*) strategies in some DCARP-OC and DCARP-CT scenarios.

Table 5.2: Comparison of MAENS and RTS meta-heuristic algorithms using SBBA against GRD in DCARP-OC and DCARP-CT scenarios. The values in each cell represent the number of “win-draw-lose” for the SBBA strategy based on Wilcoxon signed-rank tests with a significance level of 0.05 in 10 DCARP scenarios of each map.

MapName	DCARP-OC Scenarios		DCARP-CT Scenarios	
	MAENS	RTS	MAENS	RTS
egl-g1-A	10-0-0	8-2-0	6-4-0	7-1-2
egl-g1-B	10-0-0	7-1-2	3-7-0	6-2-2
egl-g1-C	10-0-0	8-0-2	2-6-2	4-3-3
egl-g1-D	10-0-0	8-2-0	4-5-1	2-3-5
egl-g1-E	10-0-0	8-2-0	4-5-1	1-5-4
egl-g2-A	10-0-0	7-1-2	3-5-2	8-1-1
egl-g2-B	10-0-0	6-2-2	4-6-0	4-3-3
egl-g2-C	10-0-0	10-0-0	5-5-0	5-2-3
egl-g2-D	10-0-0	7-0-3	5-5-0	6-1-3
egl-g2-E	10-0-0	6-3-1	6-4-0	4-3-3
egl-s1-A	4-4-2	9-1-0	4-5-1	5-1-4
egl-s1-B	3-7-0	10-0-0	4-4-2	7-0-3
egl-s1-C	8-1-1	10-0-0	2-7-1	3-4-3
egl-s2-A	8-2-0	3-5-2	1-8-1	2-7-1
egl-s2-B	5-5-0	4-5-1	1-7-2	2-7-1
egl-s2-C	3-6-1	2-8-0	2-6-2	2-8-0
egl-s3-A	6-4-0	7-3-0	3-5-2	2-5-3
egl-s3-B	7-3-0	4-6-0	3-6-1	1-8-1
egl-s3-C	5-4-1	5-5-0	2-7-1	1-9-0
egl-s4-A	5-5-0	6-4-0	2-8-0	1-6-3
egl-s4-B	5-5-0	2-5-3	0-10-0	2-7-1
egl-s4-C	4-5-1	4-6-0	4-6-0	1-7-2

in most DCARP scenarios, especially in DCARP-OC scenarios. However, in most DCARP-CT scenarios, the MAENS/RTS with the SBBA strategy performs similarly to MAENS/RTS with the GRD strategy. This is understandable, since the SBBA analysis shown in Section 5.3.3 showed that this strategy usually leads to larger improvements in the initial solutions for DCARP-OC than for DCARP-CT. The convergence speed is also faster with the SBBA strategy, as shown in Figure 5.5. This is due to the higher-quality and higher-similarity initial solutions generated by the SBBA strategy (Figure 5.4). The higher-quality solutions used for optimization can lead to faster convergence, and solutions with higher similarity indicate they are closer to the best solution, which can also lead to faster convergence of meta-heuristic algorithms.

The SBBA strategy is shown to be effective in promoting the performance of meta-heuristic algorithms in DCARP scenarios, especially in DCARP-OC scenarios. This indicates that the SBBA is much more effective for DCARP-OC scenarios without task changing. Additionally, in Table 5.2, MAENS with SBBA strategy outperforms the GRD strategy in all DCARP-OC scenarios of *EGL-G* maps, which has larger number of tasks. This indicates that the MAENS with SBBA strategy significantly outperforms the GRD strategy in DCARP scenarios with more tasks because scenarios with a smaller number of tasks are too easy for MAENS so that the effectiveness of SBBA is not obvious. For RTS, since it only uses one adaptive solution for optimization, the effectiveness of SBBA is also not obvious. However, it still provides a performance boost for RTS if the adaptive solution has a higher quality according to Figure 5.5b and 5.5d.

Therefore, we can conclude from our experimental results that DO-SBBA can promote the dynamic optimization performance of meta-heuristic algorithms. Specifically, the DO-SBBA is much more effective in DCARP-OC scenarios and DCARP scenarios that are harder for the original meta-heuristic algorithms to handle.

5.3.5 How Well SBBA Performs under Different Severities of Dynamic Changes

To investigate the generalization ability of the proposed SBBA, we applied it to the DCARP scenarios with varying severity of dynamic changes. We employed the MAENS algorithm to optimize each DCARP instance in the dynamic optimization framework.

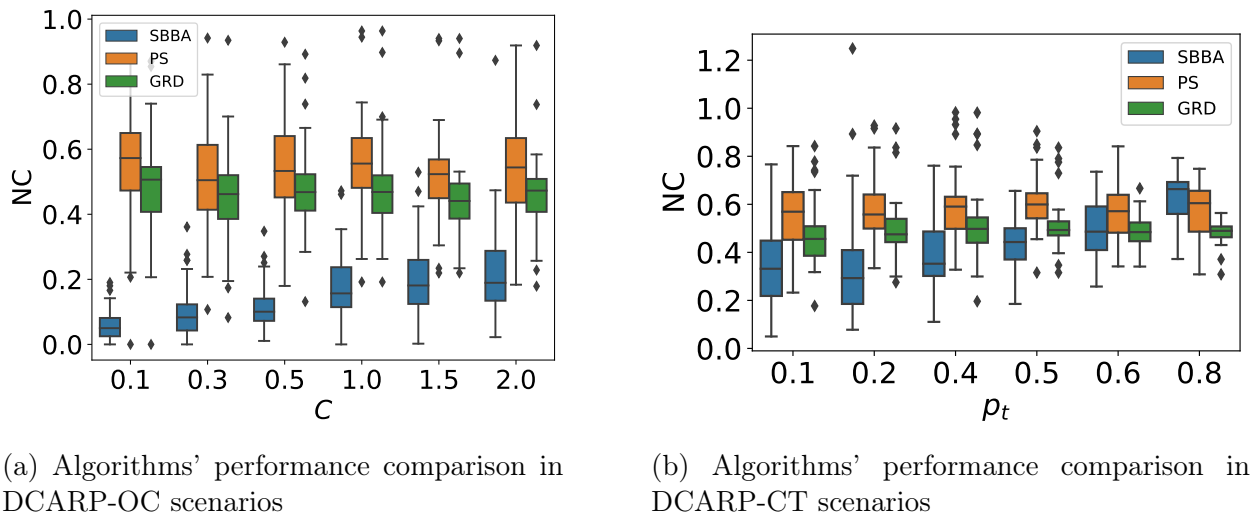


Figure 5.6: The performance comparison between path-scanning (PS), greedy generation (GRD) and SBBA strategies on DCARP scenarios with different severity of dynamic changes.

For each setting of C and p_t , which control the severity of dynamic changes, we generated 10 different DCARP scenarios independently. Smaller C and p_t indicate that smaller changes in edges' deadheading costs and fewer number of added tasks, respectively. We calculated the NC of solutions generated by SBBA, path-scanning (PS), and greedy generation (GRD) in all DCARP instances of the 10 scenarios. The results are presented in the boxplots in Figure 5.6. For the DCARP-OC scenarios in Figure 5.6a, our SBBA strategy outperforms the other two algorithms, even when the severity of dynamic changes is larger. However, the performance gap between SBBA and the other two algorithms decreases as the severity of dynamic changes increases. For the DCARP-CT scenarios in Figure 5.6b, our SBBA strategy is slightly better than the other two constructive heuristic algorithms when the

number of added tasks is small. However, when the number of added tasks becomes larger, the performance of SBBA decreases substantially and is even worse than the other two algorithms when $p_t > 0.6$.

Therefore, based on the experimental results, our proposed SBBA strategy is more effective when the severity of dynamic changes is small. This is because small dynamic changes make the new DCARP instance more similar to the previously optimized DCARP instance, causing high-quality solutions for both DCARP instances to also be similar. The SBBA strategy utilizes these similarities to generate solutions for new DCARP instances. Therefore, small dynamic changes are more suitable for our SBBA strategy. When the severity of dynamic changes becomes large, our SBBA strategy is more robust in DCARP-OC scenarios than in DCARP-CT scenarios. Furthermore, the SBBA strategy performs even worse than the other two algorithms when the number of added tasks is large in DCARP-CT scenarios. This is because in good solutions of new DCARP instances, the newly added tasks and remaining tasks might form a new building block, and so the building blocks for the former DCARP instances are likely to cause a bad solution.

5.4 Summary

In this chapter, we focused on dynamic optimization for the DCARP scenario, which considers unforeseen dynamic events occurring during the deployment of the CARP solution. Existing works on DCARP optimization usually apply re-optimization from scratch to solve the DCARP scenario. However, since DCARP instances in a scenario are generated based on the previous instance, they share many similarities. The optimization experience for the previous instance can be leveraged to facilitate dynamic optimization for the new instance. Therefore, we proposed a solution building block adaptive strategy based dynamic opti-

mization framework (DO-SBBA) to assist meta-heuristic algorithms in solving the DCARP scenario. We considered two different DCARP scenarios: the DCARP-OC scenario containing instances generated only from cost-changing dynamic events, and the DCARP-CT scenario containing instances generated from both cost-changing and task-changing dynamic events. We proposed the *building blocks* to extract valuable experience from historical solutions. For DCARP-OC scenarios, we proposed a building block-based constructive strategy called SBBA to adapt historical solutions to new DCARP-OC instances. Furthermore, for DCARP-CT scenarios, we proposed a building block-based constructive strategy and an insertion strategy to handle new DCARP-CT instances.

To evaluate the efficacy of the proposed SBBA strategy and the optimization framework, we applied them to a series of DCARP-OC and DCARP-CT scenarios. The SBBA strategy was compared with two conventional constructive strategies in terms of the quality of directly generated solutions. The experimental results demonstrated that the SBBA benefits a lot from the experience contained in historical solutions and significantly outperforms the other two strategies. Empirical analysis indicated that solutions constructed by the SBBA are more similar to the best solutions with the help of the extracted experience. Furthermore, we analyzed the performance of the DO-SBBA framework for dynamic optimization by embedding two meta-heuristic algorithms, MAENS [113] and RTS [78]. The experimental results showed that the DO-SBBA framework led to fast convergence and helped the original algorithm obtain better solutions for both algorithms in both DCARP-OC and DCARP-CT scenarios. Since SBBA can generate a set of adaptive solutions, the MAENS, which is a population-based algorithm, benefits more from the SBBA strategy than the RTS algorithm, which is an individual-based algorithm. We also found that the proposed framework is particularly helpful for DCARP-OC scenarios, with improvements in performance being observed across different severity of change in cost. For DCARP-CT scenarios, improvements in performance were larger for smaller changes in the number of added tasks.

Therefore, the SBBA strategies and the new dynamic optimization framework proposed in this chapter answer the *RQ3*: “How to extract optimization experience from a DCARP instance and integrate into a meta-heuristic framework for enhanced performance in different DCARP scenarios?”. However, we have only evaluated the proposed algorithms through comparison with baseline algorithms. The performance of algorithms relative to the theoretical global optimum remains unknown. Since the theoretical global optimum is typically unknown for most DCARP instances, we are going to derive a lower bound in the next chapter to approximately estimate the performance of our algorithm relative to theoretical global optimum to answer the *RQ4*: “How can we derive a lower bound for evaluating the DCARP solution quality rather than relying solely on comparison with baseline algorithms?”. Besides, the performance of our algorithm in real-world applications is also still unknown. Therefore, we will also focus on applying our algorithms to DCARP scenarios with real traffic conditions to in following chapters to answer *RQ5*: “How can we evaluate and how well do our proposed DCARP algorithm perform in real-world scenarios in addition to artificial benchmarks?”.

Chapter Six

Performance Measurement with Lower Bound

This chapter is based on following publication: *Tong, Hao, Leandro L. Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao. “Evaluating meta-heuristic algorithms for dynamic capacitated arc routing problems by deriving lower bounds.” In submission to IEEE Computation Intelligence Magazine.*

6.1 Introduction

The performance of proposed algorithms for DCARP has been validated through comparing the optimization result with the baseline algorithm using statistical tests. It only measures the relative performance compared to the baseline algorithm without any estimation about the performance relative to the global optimum. However, the algorithm's performance in relation to the global optimum is crucial for assessing the remaining potential for improvement in the optimization algorithms.

To the best of our knowledge, only a percentage deviation was proposed based on a *posterior* lower bound to estimate the performance of DCARP optimization algorithms [94]. However, this *posterior* lower bound corresponds to the static CARP instance, which means all tasks are treated as if they are all known in advance, instead of the lower bound of each actual DCARP instance. Therefore, when the theoretical optimum is unknown, the lower bound would be very important for measuring the performance of the meta-heuristic algorithm (Hochba, 1997), and also for a better understanding of the problem and its solution method. However, methods for calculating the lower bound for DCARP instances are missing in the literature.

Therefore, in this chapter, we derived a tight lower bound for DCARP instances. Then, the performance of meta-heuristic algorithms for DCARP is evaluated through assessing the quality of DCARP solutions based on the derived lower bound, answering the *RQ4*: “How can we derive a lower bound for evaluating the DCARP solution quality rather than relying solely on comparison with baseline algorithms?”.

Our contributions are as follows:

- We propose the first lower bound approach for DCARP, named node matching lower bounds (NMLB), which converts the lower bound problem to a minimum cost matching

problem (MCMP). A lower bound for a DCARP instance is obtained by applying the Blossom algorithm [35] to solve the MCMP. The tightness of the lower bound is analyzed on a set of DCARP instances whose theoretical optima are known and the analysis concludes that the obtained NSLB is very close to the theoretical optimum on the testing DCARP instances.

- We also propose a graph pruning strategy that removes a subset of redundant vertices in the MCMP, which increases the computational efficiency of the proposed NMLB approach. Its efficiency is demonstrated and analyzed in our experimental studies.
- We applied the derived lower bound to evaluate the performance of the proposed algorithms in Chapters 3 and 5 on a benchmark of DCARP instances. The experimental analysis reveals its excellent performance on relatively small instances and potential improvements on large complex instances, which is the first time in the literature.

The remainder of this chapter is organized as follows. Section 6.2 introduces lower bound approaches for static CARP instances. Section 6.3 introduces our proposed node matching lower bound approach and the graph pruning strategy. Section 6.4 presents experimental results about the tightness of lower bound, measuring the performance of optimization algorithm and the efficacy of the pruning strategy. The summaries of this chapter are provided in Section 6.5.

6.2 Lower Bound for Static CARP Instance

During the past decades, several lower-bound methods have been introduced for static CARP instances. Two fundamental approaches, including the matching lower bound (MLB) [37] and the nodes scanning lower bound (NSLB) [4], stand out. MLB deduces the lower bound by adding artificial edges to transform the graph into an Euler graph [37]. NSLB considers

that vehicles would traverse a path from the depot to a given node before serving any tasks and that the number of paths linking to a node would not exceed the number of connected tasks. Thus, NSLB adds artificial edges between the depot and nodes to obtain a lower bound [4]. Further approaches have been proposed based on these two basic methods. For example, [9] proposed two lower bounds (LB1 and LB2) based on NSLB, and [108] proposed the node duplication lower bounds (NDLB) method based on MLB. [131] combined MLB and NSLB and proposed a tighter lower bound, called MCNDLB. Besides, [5] used cut-and-column generation to get a new improved lower bound for static CARP instances. Subsequently, the MLB and NSLB will be introduced in detail.

6.2.0.1 Node Scanning Lower Bound

Node scanning lower bound [4] is a fundamental method for static CARP. It is based on the constraint that each node $v_i \in G_R$ can appear no more than $d(v_i)$ times in the solution because each task can only be served once. Let C^* be the optimal cost for a CARP instance. It can be divided into three parts: the total service costs for all tasks (C_T), the costs of artificial edges directly linked to the depot (C_1), and the costs of artificial edges linking different tasks (C_2). Thus, we can have the following condition:

$$C^* = C_T + C_1 + C_2 \geq C_T + C_1 \quad (6.1)$$

Since the value of C_T can be calculated directly for a given CARP instance, the lower bound for C^* can be obtained by finding a lower bound of C_1 .

As the minimum number of vehicles/routes required to serve all tasks is $k_0 = \left\lceil \frac{W_T}{Q} \right\rceil$, the number of edges directly linking to the depot in the solution is at least $2k_0$. If the degree of the depot $d(v_0)$ in G_R is greater than $2k_0$, i.e., $2k_0 \leq d(v_0)$, artificial edges may not be required in the optimal solution because the number of existing edges linking to the depot

are enough to construct k_0 routes. Therefore, the lower bound for C_1 is 0, and C_T becomes the lower bound for the CARP instance. On the other hand, for $2k_0 > d(v_0)$, to satisfy the minimum requirement of constructing k_0 routes, we have to add $J = 2k_0 - d(v_0)$ artificial edges directly linking to the depot with minimum costs in G_R . Each node $v_i \in V_R \setminus \{v_0\}$ will never appear more than $d(v_i)$ times in a solution. Thus, each node $v_i \in V_R \setminus \{v_0\}$ can link at most $d(v_i)$ artificial edges in the solution. Therefore, the NSLB can be obtained by the following three steps:

Step 1: Renumber all vertices in $V_R/\{v_0\}$ (excluding the depot) according to their minimum costs from the depot, i.e., $mdc(v_0, v_1) \leq mdc(v_0, v_2) \leq \dots \leq mdc(v_0, v_{|V_R|-1})$, where $|V_R|$ denotes the number of vertices in V_R .

Step 2: Determine the smallest H that satisfies $\sum_{i=1}^H d(v_i) \geq J = 2k_0 - d(v_0)$.

Step 3: Reset the degree of v_K to $d(v_K) = J - \sum_{i=1}^{H-1} d(v_i)$.

As a result, the node scanning lower bound $NSLB$ for static CARP is calculated by

$$NSLB = C_T + \sum_{i=1}^H d(v_i) mdc(v_0, v_i) \quad (6.2)$$

6.2.0.2 Node Duplication Lower Bound

Node duplication lower bound [108] also considers that each task can only be served once in a solution. Therefore, they created an auxiliary graph by duplicating each node $v_i \in V_R$, which is incident with at least one task, with $d(v_i)$ copies. The lower bound is obtained by solving a minimum cost matching problem (MCMP) on the auxiliary graph.

Since the $k_0 = \left\lceil \frac{W_T}{Q} \right\rceil$ is also the minimum number of vehicles required for serving all tasks, the depot will occur at least $2k_0$ times in the optimal solution as the starting and ending

vertices of the routes. The node v_i will occur $d(v_i)$ times in any solution to the CARP. Therefore, the NDLB method constructs an auxiliary graph by creating copies of the depot and all nodes in V_R including the following steps: [108]:

Step 1: Define a family of nodes $v_i \in V_R \cup v_0$ as $fam(v_i)$, and each $fam(v_i)$ consists of a set of $d(v_i)$ nodes that represent copies of node v_i . For $fam(v_0)$, add $\max(0, 2k_0 - d(v_0))$ nodes. The set of all nodes in V_x is denoted as $\bigcup_{v_i \in V_R \cup v_0} fam(v_i)$.

Step 2: For each pair of nodes $v_i, v_j \in V_x$, add one artificial edge between them to form the set E_x .

Step 3: In E_x , determine the demand edges according to the tasks in the CARP instance and assign the demand to the corresponding demand edge.

Step 4: For each edge $(v_i, v_j) \in E_x$, assign its cost with the cost of the shortest path in the original graph. If two nodes belong to the same family (resp. $fam(v_0)$), set the edge's cost with 0 (resp. ∞). If the edge links two demand edges whose total demands are greater than the capacity, set the edge's cost with ∞ . If one end node of the edge is a v_0 belongs to a demand edge, set the edge's cost with ∞ .

Step 5: Solve a minimum cost matching problem in $G_x = (V_x, E_x)$.

Let M , and $opv(M)$ be the solution of MCMP and its optimal value, the lower bound of the CARP instance $NDLB = opv(M) + C_T$, in which C_T is also the total service costs for all tasks.

6.2.0.3 The Gap of Using Existing Lower Bounds Approaches in DCARP

Although the current lower bounds for static CARP instances have been reasonably tight, the lower bound approaches are unsuitable for deriving the lower bound of DCARP instances.

This is because the lower bound approaches for static CARP instances only consider routes starting from the depot. However, since DCARP instances are generated due to the dynamic events occurrence when vehicles are in their services, the DCARP solution consists of routes belonging to outside vehicles which start from different locations instead of the depot. These outside vehicles can continue serving some tasks and then return to the depot, or they can also return to the depot directly. These cases have to be taken into account when deriving the lower bound, but the static lower bound approach is not capable of handling them.

In Chapter 3, we proposed a “virtual-task” strategy for converting a DCARP instance into a “virtual” static CARP instance, enabling optimization algorithms designed for static CARP to be applied to DCARP. The virtual-task strategy creates a virtual task between the depot and the location where each outside vehicle stops. Each virtual task is assigned the same demand as that already served by a given outside vehicle. Serving a virtual task once using a depot vehicle will result in the location and remaining capacity of this vehicle being the same as that of the corresponding outside vehicle when the change happened. Therefore, during optimization, each outside vehicle can be treated as if it was a vehicle located at the depot, but serving its corresponding virtual task once. However, even though all vehicles are located at the depot with the same capacity in the “virtual” static CARP instance, it is still not suitable to apply existing lower bound approaches for static CARP instances to such “virtual” static CARP instances. This is because the virtual task is constructed for optimization with static CARP optimization algorithms, and these virtual tasks have to be the first task in a route. Furthermore, two virtual tasks can not exist in the same route in the final solution. Therefore, the outside vehicles are still required to be considered when deriving the lower bound using the converted “virtual” static CARP instances with static lower bound approaches.

Therefore, to fill the gap of deriving the lower bound for DCARP instances, we propose a node matching lower bound approach motivated by the NSLB [4] and NDLB [108] approaches

for static CARP instances. More details of the proposed approaches are presented in Section 6.3.

6.3 Performance Evaluation with Node Matching Lower Bound for DCARP

6.3.1 Node Matching Lower Bound Algorithm

For a given DCARP instance, we assume a solution $S = \{r_1, r_2, \dots, r_K\}$ is composed of several routes. In each route $r_k : (sp_k, t_{k,1}, t_{k,2}, \dots, t_{k,l_k}, v_0)$, the sp_k is the starting point of the route and $t_{k,i}$ is the i^{th} task of the route. Suppose $head_{t_{k,i}}$, $tail_{t_{k,i}}$ denote the beginning and ending vertices for serving the task $t_{k,i}$, the route r_k can be represented by a node sequence as follows:

$$r_k = (sp_k, head_{t_{k,1}}, tail_{t_{k,1}}, head_{t_{k,2}}, tail_{t_{k,2}}, \dots, tail_{t_{k,l_k}}, v_0)$$

Therefore, a DCARP solution can be formed by adding the following three types of artificial links:

1. Artificial links between the starting vertex of the route and the head vertex of the first task.
2. Artificial links between the tail vertex of the last task and the returning vertex of the route (the depot).
3. Artificial links between the tail vertex of one task and the head vertex of its next task.

The total costs of a solution are determined by the costs of these artificial links because the total service costs of all tasks are fixed according to the objective formulation in (2.1) and

(2.3). Thus, if we can identify a case of adding artificial links between tasks and starting and returning vertices with minimum total costs, a valid lower bound for the DCARP instance can be obtained.

At first, we must determine the minimum number of routes required for the given DCARP instance to determine the minimum number of starting and returning vertices required in a DCARP solution. Due to the outside vehicles in the DCARP instance, the solution of the DCARP instance consists of routes not only starting from the depot but also from these outside vehicles. These outside vehicles can also continue to serve some tasks, thereby affecting the minimum number of vehicles required to start from the depot, which can be calculated as:

$$k_0 = \left\lceil \frac{W_T - \sum_{i=1}^{N_{ov}} q_i}{Q} \right\rceil \quad (6.3)$$

The number of routes belonging to the outside vehicles equals the number of outside vehicles, i.e., N_{ov} .

In a DCARP solution, the starting vertices of routes can be not only the depot but also nodes associated with outside vehicles (outside vehicles' nodes). The set of stop locations for outside vehicles is represented as $V_{ov} = \{sp_1, sp_2, \dots, sp_{N_{ov}}\}$. The returning vertices are always the depot. Thus, for adding artificial links related to the starting and returning vertices, five constraints have to be satisfied in a DCARP solution:

1. The depot in k_0 routes associated with vehicles starting from the depot must link to a task's node $v_i \in V_R$.
2. The depot in N_{ov} routes associated with outside vehicles can link to a task's node or directly link to a starting vertex $sp_i \in V_{ov}$.
3. The starting vertex $sp_i \in V_{ov}$ in N_{ov} outside vehicles' routes can link to a task's node or directly link to a depot.

4. If a starting vertex $sp_i \in V_{ov}$ in N_{ov} outside vehicles' routes links to a task's node, the returning vertices (depot) in these routes must also need to link to a task's node.
5. If a starting vertex $sp_i \in V_{ov}$ in N_{ov} outside vehicles' routes link to the depot, the returning vertex (depot) in these routes cannot link to other tasks' nodes anymore.

Algorithm 8: Pseudo-code of constructing the auxiliary graph

Input: Graph $G_R = (V_R, E_R)$

- 1 Initialise four empty sets **A**, **B**, **C**, **D**.
- 2 Add $2k_0$ nodes into **A** representing $2k_0$ copies of the depot v_0 . // Depots needed for new vehicles to start at, and return to.
- 3 Add N_{ov} nodes into **C** representing N_{ov} copies of the depot v_0 . // Depots needed for outside vehicles to return to.
- 4 Add all nodes in V_{ov} into **D**, i.e., $\mathbf{D} = V_{ov}$. // Nodes where outside vehicles are located.
- 5 **for** each node $v_i \in V_R$ **do**
- 6 Add $d(v_i)$ copies of node v_i into **B**. // Nodes which vehicles can pass through when serving tasks
- 7 Set $V_x = \mathbf{A} \cup \mathbf{B} \cup \mathbf{C} \cup \mathbf{D}$, $E_x = \emptyset$.
- 8 Add links between **A** and **B**, $E_x = E_x \cup \{(v_i, v_j) | v_i \in \mathbf{A}, v_j \in \mathbf{B}\}$.
- 9 Add links between **B** and **C**, $E_x = E_x \cup \{(v_i, v_j) | v_i \in \mathbf{B}, v_j \in \mathbf{C}\}$.
- 10 Add links between **B** and **D**, $E_x = E_x \cup \{(v_i, v_j) | v_i \in \mathbf{B}, v_j \in \mathbf{D}\}$.
- 11 Add links between **C** and **D**, $E_x = E_x \cup \{(v_i, v_j) | v_i \in \mathbf{C}, v_j \in \mathbf{D}\}$.
- 12 Add links inside set **B**, $E_x = E_x \cup \{(v_i, v_j) | v_i \in \mathbf{B}, v_j \in \mathbf{B}, v_i \neq v_j\}$.
- 13 **for** each link $(v_i, v_j) \in E_x$ **do**
- 14 Set its cost equal to the cost between the two corresponding nodes in G_R .

Output: Auxiliary graph $G_x = (V_x, E_x)$

Therefore, an auxiliary graph G_x can be constructed based on the above constraints, and the pseudo-code for constructing the auxiliary graph is presented in Algorithm 8. Four different empty subsets, named as **A**, **B**, **C** and **D**, are initialized to include nodes with different roles in a DCARP solution. Set **A** includes $2k_0$ nodes representing the starting and returning vertices of k_0 routes starting from the depot, which are all depots. Set **C** includes N_{ov} nodes representing the returning vertices of N_{ov} routes associated with the outside vehicles. Set **D** includes the starting vertices of N_{ov} routes belonging to the outside vehicles, i.e., the locations where outside vehicles stop. Set **B** mainly consists of the vertices

incident with the tasks in G_R . Since each task can only be served once in a DCARP solution, any vertex v_i can not be linked over $d(v_i)$ times (i.e., the degree in G_R) when adding artificial links to form the DCARP solution. Thus, we add $d(v_i)$ copies of each vertex $v_i \in V_R$ into set **B**. These four sets are combined to form the set V_x . Then, according to the above five constraints, we add links between sets **A** and **B**, **B** and **C**, **B** and **D**, and sets **C** and **D**. We also add links between nodes in set **B**, representing the links between tasks in a solution. All these links form a set E_x . For each link in E_x , its cost equals to the corresponding minimum costs between two vertices in G_R . Finally, the auxiliary graph $G_x = (V_x, E_x)$ is constructed and used to calculate the instance's lower bound.

Theorem 1. *Let C_{MCMP} be the cost of the optimal solution of the minimum cost matching problem on G_x . Then, $NMLB = C_T + C_{MCMP}$ provides a valid lower bound for a DCARP instance.*

Proof. The optimal solution of the DCARP instance comprises tasks, artificial links between tasks' head and tail vertices, and artificial links between tasks' endpoints and the starting and returning vertices of the routes. The total service costs C_T are fixed for a given DCARP instance. The case of all artificial links in the optimal solution must be a matching of the auxiliary graph G_x : each vertex in G_x must link with exactly one other vertex and each vertex can only be linked once in the G_x . Let C_{links}^* denote the total costs of all artificial links in the optimal solution of the DCARP instance. Then, C_{links}^* must be greater than or equal to the cost of the solution to the minimum cost matching problem on G_x , which we denoted as C_{MCMP} . Thus, the following inequality must be satisfied:

$$C^* = C_T + C_{links}^* \geq C_T + C_{MCMP} = NMLB \quad (6.4)$$

which completes the proof. □

6.3.2 An Example of NMLB Approach

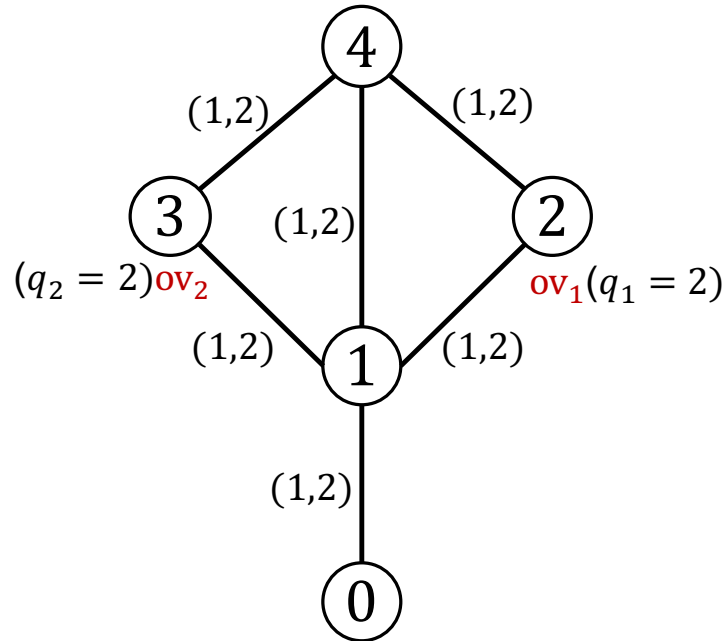
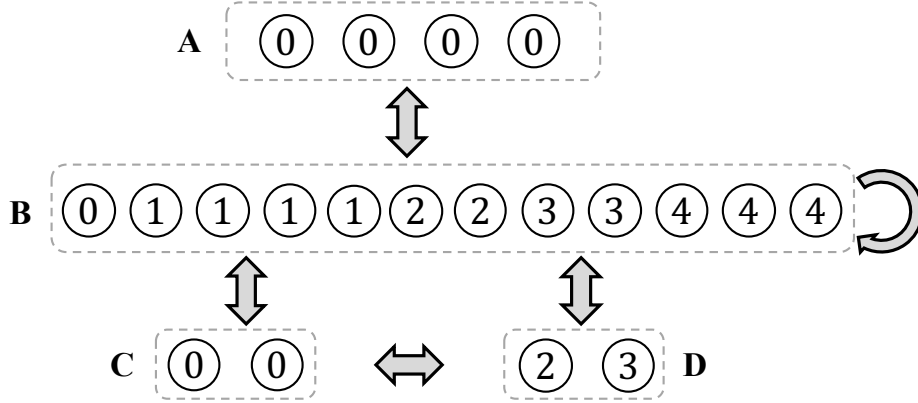


Figure 6.1: An example of a DCARP instance. Node ‘0’ represents the depot. Values in brackets denote the cost and demand of the corresponding edge.

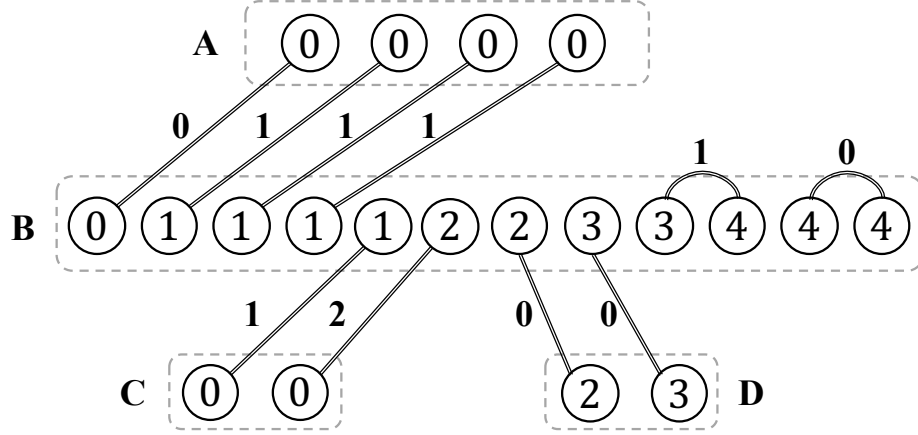
Example 1. A numerical example of a DCARP instance is shown in Figure 6.1, in which the node ‘0’ denotes the depot, and the cost and demand of each edge are provided in brackets. In our example, the traversing cost is equal to the serving cost for all tasks. The full capacities Q of the vehicle are equal to 4. Two outside vehicles, i.e., ov_1 and ov_2 , stop at nodes ‘2’ and ‘3’, respectively, with remaining capacities $q_1 = q_2 = 2$ in the example.

The optimal solution for this instance has the following routes: $\{1 : v_2 \Rightarrow v_1 - v_0, 2 : v_3 \Rightarrow v_1 - v_0, 3 : v_0 \Rightarrow v_1 - v_4 \Rightarrow v_2 - v_1 - v_0, 4 : v_0 - v_1 \Rightarrow v_4 \Rightarrow v_3 - v_1 - v_0\}$, where “ $\Rightarrow$ ” denotes that the edge between two nodes is a task, and the “ $-$ ” denotes the traversed edge between two nodes. Thus, the optimal solution’s cost for this example is 14.

The total demand for all tasks is $W_T = 12$. Thus, the minimum number of new vehicles dispatched from the depot is $k_0 = 2$. Therefore, we can construct the auxiliary graph G_x for the given instance as shown in Figure 6.2a including four different sets. Set **A** includes



(a) Graph G_x for example 1 as shown in Figure 6.1. It has links between all pairs of nodes in set A and B, set B and C, set B and D, set C and D, and between nodes in set B, as shown with big grey arrows.



(b) The solution of MCMP for the graph G_x of example 1.

Figure 6.2: Example of applying NMLB approach in a DCARP instance.

$2k_0 = 4$ copies of the depot. Set **B** includes all nodes in G_R , and each node is duplicated by its degree copies. Set **C** includes $N_{ov} = 2$ copies of the depot. Set **D** includes the nodes where outside vehicles stop. The big grey arrows indicate the links between all pairs of nodes in the two sets.

The solution for the MCMP on G_x is shown in Figure 6.2b with solid lines. Thus, the total costs for the solution of MCMP, i.e., C_{MCMP} , is equal to 7. Therefore, the lower bound $NALB$ for this example is 13, since the total services costs are $C_T = 6$.

6.3.3 Graph Pruning Strategy

Even though the above approach can obtain a lower bound for the DCARP instance, the process of solving the minimum cost matching problem on G_x is computationally expensive when the number of tasks in the instance is large. This is because there will be a large number of nodes in the set $\mathbf{B}$ of G_x for instances with many tasks and the time complexity of “Blossom algorithm¹” [35] for MCMP is $O(|V_x|^3)$. However, since the degree of many nodes in G_R is greater than 1 resulting in multiple copies of these nodes are in the set $\mathbf{B}$, many nodes in set $\mathbf{B}$ probably match their own copies in the final solution of MCMP. Thus, it is possible to exclude the nodes which definitely will match their copies in the final solution of MCMP to reduce the number of nodes in the set $\mathbf{B}$. Therefore, we propose a pruning strategy in this subsection to mitigate the computational burden of solving the MCMP.

To remove the nodes which will definitely match their copies in set $\mathbf{B}$ of G_x in the final MCMP solution, it is necessary to identify those nodes in set $\mathbf{B}$ which are possible to match to nodes in sets $\mathbf{A}$, $\mathbf{C}$ and $\mathbf{D}$ first. Then, the remaining nodes in set $\mathbf{B}$ can be pruned to remove any duplicates and reduce the size of set $\mathbf{B}$. The pruning algorithm for set $\mathbf{B}$ is presented in Algorithm 9 for obtaining a new set $\mathbf{B}'$ before solving the MCMP.

First, to identify which nodes in set $\mathbf{B}$ could potentially match the nodes in sets $\mathbf{A}$, $\mathbf{C}$ and $\mathbf{D}$, we need to consider the $2k_0 + N_{ov}$ depots in sets $\mathbf{A}$ and $\mathbf{C}$ as well as the nodes of the outside vehicles in set $\mathbf{D}$. Therefore, in Line 2-3 of Algorithm 9, we sort all nodes in $\mathbf{B}$ according to their minimum costs from the depot and add the nearest $2(k_0 + N_{ov})$ nodes into $\mathbf{B}'$ because the outside vehicles' nodes at most match to N_{ov} nodes from the nearest $2(k_0 + N_{ov})$ nodes from $\mathbf{B}$, and the remaining $2k_0 + N_{ov}$ nearest nodes are enough for matching the $2k_0 + N_{ov}$ depots in $\mathbf{A}$ and $\mathbf{C}$. Then, for each outside vehicle node, we sort the whole set $\mathbf{B}$ according to their costs to the outside vehicle node. For each node in the nearest

¹We use the python library (Networkx) to solve the MCMP.

Algorithm 9: Pseudo-code of pruning the set $\mathbf{B}$.

Input: Graph G_R , set $\mathbf{B}$

- 1 Set $\mathbf{B}' = \emptyset$.
 - 2 Sort nodes in $\mathbf{B}$ according to their minimum costs from the depot.
 - 3 Select the front $2(k_0 + N_{ov})$ nodes in $\mathbf{B}$ and add them to $\mathbf{B}'$.
 - 4 **for** each $sp_i \in V_{ov}$ **do**
 - 5 Set $\mathbf{B}_{sort}$ by sorting nodes in $\mathbf{B}$ according to their minimum costs to sp_i .
 - 6 **for** each $v_i \in \mathbf{B}_{sort}[1 : N_{ov}]$ **do**
 - 7 **if** $v_i \notin \mathbf{B}'$ **then**
 - 8 Add v_i to $\mathbf{B}'$
 - 9 Set $X = \mathbf{B} \setminus \mathbf{B}'$.
 - 10 **for** each $v_i \in X$ **do**
 - 11 **if** v_i has even number of copies **then**
 - 12 Remove all v_i in X .
 - 13 **if** v_i has odd number of copies **then**
 - 14 Just keep one v_i in X .
 - 15 $\mathbf{B}' = \mathbf{B}' \cup X$.
- Output:** $\mathbf{B}'$
-

N_{ov} nodes of the sorted $\mathbf{B}$, if it has not been included in set $\mathbf{B}'$, then add it into the set $\mathbf{B}'$, which are presented in Line 4-8 of Algorithm 9. This is because the remaining $N_{ov} - 1$ outside vehicles might match the first $N_{ov} - 1$ nearest nodes, and each outside vehicle node at least will match its N_{ov}^{th} nearest node. Thus, we add the first N_{ov} nearest nodes to set $\mathbf{B}'$ which could guarantee the remaining nodes will not match the outside vehicles' nodes in set $\mathbf{D}$.

Then, we define a set $\mathbf{X}$ consisting of nodes that definitely match nodes in set $\mathbf{B}$ in the solution of MCMP. It is obtained by removing the set $\mathbf{B}'$ from the set $\mathbf{B}$ as presented in Line 9. Each node will first match to the node, which is one copy of this node, in the solution of MCMP. This is proved in Theorem 2. Thus, for each node in set $\mathbf{X}$, if it has an even number of copies, we remove all copies of this node, while if it has an odd number of copies, we keep one copy of this node in X , as shown in Line 10-14.

Finally, the sets $\mathbf{X}$ and $\mathbf{B}'$ are combined to be the new set $\mathbf{B}$ for constructing the auxiliary

graph G_x and solving the MCMP to obtain the lower bound.

Theorem 2. *Let $X = \{v_1^1, v_1^2, \dots, v_1^{d_1}, \dots, v_n^1, v_n^2, \dots, v_n^{d_n}\}$ be a node set, in which the distance between each two nodes in the subset $\{v_i^1, v_i^2, \dots, v_i^{d_i}\}$ is 0. Then, in the solution of MCMP on the complete graph constructed based on X :*

- *If d_i is an even number, the pairs $\{(v_i^1, v_i^2), \dots, (v_i^{d_i-1}, v_i^{d_i})\}$ must be in the MCMP's solution.*
- *If d_i is an odd number, the pairs $\{(v_i^1, v_i^2), \dots, (v_i^{d_i-2}, v_i^{d_i-1})\}$ must be in the MCMP's solution.*

Proof. For the case that d_i is an even number, if one pair $(v_i^p, v_{j_1}^{q_1})$ exists in the MCMP's solution, it must have another pair $(v_i^{p+1}, v_{j_2}^{q_2})$ among the MCMP's solution (suppose p is the odd number). However, $\text{dis}(v_i^p, v_{j_1}^{q_1}) + \text{dis}(v_i^{p+1}, v_{j_2}^{q_2})$ is greater than $\text{dis}(v_i^p, v_i^{p+1}) + \text{dis}(v_{j_1}^{q_1}, v_{j_2}^{q_2})$ due to the triangle inequality because $\text{dis}(v_i^p, v_i^{p+1}) = 0$ so that they can be regarded as one point. Therefore, the pairs $\{(v_i^p, v_i^{p+1}), (v_{j_1}^{q_1}, v_{j_2}^{q_2})\}$ must be better than $\{(v_i^p, v_{j_1}^{q_1}), (v_i^{p+1}, v_{j_2}^{q_2})\}$. This is shown with a demonstration in Figure 6.3. Therefore, the nodes in $\{v_i^1, v_i^2, \dots, v_i^{d_i}\}$ must have the pairs $\{(v_i^1, v_i^2), \dots, (v_i^{d_i-1}, v_i^{d_i})\}$ in the MCMP's solution.

For the case that d_i is an odd number, if we exclude one node, such as $v_i^{d_i}$, the number of nodes in set $\{v_i^1, v_i^2, \dots, v_i^{d_i-1}\}$ becomes even, and it is the same as the above case, which completes the proof. $\square$

6.4 Experimental Studies

In this section, the tightness of the lower bound was evaluated first on a set of DCARP instances with known global optima. Then, we evaluated the optimization algorithm MAENS-SBBA generated in Chapter 5 using the newly introduced *approximate percentage deviation*

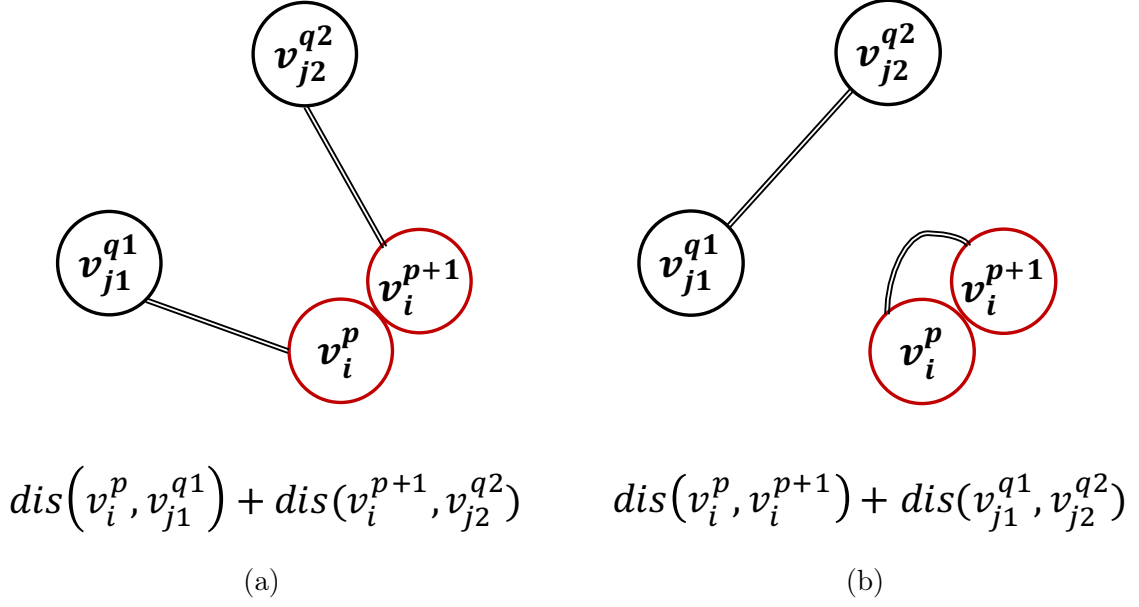


Figure 6.3: A demonstration of proof for Theorem 2.

based on NMLB on another set of more complex DCARP instances for which the global optima are unknown. Besides, the efficiency of the proposed NMLB approach with the graph pruning strategy is also evaluated on these test instances.

6.4.1 Experimental Settings

The DCARP instances used in our experiments are all generated based on a static CARP benchmark instance using the designed DCARP simulation system in Section 2.3. Firstly, to evaluate the tightness of the NMLB, a set of DCARP instances with known global optima are required. However, the global optimum for most DCARP instances is hard to be derived. Therefore, we employed the *gdb* [36] static CARP instances whose problem dimensionality are relatively low, and their global optima have been established in the literature, and generated a set of special DCARP instances whose global optima can be easily derived. For a static CARP instance, we execute its global optimal solution in the simulation system. Then, we stop vehicles in a random time point to generate a new DCARP instance and keep all

traversing costs of edges in the map unchanged. The demands of all tasks also remain the same. Thus, the service routes of the global optimal solution for the new DCARP instance remain consistent with those originally optimized from the static CARP instance.

However, the above generated DCARP instances are considered relatively straightforward for optimization algorithms studied in Chapter 3 and Chapter 5. Therefore, we calculated the lower bound for all generated DCARP instances in Chapter 5 and evaluated the performance of MAENS-SBBA based on the lower bound.

The proposed NMLB approach and the graph pruning strategy were both implemented in Python using **NetworkX** library [42] in our experiments since the algorithm for solving MCMP is available in the **NetworkX** library. The generated instances and the source code of the proposed algorithms are available on Github². The experimental results were obtained using an Intel Core i7 3.5 GHz.

6.4.2 Tightness of the NMLB

The tightness of the proposed NMLB is evaluated on a set of DCARP instances with known global optima based on the static CARP instances (*gdb*). To measure the tightness of this lower bound, we introduce the “*Gap*” metric, represented as the disparity between the NMLB and the global optimum and calculated using the following equation:

$$Gap = \frac{OPT - LB}{OPT} \times 100\% \quad (6.5)$$

The computational results are summarized on the Table 6.1. It is clear that in most created DCARP instances, the gap is zero. Only two created instances’ gap are greater than 10%,

²<https://github.com/HawkTom/NMLB>

Table 6.1: The gap between the obtained lower bound and the global optimum, i.e., Gap calculated by Eq. 6.5, on DCARP instances created based on the *gdb* static dataset. N_n , N_v and N_t denote the number of nodes, the number of required vehicles and the number of tasks, respectively. $NMLB$ and OPT represent the NMLB and the global optimal solution's total costs for the DCARP instance.

Map	N_v	N_n	N_t	$NMLB$	OPT	$Gap(\%)$
<i>gdb1</i>	5	12	18	239	245	2.4
<i>gdb2</i>	6	12	22	299	299	0
<i>gdb3</i>	5	12	18	222	222	0
<i>gdb4</i>	4	11	12	183	196	6.6
<i>gdb5</i>	6	13	20	294	294	0
<i>gdb6</i>	5	12	16	206	209	1.4
<i>gdb7</i>	5	12	15	228	241	5.4
<i>gdb8</i>	10	27	38	235	287	18.1
<i>gdb9</i>	10	26	43	215	241	10.8
<i>gdb10</i>	4	12	18	203	203	0
<i>gdb11</i>	5	22	35	303	303	0
<i>gdb12</i>	7	13	18	329	359	8.4
<i>gdb13</i>	6	10	20	421	421	0
<i>gdb14</i>	5	7	17	78	78	0
<i>gdb15</i>	4	7	15	42	42	0
<i>gdb16</i>	5	8	18	88	88	0
<i>gdb17</i>	5	8	20	64	64	0
<i>gdb18</i>	5	9	31	147	147	0
<i>gdb19</i>	3	8	8	47	47	0
<i>gdb20</i>	4	11	16	94	94	0
<i>gdb21</i>	6	11	28	137	137	0
<i>gdb22</i>	8	11	30	143	143	0
<i>gdb23</i>	10	11	52	230	230	0

i.e., *gdb8* and *gdb9*, and the gap of remaining instances are all smaller than 10%. Thus, we can conclude that our proposed NMLB can obtain a notably tight lower bound in comparison to the global optimal value on these small DCARP instances. To investigate the relation between the tightness of the lower bound and various characteristics of the DCARP instance, we presented the number of nodes in the corresponding *required graph* (G_r), the number of tasks on the instance, the least number of required vehicles, and the Gap in the Figure 6.4. We can observe from the figure that instances featuring more nodes in the *required graph*, more number of tasks and more number of vehicles had in general comparatively larger gaps between the NMLB and the global optimum.

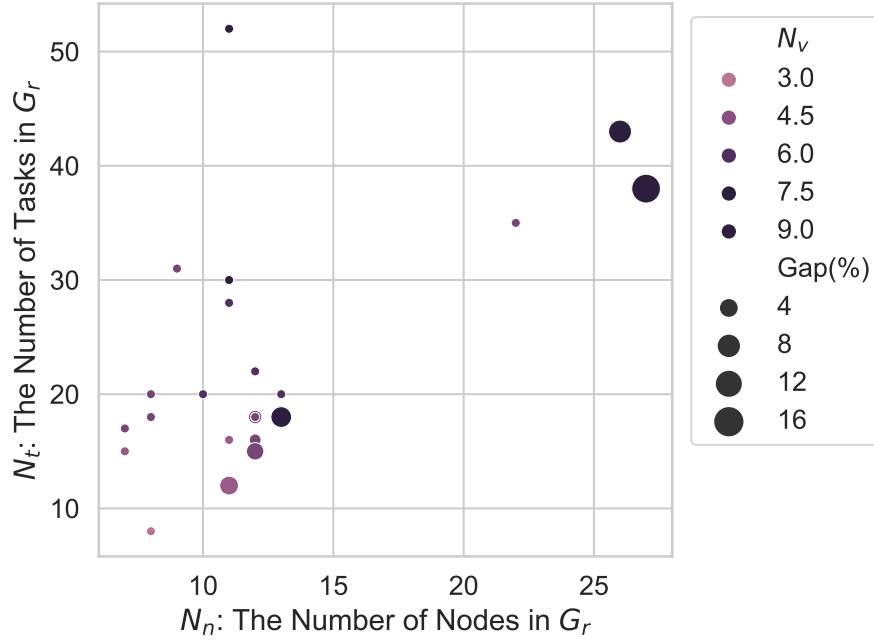


Figure 6.4: The demonstration of the relation between the tightness of the lower bound and the characteristics of the DCARP instance. Points with bigger size indicate instances with larger value of Gap .

6.4.3 Performance Measurement for the Optimization Algorithm

In our study, we proposed the virtual task and the solution building block strategy integrating them with the established MAENS [113] with virtual-task strategy. A comprehensive performance comparison with other optimization algorithm has been provided in Chapters 3 and 5. In this study, we assessed its performance relative to the proposed lower bound on all DCARP instances generated in Chapter 5.

On the basis of the proposed NMLB, we defined the following performance metric, termed as *Approximate Percentage Deviation* as presented in the Equation (6.6), where the $Best^*$ is the best objective value obtained by the optimization algorithm.

$$APD = \frac{Best^* - LB}{LB} * 100\% \quad (6.6)$$

We calculated the lower bound and the APD for all generated DCARP-OC and DCARP-CT instances. Due to page limitations, we sorted all instances in ascending order based on the APD value, and the computational results on 20 instances, selected at equal intervals, are presented in Table 6.2, with the left part dedicated to DCARP-OC instances and the right part for DCARP-CT instances. The *Rank* columns in the table represent the rank of the instance among all DCARP instances for the corresponding type. For each instance, we also provide the cost ($Best^*$) of the median optimal solution from 25 independent runs and the derived NMLB in the table. The APD is calculated based on these two values using Eq. (6.6). A box plot in Figure 6.5 visualizes the distribution of all APD values for both DCARP-OC and DCARP-CT instances. It is evident that the APD for DCARP-CT instances is marginally larger than that of DCARP-OC instances. From these two tables, most DCARP-OC and DCARP-CT instances have an APD greater than 50%. This implies that there exists potential for further optimization enhancement for our algorithm in these DCARP instances. However, since we have no knowledge about the theoretical global optimum, it is also possibly because of the large gap between lower bound and the theoretical global optimum, particularly for those instances with smaller number of tasks which are comparatively simpler to tackle. Conversely, it is certainly sure that our algorithms are significantly effective for instances with a smaller APD , such as the first DCARP-OC and DCARP-CT instances in the table.

6.4.4 Efficiency of the Graph Pruning Strategy

The computational time for the original NMLB approach (T_O) and NMLB approach with the graph pruning strategy (T_P) to calculate the lower bound are also presented in Tables 6.2. It is clear that the graph pruning strategy significantly reduces the computational time for calculating the lower bound in almost all testing DCARP instances.

Table 6.2: The approximate percentage deviation performance, i.e., APD , for the MAENS-SBBA on a subset of DCARP-OC (*left*) and DCARP-CT (*right*) instances used in Chapter 5, calculated by Equation (6.6). *Rank* represents the rank of the instance after sorting by the value of APD . N_n represents the number of nodes in the corresponding required graph. N_t represents the number of tasks on the instance. $NMLB$ represents the lower bound obtained by NMLB approach. $Best^*$ represents the median best solution's cost obtained by the MAENS-SBBA over 25 independent runs on the corresponding instance. T_o and T_p denote the computational time (unit: *sec*) for the calculating the lower bound with and without pruning strategy, respectively.

Rank	N_T	T_o	T_p	$Best^*$	$NMLB$	$APD(\%)$	Rank	N_T	T_o	T_p	$Best^*$	$NMLB$	$APD(\%)$
1	179	4.28	0.84	380382	360882	5.40	1	76	61.86	8.61	863161	624551	38.21
43	139	8.40	1.64	439362	323398	35.86	49	104	30.72	5.07	831939	524768	58.53
85	88	1.31	0.44	329835	232241	42.02	97	47	8.46	1.94	8308	4980	66.83
127	71	14.52	3.40	644268	437556	47.24	145	132	6.25	1.96	10468	6093	71.80
169	152	31.98	5.91	614301	403042	52.42	193	141	39.29	9.78	1076616	608392	76.96
211	91	19.69	4.60	840052	532753	57.68	241	151	12.16	4.35	715780	398412	79.66
253	169	4.94	1.72	413742	256359	61.39	289	63	53.58	10.67	1131919	617348	83.35
295	37	2.23	0.72	6929	4164	66.40	337	67	1.60	0.97	13047	6982	86.87
337	54	7.68	3.07	632641	374203	69.06	385	96	2.60	1.08	9911	5230	89.50
379	35	2.48	0.73	10004	5739	74.32	433	70	1.05	0.82	8551	4423	93.33
421	73	3.14	0.99	443601	248117	78.79	481	68	6.11	2.57	15872	8073	96.61
463	49	1.04	0.28	302271	165128	83.05	529	44	2.19	0.91	11633	5788	100.98
505	45	0.16	0.05	2328	1244	87.14	577	22	1.78	0.82	9224	4496	105.16
547	118	1.81	0.49	5258	2746	91.48	625	63	41.34	13.49	1147213	549029	108.95
589	137	23.54	5.96	1107698	563203	96.68	673	160	35.52	10.24	1365296	642005	112.66
631	83	3.93	1.26	468524	231885	102.05	721	75	7.78	3.06	15088	6967	116.56
673	39	0.47	0.32	5820	2794	108.30	769	32	1.23	0.86	9943	4474	122.24
715	101	3.21	1.40	9548	4427	115.68	817	73	18.68	11.48	1244733	544477	128.61
757	138	0.76	0.32	6314	2814	124.38	865	40	33.77	8.59	1304265	552014	136.27
799	42	2.40	2.02	14849	6284	136.30	913	63	1.59	1.11	12911	5267	145.13
841	53	0.69	0.34	5867	2164	171.12	961	39	3.89	2.74	15556	5742	170.92

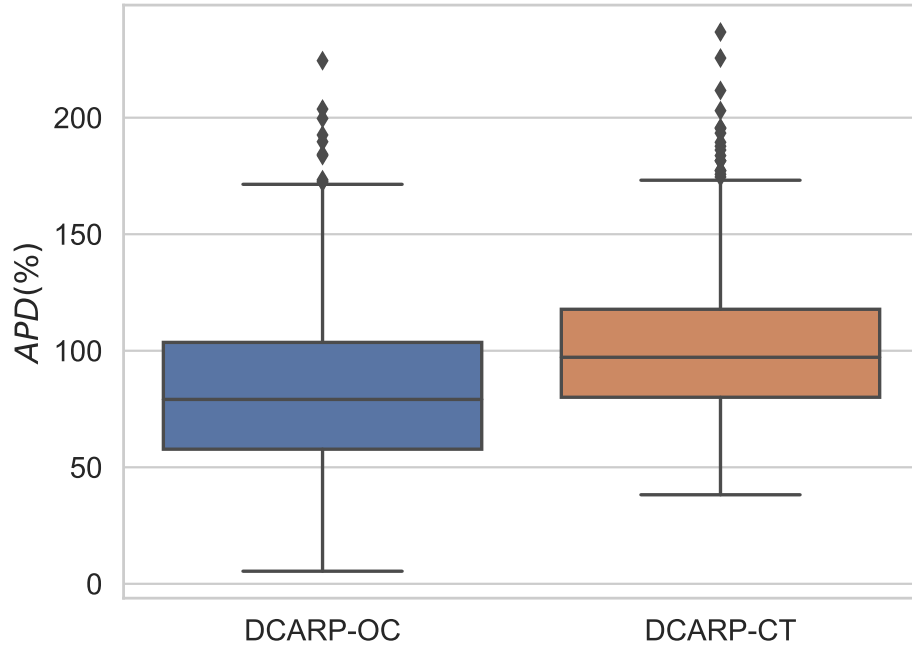


Figure 6.5: The distribution of all APD for DCARP-OC and DCARP-CT instances.

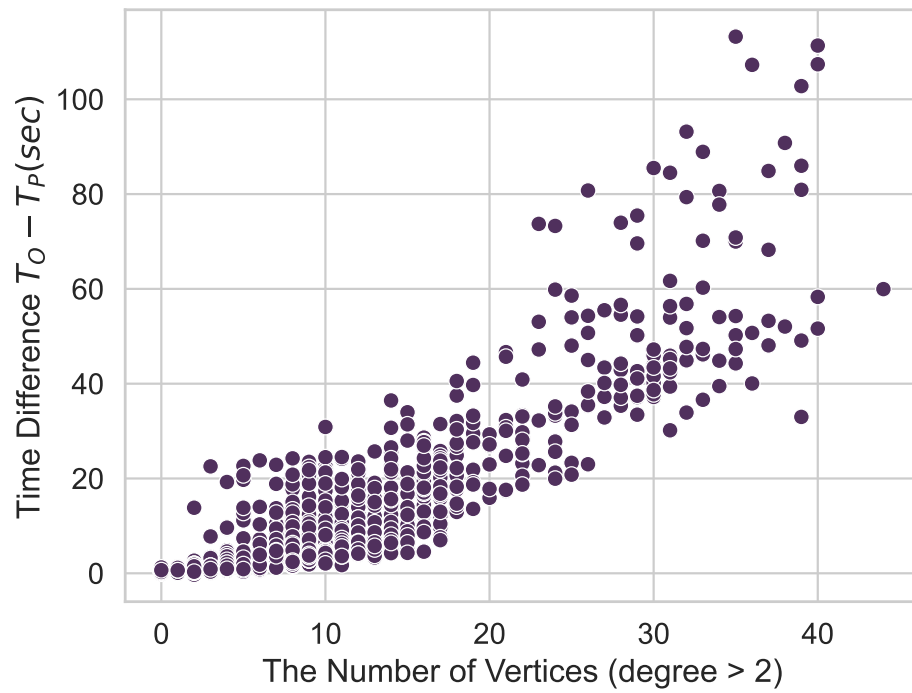


Figure 6.6: Relationship between the difference of computational time and the number of vertices in G_R whose degree are greater than 2.

In the proposed graph pruning strategy, we remove a subset of nodes in the constructed auxiliary graph, which will definitely match its copies in the solution of MCMP. Thus, if a vertex has more copies in the auxiliary graph, more vertices will be removed by the pruning strategy. For each vertex in G_R , the number of its copies in the auxiliary graph is equal to its degree in G_R . Since one vertex is likely to be pruned only if it has at least two copies in the auxiliary graph, we calculated the number of vertices whose degrees are greater than 2 in G_R of all DCARP instances. The results are presented in Figure 6.6, in which the time difference is more or less linearly correlated with the number of vertices whose degrees are greater than 2.

6.5 Summary

In this chapter, we mainly focus on measuring the performance of proposed meta-heuristic algorithms relative to the global optimum. Since the theoretical global optimum of most DCARP instances is not known, we derived a node matching lower bound (NMLB) as the alternative to the unknown global optimum to assess the quality of solutions. As a result, we used the derived lower bound to evaluate the performance of meta-heuristic algorithms for DCARP.

To derive a tight lower bound for DCARP instances, we considered the *required graph* which is a completed graph constructed by the depot and all nodes linking at least one task in the original DCARP instance. Then, our proposed node matching lower bound (NMLB) is based on the idea that the number of times a node can be assigned to link different tasks, starting point, or the ending point of a route cannot exceed its degree in the *required graph*. To handle the routes belonging to the outside vehicles, which either directly return to the depot without serving any tasks or continue serving remaining tasks, we modeled the lower

bound derivation problem as a minimum cost matching problem (MCMP). The solution to the constructed MCMP is the case with minimum costs for adding the *artificial links* in the *required graph*. In addition, to enhance the computational efficiency of our proposed NMLB, we further proposed a pruning strategy.

The experimental studies evaluated the tightness of NMLB, i.e., the gap between obtained lower bound and the global optimum, in some specifically designed DCARP instances. The results demonstrate that the proposed NMLB provides a tight lower bound for small DCARP instances in small DCARP instances. Then, we used the performance measurement metric, i.e., APD, to evaluate the performance of our proposed dynamic optimization algorithm. The large APD in these complex DCARP instances with more tasks suggest that our algorithm may still have potential for further improvement on these DCARP instances. However, this also could be attributed to the NMLB not being tight enough for these DCARP instances. Given that the true optima for these instances are unknown, it is impossible to determine the tightness of the NMLB for these specific problem instances. In addition, our experimental studies also demonstrated that the NMLB approach benefits a lot from the pruning strategy for DCARP instances with more vertices whose degrees are greater than 2 in the *required graph*.

The performance of our algorithms was successfully evaluated from a theoretical perspective through the derived lower bound in this chapter. This addresses the *RQ4*: “How can we derive a lower bound for evaluating the DCARP solution quality rather than relying solely on comparison with baseline algorithms?”. However, the practical performance of our algorithm in scenarios with real traffic conditions still remains unknown. Therefore, in the next chapter, we will shift our performance evaluation toward applying our algorithms to DCARP scenarios with real traffic conditions to answer *RQ5*: “How can we evaluate and how well do our proposed DCARP algorithm perform in real-world scenarios in addition to artificial benchmarks?”.

Chapter Seven

Performance Evaluation with Real-world Traffic Simulation

This chapter is based on following publication: *Tong, Hao, Leandro L. Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao. "Benchmarking dynamic capacitated arc routing algorithms using real-world traffic simulation." In 2022 IEEE Congress on Evolutionary Computation (CEC), pp. 1-8. IEEE, 2022. <https://doi.org/10.1109/cec55065.2022.9870399>.*

7.1 Introduction

The proposed algorithms have been successfully evaluated in DCARP scenarios generated through our designed simulation benchmarking system based on the static CARP instances and adding dynamic features. However, such artificially designed benchmark for evaluating DCARP optimization algorithms' efficacy is likely to miss partial problem characteristics of real-world applications, and it is unclear whether such artificial generators produce problem instances that are similar to real world ones. In the literature, there is still no benchmarking framework with real traffic conditions for DCARP so that it is therefore necessary to design one using more realistic traffic simulations which are built on actual traffic data for evaluating the efficacy of DCARP algorithms.

Simulation of Urban MObility (SUMO) is an open-source¹, highly portable, and continuous traffic simulation software [65]. There are three main parts in SUMO, including *Network building*, *Demand modeling* and *Simulation*. The *Networking building* and *Demand modeling* are defined by users, which provide the road networks and the simulated vehicles (vehicle types, and vehicles' routes), respectively. Then, the *Simulation* can simulate the traffic conditions according to the provided network and traffic demands (i.e., vehicles). A traffic control interface (TraCI)² is provided by SUMO which gives access to retrieve values from the simulated objects and manipulate their behavior online.

SUMO is able to simulate all traffic objects, such as vehicles and traffic lights, according to the provided real road network and real traffic, which constructs a traffic environment reflecting the real-world traffic. Thus, it is possible to use it to build a virtual environment with real traffic conditions for DCARP. Since Traci in SUMO can directly obtain the online data and control the simulated object in an online manner, it is possible to obtain the DCARP

¹<https://sumo.dlr.de/docs/index.html>

²<https://sumo.dlr.de/docs/TraCI.html>

instance from the simulation process and provide it to DCARP optimization algorithms for an improved new schedule.

Therefore, this motivated us to design a benchmarking framework based on the SUMO for DCARP which can be a standard simulation platform for evaluating DCARP algorithms in real traffic conditions besides the artificial benchmarks once we have real road maps and traffic data. We develop the first benchmarking framework in the literature being capable of using real traffic scenarios based on the SUMO. The proposed benchmarking framework builds a bridge between DCARP optimization algorithms and SUMO such that DCARP optimization algorithms can benefit from the real traffic environment simulation in SUMO and be evaluated with real traffic data. This answers the *RQ5*: “How can we evaluate and how well do our proposed DCARP algorithm perform in real-world scenarios in addition to artificial benchmarks?”. Dynamic changes in the real-world traffic environment are much more complex and unpredictable, which are more realistic compared with our previous artificial simulations. Therefore, the generated test DCARP scenarios with real traffic data contain characteristics that might not be captured by artificially generated data so that researchers can benefit from such insights and improve algorithms for the DCARP optimization.

Our contributions are as follows:

- We propose the first DCARP scenario benchmarking framework containing real traffic data which can be used to evaluate the efficacy of algorithms in a real-world environment. The framework is able to integrate the solution produced by DCARP algorithms into a transportation environment that simulates real traffic conditions.
- To demonstrate the generalization ability and flexibility of the proposed benchmarking framework, we construct 12 different dynamic scenarios with real traffic data in the designed simulation platform including different locations of depot, tasks and different service times of a day. These dynamic scenarios can be used as a benchmark, containing

the real traffic information in contrast to artificial datasets for evaluating DCARP algorithms in the future.

- Based on the proposed simulation platform, we applied the proposed DCARP optimization algorithm to both SUMO-based DCARP-OC and DCARP-CT scenarios. The results helped to strengthen that the DCARP optimization is necessary for increasing the service efficiency. Moreover, we also compared different DCARP optimization algorithms using this framework, and experimental results demonstrated that our proposed algorithms perform better than existing optimization algorithms in DCARP scenarios with real traffic conditions.
- Based on the DCARP scenarios with real traffic conditions, we conducted further analysis from much more practical perspectives which is not suitable with artificial benchmarks. We focused on the algorithmic stability over runs, the changes in vehicle paths and the variations of vehicles working times after applying with our proposed dynamic optimization algorithm. The empirical findings show that proposed algorithm is highly stable for solving DCARP instances and can maintain a balanced vehicle workload. However, our algorithm is likely to cause significant changes to the vehicle paths, especially in DCARP-CT scenarios.

The remainder of this chapter is organized as follows. Section 7.2 introduces the proposed benchmarking framework with online optimization in detail. Section 7.3 presents the constructed scenarios based on the designed platform and the results of using DCARP algorithms in all generated dynamic scenarios. Finally, Section 7.4 presents the summaries of this chapter.

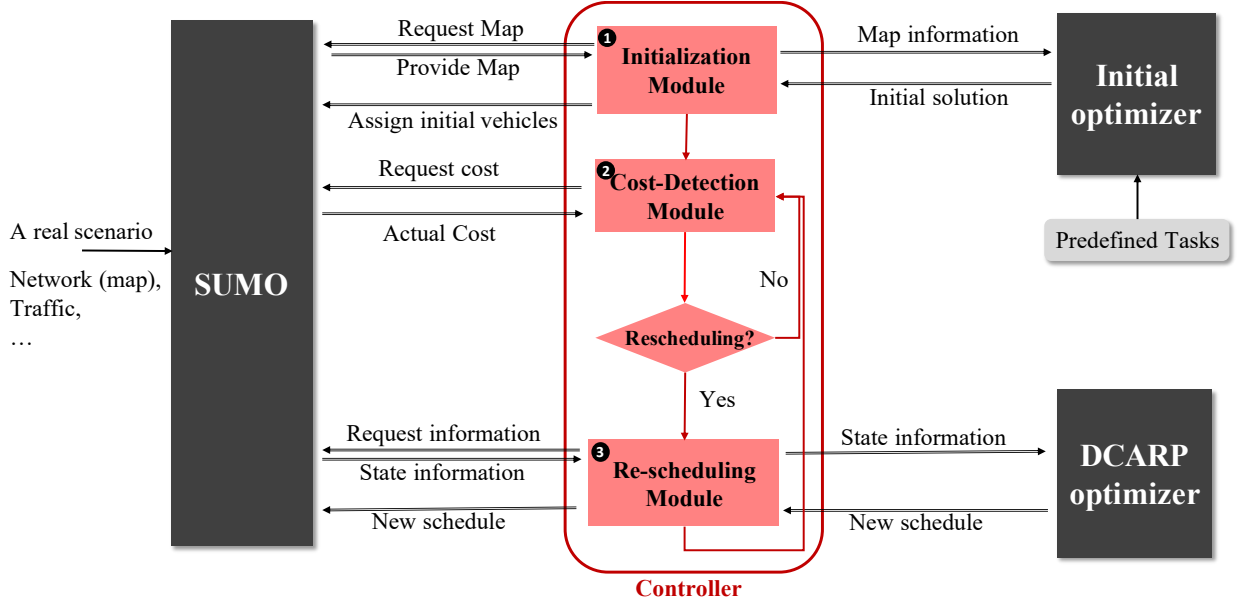


Figure 7.1: The structure of SUMO-based benchmarking framework. Our main contribution is the “*Controller*” part.

7.2 SUMO-based Benchmarking Framework

SUMO can retrieve the real road network and the online traffic data and provide information to (D)CARP optimization algorithms for high-quality executable solutions. The solutions obtained by optimization algorithms are required to be assigned and updated to the transportation environment in SUMO. Therefore, we proposed a SUMO-based benchmarking framework comprising SUMO, optimization algorithms, and the intermediate component linking the SUMO and (D)CARP optimization algorithms. The structure of the proposed SUMO-based benchmarking framework is presented in Figure 7.1.

Three different components form the framework. First, the data of the real environment for the simulation is required to be provided to SUMO, such as the road map and real traffic data. Then, the middle component, named as “*Controller*” in Figure 7.1, is the core component of this platform that transmits information between SUMO and the optimization algorithms. The final part of this benchmarking framework is the optimizer. The initial

optimizer can be any suitable practical static CARP optimization algorithm for optimizing the initial solution for the static CARP instance. The DCARP optimizer is the algorithm whose performance is required to be evaluated by the platform. The three components are independent of each other. The communication between SUMO and the controller is based on TraCI, and between the controller and the optimizers is currently based on shared files. It is worthy to mention the communication based on shared files is not ideal especially when the optimizer takes a long time to solve DCARP, as the controller would have to wait for the new schedule for a long time instead of deciding an acceptable solution on its own during the optimization. Therefore, it is valuable to employ message queue as the communication method in the future to make it more flexible.

The intermediate component *Controller* is the main component for driving the entire system. It mainly consists of three modules including *Initialization Module*, *Cost-Detection Module*, and *Re-schedule Module*:

- **Initialization Module** The initialization module is first activated before the start of the simulation. It is mainly responsible for requesting the map's road network after SUMO successfully loads the map, and then providing the network information to a predefined static CARP optimization algorithm. The algorithm feedbacks an initial solution to the initialization module. Then, the module activates the simulation process in SUMO with starting the initial vehicles and setting their service paths according to the solution.
- **Cost-Detection Module** After initial vehicles are assigned to serve tasks, and the simulation process starts running, the cost-detection module activates. A selection of dynamic events are likely to occur during the vehicles' service process, potentially affecting the cost of the current service plan. Therefore, the cost-detection module is used to collect the costs of the roads online during the simulation. Then, the cost-detection module calculates the total costs of the current service plan, which are transmitted to the next re-scheduling module to determine whether the dynamic events are too severe and deteriorate

the current service plan a lot.

- **Re-scheduling Module** The re-scheduling module is activated if the difference between the current cost provided by the cost-detection module and the last recorded cost exceeds a pre-defined threshold, such as 10% increment. This indicates that some dynamic events have happened and have severely deteriorated the current service plan. The re-scheduling module requests the necessary state information from SUMO, including the current road map, the cost of each road, the positions of in-service vehicles, the remaining capacities of in-service vehicles and the remaining tasks, to construct a new DCARP instance. The module then passes the new DCARP instance to the DCARP optimization algorithm for an updated service plan. Finally, this module updates the paths of the vehicles or assign new vehicles in SUMO based on the new schedule obtained from the DCARP optimizer to continue serving the remaining tasks.

The controller continues to run the cost-detection module until the re-scheduling process is activated again. The whole process terminates after all tasks have been served and all vehicles return to the depot.

As the traffic data provided to SUMO only contains the traffic conditions, the tasks of CARP need to be defined and constructed by ourselves. For simplicity, in our benchmarking framework, the initial tasks are required to be defined in a separate file which is then provided to the initial optimizer for an initial solution. The simulation for the new tasks event is embedded in the re-scheduling module. For simplicity, in this version, we added some new tasks into the simulated DCARP scenario to simulate the new-task events in the real world, once the re-scheduling module is activated. In the future, we will update our simulator so that it could enable adding new tasks separately from the increase in traffic events. The source code of the proposed benchmarking framework is available in Github³.

³<https://github.com/HawkTom/DCARP-SUMO-Simulation>

7.3 Experimental Studies

In this section, in order to demonstrate the generalization ability and flexibility of our proposed benchmarking framework, we generate a series of DCARP scenarios using it based on real open-source traffic data. Then, the proposed DCARP optimization algorithms are evaluated in all generated scenarios. The efficacy of proposed optimization DCARP algorithms was first evaluated in SUMO-based DCARP scenarios. Subsequently, we compared different optimization algorithms within these SUMO-based DCARP scenarios to investigate if the performance aligns with that concluded in artificially designed benchmark. This enables us to check whether the artificial benchmark is sufficient to evaluate the performance of DCARP algorithms. Finally, we analyzed the algorithm from a more practical angle by checking the stability of its performance across different runs, how much changes it causes to the vehicle paths, and how much variations in vehicle working times it introduces in SUMO-based DCARP scenarios under real-traffic conditions.

7.3.1 Experimental Setup

The actual traffic data is the source of our DCARP simulation. In our experiment, we use a set of open-source 24-hour traffic data from Dublin⁴. The traffic data comes from the Dublin SCATS dataset⁵, which consists of vehicle counts at every 6 minutes at 480 locations in the city center [40]. The road network of Dublin city is presented in Figure 7.2a, which has 6633 edges and 2895 nodes in total. We divide the whole map into five different areas including both city center and suburb areas for generating different DCARP scenarios according to the road's distribution, as shown in Figure 7.2b.

Based on the above road network, we generated 12 different initial CARP scenarios with

⁴https://github.com/maxime-gueriau/ITSC2020_CAV_impact/tree/master/Urban/Simulations/Base

⁵<https://data.gov.ie/dataset/traffic-volumes>

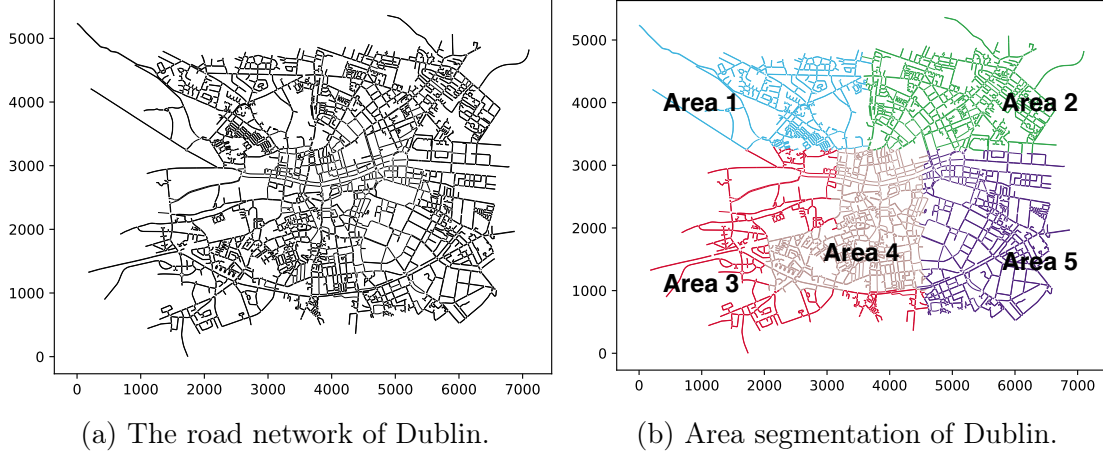


Figure 7.2: Information about road network of Dublin.

different settings for the location of the depots, the distribution of tasks and service starting time. The settings for each factor are as follows:

- Location of depots: busy area, uncrowded area.
- Distribution of tasks: whole city, busy area, uncrowded area.
- Service starting time: off-peak hours, near-peak hours.

To determine the busy degree of each area in Figure 7.2b and the peak/off-peak hours in provided traffic data, we define the “*crowded degree* (α)” for each road and the “*busy degree* (β)” for an area, as formulated by Eq. (7.1) and Eq. (7.2):

$$\alpha_i = 1 - \frac{\bar{v}_i}{v_i^{max}} \quad (7.1)$$

$$\beta = \frac{\sum_{i=1}^{N_{roads}} I(\alpha_i > 0.5)}{N_{roads}} \quad (7.2)$$

where $\bar{v}_i$ and v_i^{max} represent the average speed in the last 60 seconds and the limited maximum speed of one road i , respectively, N_{roads} denotes the total number of roads in an area and $I(\cdot)$ is the indicator function. Therefore, we calculated the *crowded degree* of each road

in the whole network for every 60 seconds. Then, the *busy degree* of each area of Figure 7.2b can be estimated by Eq. (7.2). The dynamic *busy degree* of each area for every 60 seconds is presented in Figure 7.3.

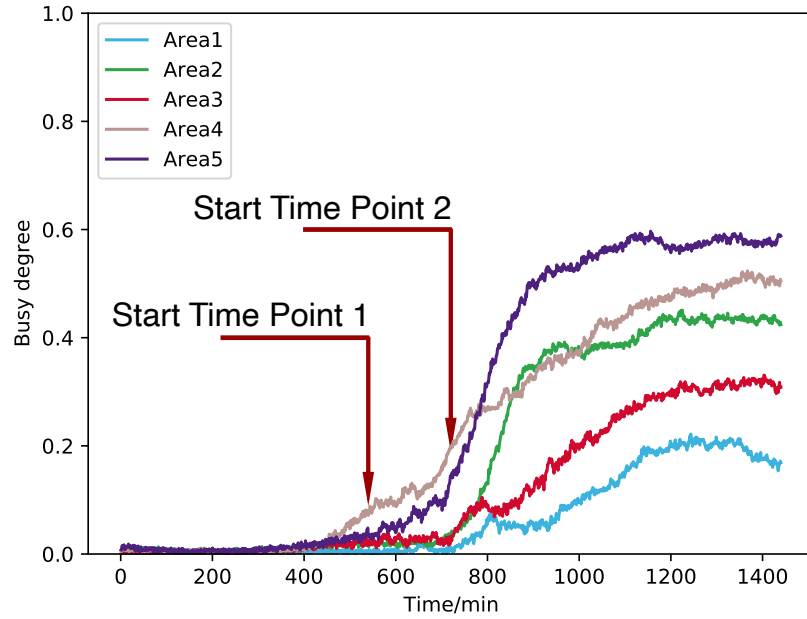


Figure 7.3: The dynamic *busy degree* of each area for every 60 seconds.

According to Figure 7.3, *Area 1* and *Area 3* are determined as the uncrowded areas, and the remaining three areas are regarded as busy areas (Figure 7.4). Therefore, we can select depots and tasks in these divided areas. In Figure 7.4, we marked two points as locations of the depot in our DCARP scenarios, where one is in the uncrowded area, and another is in the busy area. The peak and off-peak hours can also be obtained from Figure 7.3, in which a small busy degree indicates the off-peak hours. Therefore, we select two time-points, i.e., 540 and 720, to be the starting time of the simulation, as pointed in Figure 7.3. Both starting time points belong to the off-peak hours because the service in the real-world application is usually provided in off-peak time. Nevertheless, the service can last until the peak hours, so we select a starting time point from which the vehicles are likely to meet peak hours, i.e., near-peak hours.

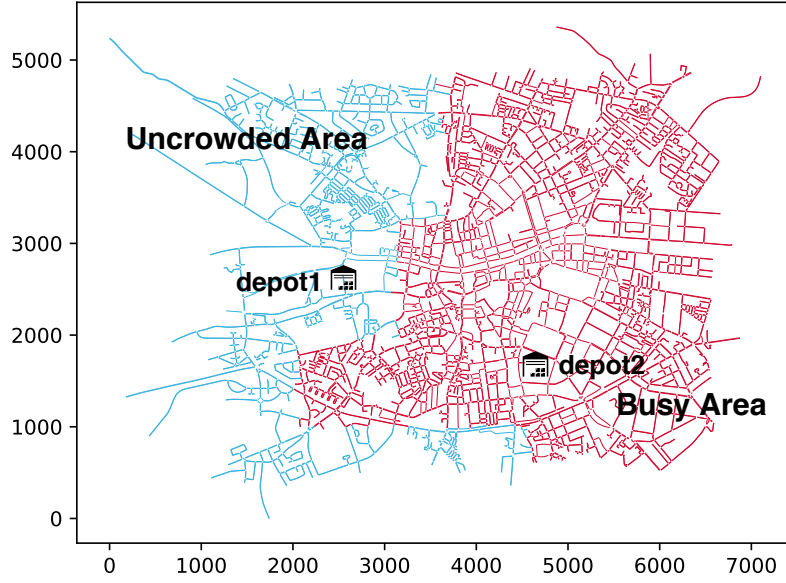


Figure 7.4: The distribution of busy area and uncrowded area. The two positions of the depot.

Besides, it is also important to determine the tasks and the service cost. As the whole map has been divided into busy and uncrowded areas, we randomly selected 200 tasks from one or multiple designated areas for each DCARP scenario. A screenshot of a DCARP scenario simulation is presented in Figure 7.5 for the scenario with a depot in the busy area, tasks distributed in the whole city and service starting from the off-peak time, containing all cars and vehicles defined in the traffic data. However, only all unserved tasks and in-service vehicles are highlighted. In the simulated traffic environment, the traffic congestion highly influences the service time. Therefore, the total time for completing all remaining tasks from the current time is defined as the cost in our DCARP scenarios, calculated by the cost-detection module every 30 seconds during the whole service process. Then, the re-scheduling module activates if the current estimated total cost is 10% more than the last recorded cost. In our experiments, we generated both DCARP-OC and DCARP-CT scenarios to evaluate and compare optimization algorithms. The dynamic events related to

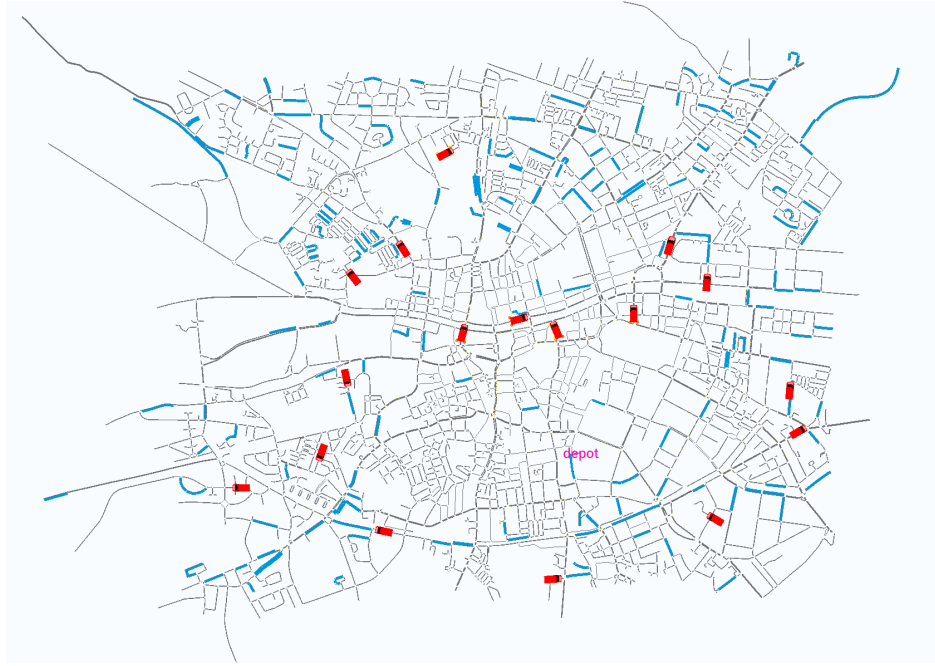


Figure 7.5: One screenshot of DCARP scenario simulation in SUMO-based benchmarking framework in our experiment. It contains all cars and vehicles defined in the Dublin real traffic data, but only all unserved tasks (in blue) and in-service vehicles (in red) are highlighted.

cost changes are determined by the traffic data itself without modifications. Additionally, to simulate newly added tasks when generating DCARP-CT scenarios, we randomly select 10 arcs which are not pre-defined tasks within the same area as initial tasks to be new tasks and add them into the simulation once the re-schedule module activates. All generated scenarios are available in Github⁶.

7.3.2 Efficacy of Algorithms in SUMO-based DCARP Scenarios

The DCARP optimization algorithms target to reduce the total cost of serving tasks in the scenario with dynamic events. In this subsection, we applied the constructed DCARP algorithm, i.e., MAENS-SBBA from Chapter 5, to the 12 different DCARP-CT scenarios to investigate the influence of the dynamic optimization on DCARP scenarios such that an

⁶<https://github.com/HawkTom/DCARP-SUMO-Simulation>

insight can be obtained as to why and when DCARP optimization algorithm works well. The termination condition for MAENS-SBBA in each round of optimization is 1 minute, to satisfy the requirement of a quick response in the real scenario.

In all different generated DCARP-CT scenarios, we recorded the total used time for vehicles to complete all tasks and return to the depot, i.e., the total cost in our scenarios. The obtained results are presented in Table 7.1, in which there are 12 scenarios in total with different settings of starting time, depot location, and distribution of tasks. The number of instances in each DCARP scenario is presented in column $\mathbf{N_I}$. These instances were produced as a result of the dynamic events that were severe enough to cause the rescheduling module to activate. $\mathbf{T_{DO}}$ and $\mathbf{T_{WDO}}$ denote the total time used after using dynamic optimization and without dynamic optimization, respectively. We also calculated the time difference between using and not using dynamic optimization in column $\mathbf{T_{Diff}}$ as calculated by Eq. (7.3):

$$T_{Diff} = \frac{T_{WDO} - T_{DO}}{T_{WDO}} \times 100\% \quad (7.3)$$

The time difference between using dynamic optimization and without dynamic optimization reflects the effectiveness of dynamic optimization algorithms. A big time difference indicates that the corresponding DCARP scenario is highly affected by the dynamic events, and such scenarios can significantly benefit from dynamic optimization algorithms. As presented in Table 7.1, scenarios with tasks distributed in the busy area of the city have the larger time difference compared with scenarios with tasks distributed in uncrowded area or the whole city. It is because the busy area is easy to have traffic jam leading to vehicle delays due to road congestion. When tasks are distributed over the whole city, the time difference is also very large. Interestingly, the starting service time has no significant influence on the time difference in our experiments, as shown in Table 7.1, which contrasts with our intuitive conjecture that the service starting from an off-peak time point will be more marginally

Table 7.1: The results of using dynamic optimization (MAENS-SBBA algorithm) and without dynamic optimization in each generated DCARP-CT scenario.

Starting Time	Depot Location	Tasks' Distribution	Index	$T_{Do}(s)$	$T_{wDo}(s)$	$T_{Diff}(s)$
Off-peak hours	centre	whole city	1	5488	134300	95.91%
		busy area	2	11350	78074	85.46%
		uncrowded area	3	5534	10856	49.02%
	suburb	whole city	4	3230	12314	73.77%
		busy area	5	5668	106168	94.66%
		uncrowded area	6	2520	2520	0
Near-peak hours	centre	whole city	7	4548	8686	47.64%
		busy area	8	4639	9780	52.57%
		uncrowded area	9	5088	10665	52.29%
	suburb	whole city	10	5679	6202	8.43%
		busy area	11	6514	14339	54.57%
		uncrowded area	12	1730	1730	0

N_I : The number of instances in this DCARP-CT scenario.

T_{Do} : Total time used after dynamic optimization. T_{wDo} : Total time used without dynamic optimization.

T_{Diff} : The time difference between using and not using dynamic optimization.

influenced by traffic congestion. Both observations can be attributed to the introduction of new tasks into scenarios which has a significant impact on the total service time. If new tasks appear in the peak time and are far away from the depot, vehicles are likely to be affected by the coming traffic jams in the peak hours when serving these new tasks, leading to a long time to serve all tasks. Finally, the influence of depot location highly depends on the starting time because the traffic difference between busy areas and uncrowded areas in off-peak hours is smaller than that of peak hours. Therefore, we can conclude that the scenarios with tasks distributed in a large area, including both busy and uncrowded areas, have a greater need for dynamic optimization, and the total cost can be greatly saved.

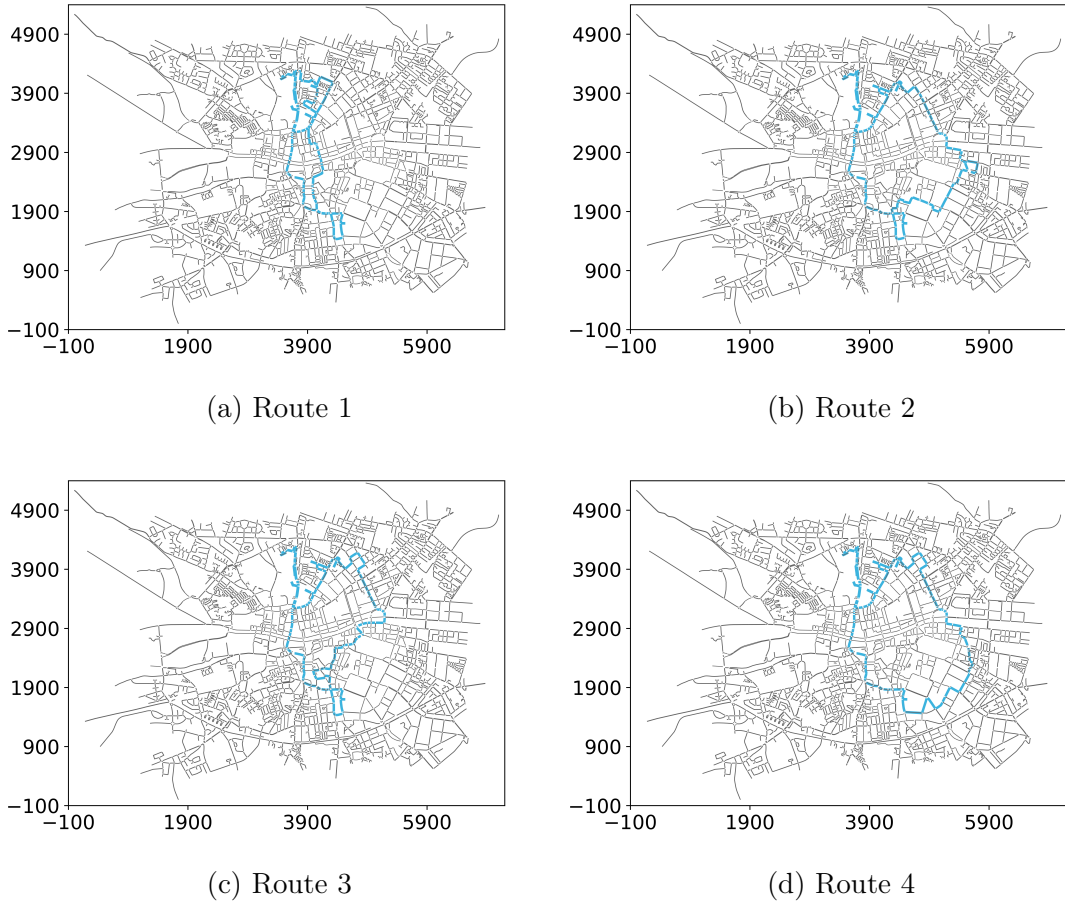


Figure 7.6: An example of one vehicle's route's updates in the scenario 1 of Table 7.1.

Besides, we also present an example of one vehicle's route's updates by dynamic optimiza-

tion in a DCARP scenario in Figure 7.6. Route 1 is its initial route, and three updates are applied to this vehicle. After the simulation begins, this vehicle starts its service according to its initially assigned route. However, the cost-detection module detects that the current service plan is affected by dynamic events, so that the re-schedule module activates. As a result, this vehicle is assigned with a new service path to save the total cost.

7.3.3 Comparing Algorithms with SUMO-based DCARP Instances

The effectiveness of using DCARP optimization algorithm has been evaluated in the DCARP scenarios with real traffic conditions in the previous subsection. Now, we regenerate a series of DCARP-OC and DCARP-CT scenarios based on 12 initial CARP scenarios for comparing different DCARP optimization algorithms under real traffic conditions. Similar to the experiments in Section 5.3, we evaluated the quality of initial solutions generated by SBBA, PS and GRD strategies, and final solutions using MAENS-SBBA and MAENS-GRD algorithms. Additionally, we also compared our optimization algorithms with a state-of-the-art optimization algorithm which was specifically designed for DCARP scenarios with general dynamic events, i.e., memetic algorithm with a new distance-based split scheme (MASDC) [60].

The comparative results in SUMO-based DCARP-OC and DCARP-CT scenarios are detailed in Table 7.2 and Table 7.3, respectively. The columns under “Strategies Comparison” present the costs of the best solution generated by the SBBA, PS and GRD strategies. The columns under “Algorithms Comparison” present the costs of the optimal solution found by MAENS-SBBA, MAENS-GRD and MASDC algorithms. Each cell specifies the “MEAN $\pm$ STD” of the best solutions obtained over 25 runs for each algorithm and the best result(s) based on statistical tests are highlighted. In both tables, the scenarios 6 and 12 are omitted because no dynamic event affects the service of vehicles in these two scenarios according to

Table 7.2: Results of generated solutions' quality for SBBA, PS and GRD strategies, and results of the best solutions obtained by MAENS-SBBA, MAENS-GRD and MASDC algorithms on SUMO-based DCARP-OC scenarios. For the strategy comparison, the results represent the total costs of the best solution produced by each strategy, with the best result highlighted. The algorithm comparison results present the "MEAN $\pm$ STD" of the best solutions acquired over 25 runs for each algorithm and the best result(s) under statistical tests are highlighted.

Scn Index	Ins Index	Strategies Comparison			Algorithms Comparison		
		SBBA	PS	GRD	MAENS-SBBA	MAENS-GRD	MASDC
1	1	21492	25204	25204	21179$\pm$73	21314 $\pm$ 96	48753 $\pm$ 1013
	2	19061	21595	21309	18790$\pm$33	18866 $\pm$ 36	39303 $\pm$ 1106
	3	4811	5342	5046	4811$\pm$0	4811$\pm$0	6065 $\pm$ 0
	4	2580	2580	2580	2580$\pm$0	2580$\pm$0	3815 $\pm$ 0
2	1	17146	18455	18336	16514$\pm$53	16517$\pm$63	34722 $\pm$ 895
	2	10781	11938	11556	10734$\pm$0	10734$\pm$0	16838 $\pm$ 323
3	1	22412	25495	25495	21734$\pm$50	21791 $\pm$ 73	46046 $\pm$ 837
	2	16511	17912	17859	16453$\pm$0	16453$\pm$0	24630 $\pm$ 327
4	1	21227	24559	24559	20652$\pm$94	20669$\pm$96	47077 $\pm$ 1119
	2	19198	23091	22891	18972$\pm$60	19159 $\pm$ 106	41502 $\pm$ 1064
	3	4094	4274	4274	4094$\pm$0	4094$\pm$0	5777 $\pm$ 0
	4	1655	1655	1655	1655$\pm$0	1655$\pm$0	1655 $\pm$ 0
5	1	20459	23282	23282	19970$\pm$91	20065 $\pm$ 136	44326 $\pm$ 770
	2	16808	19152	18708	16529$\pm$18	16617 $\pm$ 46	30694 $\pm$ 602
	3	14357	15935	15614	14301$\pm$5	14303$\pm$5	23117 $\pm$ 372
	4	9308	9698	9682	9218$\pm$0	9218$\pm$0	13133 $\pm$ 258
7	1	17668	20632	20316	17429$\pm$14	17478 $\pm$ 39	34756 $\pm$ 631
	2	12129	13104	13007	12095$\pm$0	12095$\pm$0	17900 $\pm$ 312
8	1	14299	15724	15722	13816$\pm$24	13861$\pm$42	26564 $\pm$ 530
	2	8455	10056	9758	8410$\pm$0	8410$\pm$0	11316 $\pm$ 285
	3	5711	6550	5970	5711$\pm$0	5711$\pm$0	5969 $\pm$ 0
9	1	19819	21501	21460	19284$\pm$29	19275$\pm$25	35514 $\pm$ 803
	2	15202	16856	16452	15174$\pm$0	15174$\pm$0	22180 $\pm$ 326
10	1	16655	18885	18275	16199$\pm$16	16198$\pm$23	29850 $\pm$ 550
	2	12367	14437	13767	12348$\pm$0	12348$\pm$0	18590 $\pm$ 380
	3	3304	3758	3758	3304$\pm$0	3304$\pm$0	4514 $\pm$ 0
	4	2537	2868	2868	2537$\pm$0	2537$\pm$0	3110 $\pm$ 0
	5	1051	1051	1051	1051$\pm$0	1051$\pm$0	1051 $\pm$ 0
11	1	19078	21374	21067	18766$\pm$42	18779$\pm$75	37503 $\pm$ 781

Table 7.3: Results of generated solutions' quality for SBBA, PS and GRD strategies, and results of the best solutions obtained by MAENS-SBBA, MAENS-GRD and MASDC algorithms on SUMO-based DCARP-CT scenarios. For the strategy comparison, the results represent the total costs of the best solution produced by each strategy, with the best result highlighted. The algorithm comparison results present the "MEAN $\pm$ STD" of the best solutions acquired over 25 runs for each algorithm and the best result(s) under statistical tests are highlighted.

Scn Index	Ins Index	Strategies Comparison			Algorithms Comparison		
		SBBA	PS	GRD	MAENS-SBBA	MAENS-GRD	MASDC
1	1	27232	26178	28506	22376$\pm$90	22470 $\pm$ 126	51487 $\pm$ 922
	2	10412	11390	10896	9703$\pm$0	9703$\pm$0	11134 $\pm$ 163
	3	6473	6327	6321	5266$\pm$0	5266$\pm$0	6528 $\pm$ 56
	4	5698	6308	6308	5311$\pm$1	5311$\pm$1	6114 $\pm$ 159
2	1	19699	19918	19392	17127 $\pm$ 107	17077$\pm$63	37569 $\pm$ 716
	2	15542	16861	16740	14523$\pm$27	14530$\pm$41	27161 $\pm$ 624
	3	14017	15223	14991	12741$\pm$26	12733$\pm$26	21282 $\pm$ 360
	4	10161	10827	10898	9370$\pm$0	9370$\pm$0	13638 $\pm$ 286
3	1	26669	26457	27760	22318$\pm$80	22303$\pm$80	47994 $\pm$ 801
	2	20865	22702	22183	18905$\pm$13	18947 $\pm$ 39	32743 $\pm$ 491
	3	6579	6579	6579	5750$\pm$0	5750$\pm$0	8309 $\pm$ 0
	4	4707	4681	4889	4198$\pm$0	4198$\pm$0	4469 $\pm$ 61
4	1	22617	24818	26243	21465$\pm$83	21438$\pm$136	50597 $\pm$ 1020
	2	19383	22346	23181	18862$\pm$43	18989 $\pm$ 73	40144 $\pm$ 817
	3	17578	18574	18309	16078$\pm$11	16090 $\pm$ 16	29337 $\pm$ 573
	4	22738	24169	26871	20786$\pm$95	20922 $\pm$ 118	47459 $\pm$ 1047
5	1	4943	5257	5296	4261$\pm$0	4261$\pm$0	4897 $\pm$ 6
	2	4590	5137	5021	4067$\pm$0	4067$\pm$0	4200 $\pm$ 90
	3	4590	5137	5021	4067$\pm$0	4067$\pm$0	4200 $\pm$ 90
	4	5906	5195	6020	4731$\pm$6	4730$\pm$0	5191 $\pm$ 61
7	1	19859	21583	20803	18429 $\pm$ 43	18406$\pm$35	37953 $\pm$ 714
	2	16185	17449	16374	14015$\pm$17	14066 $\pm$ 45	23799 $\pm$ 543
	3	3673	3841	3841	3403$\pm$0	3403$\pm$0	3427 $\pm$ 36
	4	4485	4140	4341	3768$\pm$0	3768$\pm$0	3862 $\pm$ 88
8	1	16958	17075	17563	14779$\pm$35	14777$\pm$35	29329 $\pm$ 610
	2	7252	9428	8556	7075$\pm$0	7075$\pm$0	8402 $\pm$ 124
	3	4001	3970	3943	3616$\pm$0	3616$\pm$0	3644 $\pm$ 68
	4	5184	5372	5451	4593$\pm$0	4593$\pm$0	5038 $\pm$ 135
9	1	22118	23856	23322	20638 $\pm$ 43	20600$\pm$32	38674 $\pm$ 531
	2	4390	5086	5778	4144$\pm$0	4144$\pm$0	5663 $\pm$ 29
	3	6016	5789	6042	4926$\pm$7	4929$\pm$7	5109 $\pm$ 75
	4	6148	6434	6440	5217$\pm$0	5217$\pm$0	5946 $\pm$ 163
10	1	18922	21258	20429	17775$\pm$51	17779$\pm$61	33453 $\pm$ 629
	2	15607	17279	17067	14446$\pm$44	14473 $\pm$ 40	23804 $\pm$ 406
	3	9704	11538	10457	9191$\pm$4	9193$\pm$8	12704 $\pm$ 273
	4	5418	5980	5546	4898$\pm$0	4899$\pm$5	5166 $\pm$ 106
11	1	20935	23142	22731	19881$\pm$71	19958 $\pm$ 77	40733 $\pm$ 737

our settings, which is also consistent with the results presented in Table 7.1.

From the presented results in two tables, it is evident that the SBBA strategy consistently outperforms the other two strategies in both DCARP-OC and DCARP-CT instances, with a few exceptions in few DCARP-CT instances. This indicates that our proposed strategy is also effective in DCARP instances with real traffic conditions. This conclusion aligns with the experimental analysis conducted on artificial benchmarks as presented in Chapter 5.

Furthermore, when integrating the SBBA strategy into MAENS and comparing the optimization capabilities of MAENS-SBBA with MAENS-GRD and MASDC, the experimental results in these two tables showed that MAENS-SBBA and MAENS-GRD exhibit comparable performance and both of them outperform MASDC in both DCARP-OC and DCARP-CT instances in terms of total costs of the optimal solutions. It indicates that SBBA strategy does not notably enhance the optimization efficacy of MAENS for DCARP instances with real traffic conditions. It is mainly because the number of tasks in our DCARP instances is small, making them easier to solve. The meta-heuristic algorithm, which plays a more significant role than the solution construction strategy in solving DCARP, has been sufficient for these easier DCARP instances. Additionally, we also illustrated convergence curves of MAENS-SBBA and MAENS-GRD in a DCARP-OC and DCARP-CT scenario as shown in Figure 7.7. It is clear that the SBBA is able to obtain better solution quality under a more constrained optimization time and promotes the convergence speed of the optimization algorithm in both DCARP-OC and DCARP-CT scenarios. These conclusions are also consistent with the findings on artificial benchmarks in Chapter 5. Specifically, MAENS-SBBA presents a similar performance to MAENS-GRD on a few DCARP-OC scenarios and most DCARP-CT scenarios of *EGL-S*, which has a similar number of tasks compared with the generated DCARP instances in this experiment.

Therefore, we can conclude that the SBBA strategy demonstrates superior performance

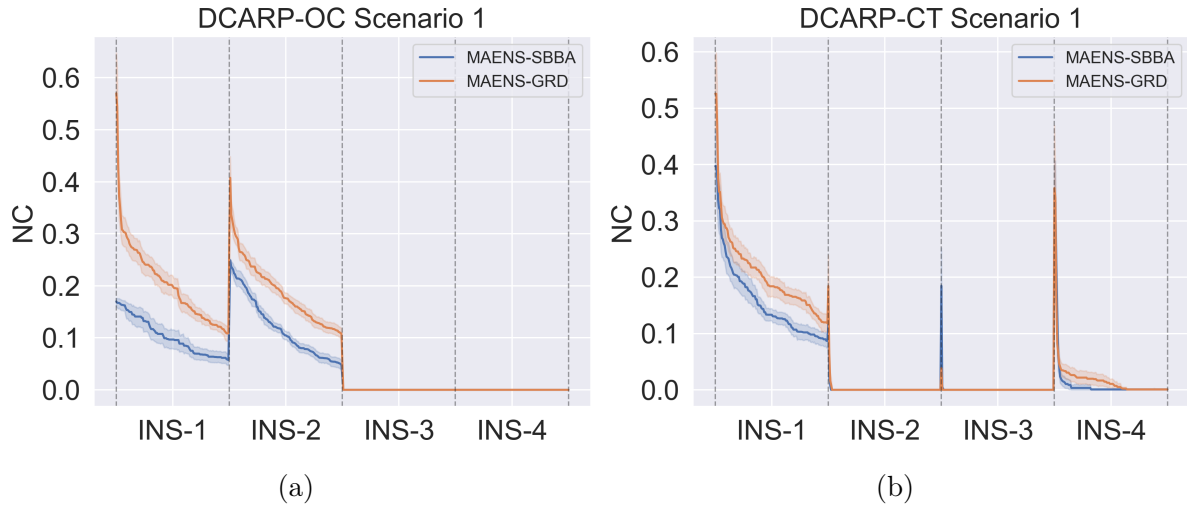


Figure 7.7: The convergence curves of MAENS-SBBA (*blue lines*) and MAENS-GRD (*orange lines*) algorithms in the first DCARP-OC and DCARP-CT scenarios. The results have been normalized according to Eq. (5.3), where TC_{min} and TC_{max} are the minimal and maximum total costs obtained during the optimization process.

for generating better initial solutions and promoting the convergence speed in both artificial benchmarks and testing scenarios with real traffic conditions. This suggests that the artificial benchmark can well evaluate the performance of the optimization algorithm and is sufficient for performance evaluation in the future. However, the magnitude and frequency of dynamic changes in real-world scenarios are much more complex and unpredictable. Consequently, the effectiveness of the entire service might deteriorate in real-world scenarios, as our algorithm lacks the ability of predicting the future events, which was demonstrated in the following subsections.

7.3.4 Further Practical Performance Analysis

In this section, we will now use the SUMO-based DCARP scenarios, which incorporate real traffic conditions, to investigate the performance of optimization algorithms from much more practical angles related to real-world applications.

7.3.4.1 Algorithmic Stability over Runs

In practice, besides the quality of obtained solutions, the stability of an algorithm is also very essential. Such stability reveals the difference of obtained solutions across different runs. If the performance of an algorithm is unstable across different runs, the algorithm is likely to yield inferior solutions with a single optimization effort. Thus, it is not suitable for a few real-world applications, such as dynamic problems which requires the quick response in practical situations. Consequently, we assessed the stability of our proposed algorithms across multiple runs in SUMO-based DCARP scenarios to determine if it is suitable for real-world applications.

In our study, each algorithm was executed with 25 independent runs for every DCARP instance and all resulting standard deviations are presented in Table 7.2 and Table 7.3. From the experimental results in these two tables, it is evident that both MAENS-SBBA and MAENS-GRD exhibit small standard deviations in relation to the total costs in both DCARP-OC and DCARP-CT scenarios. This indicates that these optimization algorithms are stable for solving DCARP instances, making them suitable for practical applications considering the algorithmic robustness. Additionally, for each DCARP scenario, the first DCARP instance which has the most number of tasks typically has the largest standard deviation compared to the remaining DCARP instances within the same DCARP scenario. This suggests that the algorithm is much more stable for DCARP instances with smaller number of tasks which are inherently easier to solve.

7.3.4.2 Changes in Vehicle Paths after Dynamic Optimization

The changes of vehicle paths are also crucial considerations in practical applications. Typically, modifications to service routes are likely to incur additional costs in real-world scenar-

ios. Besides, in practice, drivers are often assigned the service routes with which they are much more familiar, and significant alterations could deteriorate the efficacy of the service because drivers may be unfamiliar with the newly assigned service routes. Therefore, both decision makers and drivers prefer to minimize changes to the current service routes when a range of comparable solutions are available. In light of this, we investigate the changes in vehicle paths for solutions generated by our algorithm.

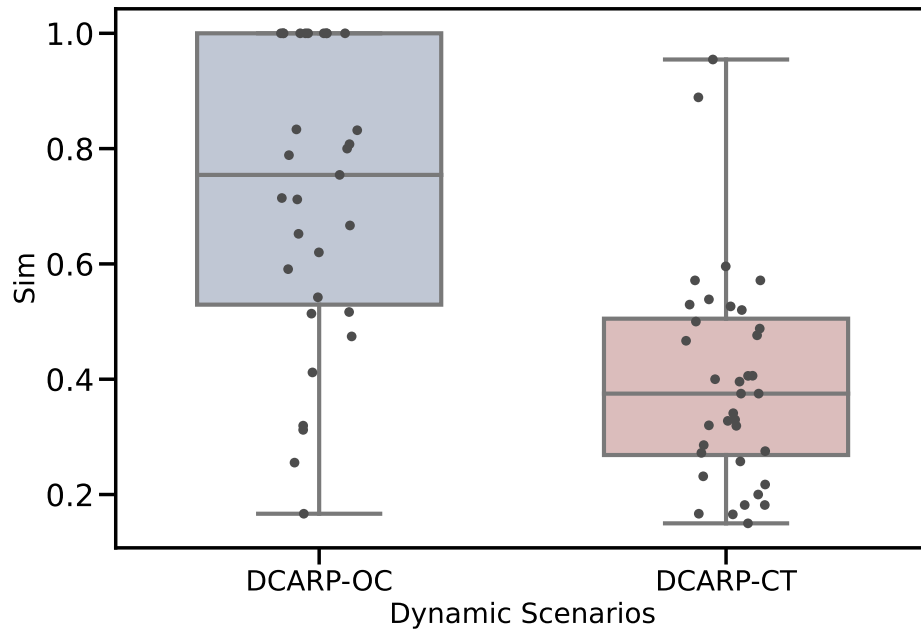


Figure 7.8: The similarity (Sim calculated by Eq. 5.5) between solutions before and after dynamic optimization for DCARP-OC and DCARP-CT instances.

The similarity between executed solutions before and after dynamic optimization is computed according to Eq. 5.5, where $\#TotalLinks$ represents the total links in the solution after dynamic optimization and $\#SameLinks$ represents the number of links inherited from the solutions before dynamic optimization. We calculated the similarity for all DCARP-OC instances and DCARP-CT instances in our experiment, with the results presented in Figure 7.8. The figure illustrates that most similarities for DCARP-OC instances are greater than 0.7, even though this criterion was never considered during the optimization process. Conversely, most similarities for DCARP-CT instances are smaller than 0.5. It is mainly be-

cause the deletion of existing tasks and insertion of new tasks cause most links in the former solution to break. Therefore, we can conclude that changes in vehicle paths for DCARP-OC scenarios are fewer than those in DCARP-CT scenarios. This observation suggests that if a practical application requires minimal shifts in vehicle paths, then new algorithms could be proposed to consider the modifications to vehicle paths during the optimization phase, particularly for DCARP-CT scenarios.

7.3.4.3 Vehicle Working Time Variations after Dynamic Optimization

Vehicle working time is also an important aspect in practical applications which considers the fairness of workloads for different vehicles. If each service route is assigned to a specific driver, significant disparities in working hours among them could lead to perceptions of unfairness. Therefore, ensuring uniformity in working hours for each vehicle is important in real-world applications. In DCARP scenarios, the dynamic optimization could potentially result in more tasks for some vehicles, thereby extending their working hours. In light of this, we explore the working time of all vehicles after using dynamic optimization in our DCARP simulation with real traffic conditions.

For each DCARP scenario, assuming each route is assigned to a distinct vehicle, we calculated the working time of each vehicle, from the moment they start at the depot until their return. The accumulated working time for all vehicles in all DCARP scenarios are presented in Figure 7.9. Analyzing the figures, it's evident that the variation of working times in all DCARP scenarios are relatively small compared to the mean working time, indicating the workloads for most vehicles are reasonably balanced. However, there are outliers in nearly all DCARP scenarios indicating that a few vehicles will spend a very long time to complete their service before returning to the depot. This is because vehicles are likely to get ensnared in the traffic jam, leading to a very long time returning to the depot, which is observed from

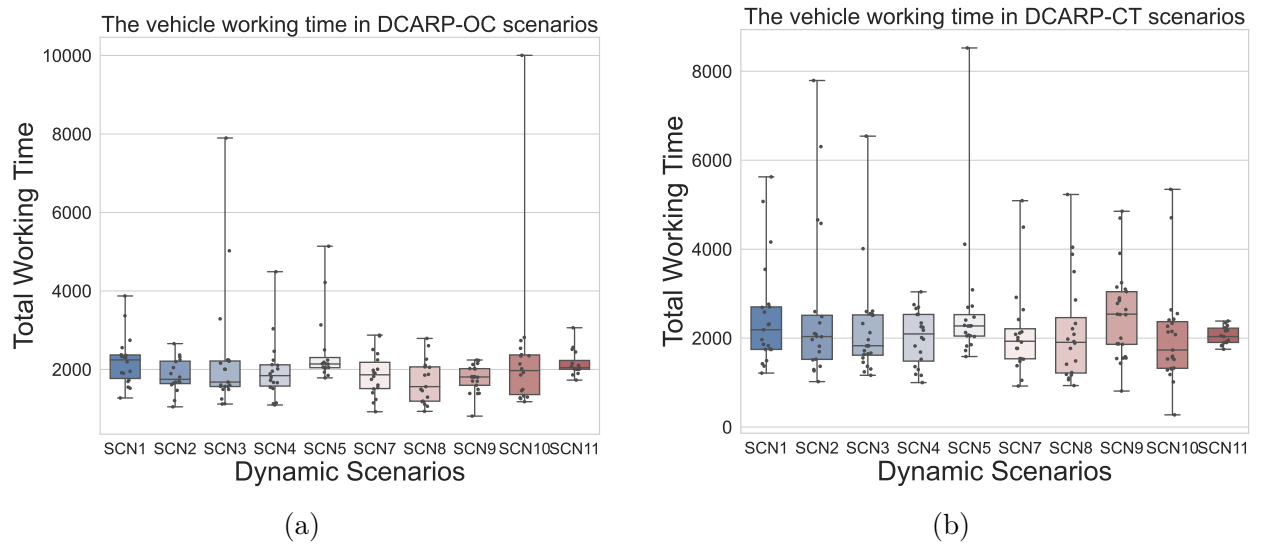


Figure 7.9: The vehicles' total working time in all DCARP scenarios after using dynamic optimization.

our simulation. Nevertheless, this is unavoidable since the new tasks might emerge either far away from the depot or during peak traffic hours, which increase the likelihood of vehicles affected by traffic jams, resulting a very long working hours. Notably, the figures also illustrated that DCARP-CT scenarios exhibit larger variation and more outliers which are also attributed to the introduction of new tasks. Consequently, we can conclude that in general our algorithm does not introduce significant unfairness regarding the working hours of various vehicles. The only cases where there were large differences in the working hours were unavoidable outliers due to traffic jam which are observed from our simulation in the SUMO. If the uniformity in working hours is necessary in real-world applications, it would be worth considering this fairness criterion during optimization.

7.4 Summary

In this chapter, we designed a dynamic CARP benchmarking framework based on a traffic simulation software, i.e., SUMO, providing a more realistic environment to evaluate DCARP

optimization algorithms. The core component in this platform is a *Controller* linking the SUMO and (D)CARP optimization algorithms. The actual traffic data is fed to the SUMO for transportation simulation. The *Controller* retrieves the traffic data from SUMO to construct a DCARP instance which is then transmitted to a dynamic optimization algorithm, and it is also responsible for updating the current service plan according to the optimization result.

Based on such benchmarking framework, we constructed different DCARP scenarios with the road network and the traffic data from the real transportation environment. In this chapter, we embedded a set of real 24-hour traffic data of Dublin city into our benchmarking framework and provided 12 different DCARP scenarios according to the different settings of the service starting time, depot location, and tasks' distribution. Therefore, the efficacy of DCARP optimization algorithm also can be evaluated in our benchmarking framework on different DCARP scenarios. We evaluated our previously proposed DCARP optimization algorithm, i.e., MAENS-SBBA, in all constructed scenarios in our experiment. The empirical results demonstrated that our algorithm is very effective on DCARP scenarios with the real traffic data for reducing the total service cost, especially on scenarios with tasks distributed in a large area, including busy and uncrowded areas. The experiments on the benchmarking framework also demonstrated the necessity of using dynamic optimization for DCARP scenarios.

Since the platform is able to simulate a complete service until all tasks are served, the benchmarking framework can also facilitate DCARP optimization algorithm comparisons and help to analyze the characteristics of different algorithms by comparing them on DCARP instances with different features. In our experiment, we compared the SBBA strategy and the MAENS-SBBA algorithm with other optimization strategies and algorithms on our platform. The observations from experiments on DCARP scenarios with real traffic conditions align with the conclusions drawn from studies on artificial benchmarks in Chapter 5. This suggests

that artificial benchmarks can well reflect the real-world benchmarks and are sufficient for evaluating the performance of optimization algorithms for DCARP in terms of the total costs of obtained solutions and the convergence speed.

In addition, based on real traffic conditions, we investigated deeper into the experimental results from three more practical perspectives. Firstly, the algorithmic robustness over different runs was studied, and the results demonstrate that our algorithm is significantly stable for solving DCARP instances. However, its stability slightly deteriorates as the problem instances become harder to solve. Secondly, the change of vehicle paths after dynamic optimization was investigated since practical applications probably prefer solutions with minimal modifications. Our studies showed that, after using our dynamic optimization algorithm, the changes in DCARP-CT scenario were more severe than those in the DCARP-OC scenarios. Finally, we focused on the variations of vehicle working time. The experimental results showed that our algorithm did not introduce significant unfairness into working time except few outliers which experience long working hours due to unavoidable traffic jam encountered when returning to the depot, and newly added unforeseen tasks, especially those coinciding with peak traffic hours, extended the working time for few vehicles, which is unavoidable in practice.

The proposed SUMO-based DCARP benchmarking system in this chapter has successfully addressed the *RQ5*: “How can we evaluate and how well do our proposed DCARP algorithm perform in real-world scenarios in addition to artificial benchmarks?”. Consequently, we have evaluated our algorithms from three distinct perspectives: comparison with baseline algorithms in artificially designed benchmarks, investigation of the approximate performance deviation relative to the lower bound in Chapter 6, and practical performance in scenarios with real-traffic conditions in this chapter. The combination of performance evaluation from these three different perspectives provides us a more comprehensive assessment of our algorithms’ performance. Even though our algorithm significantly outperformed the state-

of-the-art optimization algorithm and also performed well in practical applications under real-traffic conditions, there is still room for enhancement in most problem instances in terms of the derived lower bound.

Chapter Eight

Conclusions and Outlook

In this chapter, we first provide a comprehensive summary of the conclusions for all contributions in this thesis. Subsequently, the potential works are outlined for future research on DCARP.

8.1 Conclusions

Capacitated Arc Routing Problem (CARP) is a well-known challenging combinatorial optimization problem aiming at assigning a fleet of vehicles with limited capacities starting from a depot to serve a set of edges with demands in a graph. Due to unexpected dynamic events, such as changes in traffic conditions or service demands while vehicles are in their service, i.e., partially executing the solution, the current solution may become suboptimal or even infeasible and require an adjustment towards a better solution. When dynamic events occur, the combination of the remaining tasks, updated map information, and the status of all vehicles currently in service creates a new problem instance. Dynamic CARP (DCARP), as the main focus of this thesis, aims at optimizing this newly generated problem instance (DCARP instance) to adapt the solutions and ensure efficient service continuity. However,

the existing literature lacks work about the DCARP optimization. In light of this, in this thesis, we investigated the DCARP from multiple perspectives and proposed several effective strategies for efficient DCARP optimization.

Firstly, we provided a mathematical formulation for the studied DCARP including both the optimization objective and the constraints. In DCARP optimization, it is imperative to account for vehicles which are in service and located at different points with various remaining capacities. Due to this, existing benchmarks and algorithms for the static CARP are not applicable to DCARP optimization.

Therefore, followed by the mathematical formulation of the DCARP optimization in Chapter 2, we introduced a novel simulation system tailored for DCARP benchmarking. The proposed simulation system can generate DCARP instances through executing CARP solutions and simulating the emergence of various dynamic events, which offers an innovative research platform for subsequent DCARP studies.

Even though the existing optimization algorithms designed for static CARP are not directly applicable to DCARP, the solution representation and search operators for the static CARP can be adapted to the DCARP context. Hence, we proposed a virtual task-based novel generalized framework in Chapter 3 which can leverage all existing optimization algorithms for static CARP to address DCARP instances. This addresses *RQ1: “How can we benefit from the powerful algorithms existing for CARP in order to better solve DCARP?”*. Our virtual task strategy “virtually” treats all outside vehicles as if they were located at the depot with full capacities. It then constructs virtual tasks between stop locations and the depot to represent these outside vehicles during the optimization. We integrated several meta-heuristic algorithms into the proposed framework and evaluated them using a series of DCARP benchmarks. The experimental studies showed that newly generated algorithms perform significantly better than the state-of-the-art algorithms specifically designed

for DCARP optimization. With the support of our framework, a powerful optimization algorithm for static CARP, namely memetic algorithm with extended neighborhood search (MAENS), emerged as the state-of-the-art algorithm for DCARP optimization.

Although we have already been successful in solving DCARP instances by leveraging the wealth of algorithms from the existing CARP literature, we still lack knowledge about which dynamic factors make DCARP instances harder to solve. This gap hinders the refinement and enhancement of DCARP algorithms. Therefore, we performed the fitness landscape analysis on a range of DCARP instances to study their characteristics in Chapter 4, addressing *RQ2*: “*What makes the DCARP instance hard to solve for optimization algorithms?*”. We applied a hybrid local search algorithm to explore the landscapes of these DCARP instances with respect to the properties of both local optima and the approximate global optimum. The insights from landscape show that cost-related dynamic events have almost no impact on the characteristics of DCARP instances, while the introduction of new tasks emerges as the primary factor causing DCARP instances more challenging. As a result, we can focus more on strategies for allocating new tasks in DCARP optimization.

As a DCARP scenario comprises a series of DCARP instances with inherent similarities, the optimization experience gained from solving the former DCARP instance can potentially promote dynamic optimization for a new DCARP instance in a DCARP scenario. Therefore, we introduced the concept of “building block” as optimization experiences in DCARP scenarios in Chapter 5. Recognizing from the insights of fitness landscape analysis that cost- and task-related dynamic events have different impacts on DCARP instances, we considered two DCARP scenarios: DCARP-OC scenarios generated only with cost-changing dynamic events, and DCARP-CT scenarios generated with both cost-changing and task-changing dynamic events. A novel framework incorporating the solution building block adaptation strategy (SBBA) was proposed which effectively retains the high-quality solutions obtained in the previous optimization process, allowing for the extraction of valuable experience from

these solutions to promote the dynamic optimization process. It answers *RQ3*: “How to extract optimization experience from a DCARP instance and integrate into a meta-heuristic framework for enhanced performance in different DCARP scenarios?”.

The efficacy of the proposed SBBA strategies was first evaluated through comparing it against two conventional constructive strategies in terms of the quality of directly generated solutions. Our empirical findings demonstrated that SBBA, benefiting from the optimization experience, is able to generate significantly better solutions than the other two strategies in both DCARP-OC and DCARP-CT scenarios. Subsequent experiments, integrating MAENS with the virtual task strategy into our framework (MAENS-SBBA) to optimize DCARP scenarios, showed that MAENS-SBBA has a faster convergence rate and obtain superior solutions compared to the original algorithm, especially in DCARP-OC scenarios. For DCARP-CT scenarios, the proposed framework is much more helpful for performance improvements for smaller changes in the number of added tasks.

While our previous experiments evaluated the proposed algorithm through comparing it with other algorithms, we had not yet assessed its optimization ability on DCARP instances in relation to the theoretical global optimum. Unfortunately, as an NP-hard problem, the theoretical global optimum for most DCARP instances is unknown. Hence, we proposed a node matching lower bound approach and calculated a lower bound for DCARP instances in Chapter 6 to assist in evaluating the performance of optimization algorithms, which answers *RQ4*: “How can we derive a lower bound for evaluating the DCARP solution quality rather than relying solely on comparison with baseline algorithms?”. The lower bound was derived through solving a converted minimum cost matching problem based on the given DCARP instance. We also proposed a graph pruning strategy to increase the efficiency of calculating the lower bound. Then, armed with the obtained lower bound, we evaluated the performance of MAENS-SBBA on all generated DCARP instances. Our experimental analysis showed that the optimization results of MAENS-SBBA still possess significant potential for further

optimization enhancement on most DCARP instances compared to our lower bound.

All our previous evaluations focused on artificial benchmarks which might miss partial problem characteristics of real-world applications, especially for DCARP highly relying on the occurrence of dynamic events. Therefore, we finally designed a benchmarking system for generating DCARP scenarios based on a traffic simulation software with real-traffic data in Chapter 7. We generated 12 DCARP scenarios for algorithm performance evaluation. These scenarios were predicated on authentic map and traffic data from Dublin city, varying in depot locations, distribution of tasks, and service time periods. It answers *RQ5: “How can we evaluate and how well do our proposed DCARP algorithm perform in real-world scenarios in addition to artificial benchmarks?”*. The optimization algorithm MAENS-SBBA was evaluated in these DCARP scenarios, and the experimental results demonstrated the proposed algorithm is still effective in DCARP scenarios with real-traffic conditions.

Additionally, we compared strategies and algorithms that had previously been tested on artificial benchmarks, on SUMO-based DCARP scenarios. Our empirical studies drew the same conclusion as that from artificial benchmarks, which reflects that artificial benchmarks are sufficient for evaluating optimization algorithms in terms of total costs of finally obtained solution and the convergence performance. Moreover, leveraging these real-traffic conditions, we further investigated the performance of algorithms from practical perspectives including the robustness over runs, changes in vehicle paths after dynamic optimization, and the variance in vehicle working time with dynamic optimization. The analysis suggests that our algorithm is robust even under real-traffic conditions and will not introduce serious unbalanced workload among vehicles. However, changes in vehicle paths are not guaranteed, especially in DCARP-CT scenarios according to our empirical studies.

8.2 Future Work

Even though the contributions of this thesis have covered several perspectives in the DCARP context, there still exist a few potential improvements in the future due to the limitations of the proposed algorithms and the scope of the experimental analysis.

First of all, we proposed SBBA strategies to extract optimization experience in DCARP scenarios, which directly adapt historical solutions to new DCARP instances to promote the dynamic optimization. However, its performance is still similar to or even worse than the greedy generation strategy in a few DCARP-CT instances. In the future, more effective strategies could be proposed for DCARP-CT instances. Since SBBA directly inherits task sequences from historical solutions, it would be valuable to extract more indirect experience from historical solutions to help generate higher-quality solutions for DCARP scenarios. With the recent advancements and proven efficacy of deep learning techniques in addressing a few combinatorial optimization problems [10], it would also be worthwhile to integrate potential deep learning methods to further enhance the dynamic optimization for DCARP. Additionally, these strategies are also suitable for similar combinatorial optimization problems, such as the CVRP and TSP. Therefore, it would also be valuable to evaluate our algorithms on these similar problems in the future.

Secondly, insights from the fitness landscape analysis suggest to pay more attention to new tasks when optimizing DCARP instances. Even though we have proposed a specific insertion heuristic in SBBA strategies demonstrated to be useful in DCARP-CT scenarios, its reliance solely on greedy insertion based on the building block limits its performance currently. Therefore, it would be valuable to consider the features of new tasks in more detail and develop more effective strategies to adeptly handle new tasks in DCARP scenarios.

The DCARP scenarios studied in this thesis have also been of a small to medium scale,

encompassing no more than 200 tasks. In real-world applications, the number of tasks can often exceed 200 tasks, even greater than 1000 tasks. This requires specific strategies to navigate the vast search space. Therefore, in the future, it is worthwhile to investigate more about large-scale DCARP scenarios and propose experience-assisted strategies to enhance the dynamic optimization for these scenarios.

Further, while the tightness of the proposed node matching lower bound has been validated to be very close to the theoretical global optimum for DCARP instances with a small number of tasks, its tightness on large complex instances still remains uncertain. Therefore, it is valuable to perform a deeper exploration into how specific characteristics of instances influence the NMLB on the NMLB and develop tighter lower bounds for DCARP instances.

Finally, the current SUMO-based benchmarking system is only designed for specific DCARP scenarios with predefined settings of depots and tasks, and it is not convenient for broader research utilization with other kinds of scenarios. Hence, we are going to develop a set of user-friendly interfaces for this benchmarking framework so that more researchers can use it to evaluate their algorithms. Furthermore, it is also valuable to construct more DCARP scenarios including small, medium, and large numbers of cities for analyzing the effectiveness of each optimization algorithm in different DCARP scenarios.

The optimization experience presented in this thesis represents our initial step for learning to optimize. We only concentrated on extracting optimization experience from a single source problem instance to promote the optimization of a new, significantly related target problem instance. In the future, it would be worthwhile to explore more about learning experiences from a variety of simpler source problem instances to assist in solving more challenging problem instances. Furthermore, while we currently only consider the optimization experience within the same problem domain, the potential for transferring optimization knowledge across different problem domains also presents a valuable direction in the future.

Appendix One

Supplemental Materials

A.1 Full Results for Section 3.3.4: Analysis of the Effects of the Virtual-Task Strategy (Table 3.3)

Table A.1: Results of VT-RTS, VTtr-RTS, VT-ILMA, VTtr-ILMA, VT-MAENS, VTtr-MAENS, VT-MASDC and VTtr-MASDC on the *egl* dataset (E1-A $\sim$ E2-C). The values in each cell represent the “MEAN $\pm$ STD” with the average ranking (in the brackets) w.r.t the average total cost over 25 independent runs. The bold values are the better results under the Wilcoxon signed-rank test with the level of significance 0.05 between restart and inheriting strategies in an algorithm for a DCARP instance.

Map	Ins	VT-RTS	VTtr-RTS	VT-ILMA	VTtr-ILMA	VT-MAENS	VTtr-MAENS	VT-MASDC	VTtr-MASDC
E1-A	1	10465 $\pm$ 35(5.9)	9608$\pm$42(5.1)	8882$\pm$88(3.3)	8886$\pm$96(3.1)	8738$\pm$54(1.8)	8744$\pm$60(1.8)	17398$\pm$1359(7.5)	17398$\pm$1359(7.5)
	2	10801 $\pm$ 0(6)	9307$\pm$93(3.6)	9369$\pm$110(4.0)	9391$\pm$98(4.3)	9167$\pm$41(1.6)	9156$\pm$38(1.5)	18667$\pm$1820(7.5)	18667$\pm$1820(7.5)
	3	11518 $\pm$ 46(5.7)	10491$\pm$169(4.9)	10309$\pm$121(3.7)	10292$\pm$107(3.7)	9920$\pm$79(1.4)	9917$\pm$95(1.6)	19505$\pm$2078(7.5)	19505$\pm$2078(7.5)
	4	16543 $\pm$ 0(6)	12685$\pm$143(3.5)	12911$\pm$236(4.1)	12933$\pm$271(4.4)	12217$\pm$63(1.4)	12221$\pm$75(1.6)	29176$\pm$2612(7.5)	29176$\pm$2612(7.5)
	5	15214 $\pm$ 91(6)	12856$\pm$162(4.4)	12782$\pm$198(4.0)	12731$\pm$204(3.6)	12055$\pm$89(1.5)	12056$\pm$77(1.5)	29298$\pm$2879(7.5)	29298$\pm$2879(7.5)
E1-B	1	11847$\pm$719(4.9)	14106 $\pm$ 186(6)	10123 $\pm$ 176(3.8)	10050 $\pm$ 137(3.3)	9589 $\pm$ 34(1.5)	9582 $\pm$ 44(1.5)	19356 $\pm$ 2043(7.5)	19356 $\pm$ 2043(7.5)
	2	12769$\pm$10(5.5)	13398$\pm$1779(5.5)	10627$\pm$181(3.6)	10578$\pm$156(3.4)	10063$\pm$28(1.5)	10061$\pm$25(1.5)	19599$\pm$1581(7.5)	19599$\pm$1581(7.5)
	3	1824 $\pm$ 25(6)	16828$\pm$607(5)	14229$\pm$212(3.6)	14179$\pm$235(3.4)	13659$\pm$50(1.4)	13667$\pm$46(1.6)	26484$\pm$2919(7.5)	26484$\pm$2919(7.5)
	4	16846$\pm$49(5.3)	17849$\pm$2528(5.4)	13928$\pm$214(3.6)	13949$\pm$193(3.8)	13334$\pm$26(1.7)	13321$\pm$23(1.3)	24979$\pm$1879(7.5)	24979$\pm$1879(7.5)
	5	17444$\pm$79(5.4)	17231$\pm$905(5.6)	14055$\pm$153(3.5)	14022$\pm$131(3.5)	13526$\pm$56(1.5)	13533$\pm$56(1.5)	27082$\pm$2155(7.5)	27082$\pm$2155(7.5)
E1-C	1	12899$\pm$652(5.2)	13220 $\pm$ 682(5.7)	9985$\pm$171(3.6)	9953$\pm$152(3.6)	9604$\pm$46(1.5)	9564$\pm$36(1.5)	16584$\pm$1760(7.5)	16584$\pm$1760(7.5)
	2	17580$\pm$21(5.0)	19788 $\pm$ 1143(6.1)	14403$\pm$85(3.6)	14431$\pm$164(3.5)	13908$\pm$63(1.6)	13880$\pm$55(1.4)	22521$\pm$1437(7.4)	22521$\pm$1437(7.4)
	3	18672$\pm$88(5)	20128 $\pm$ 138(6)	13551$\pm$200(3.4)	15844$\pm$124(3.6)	15147$\pm$56(1.5)	15143$\pm$68(1.5)	26301$\pm$2000(7.5)	26301$\pm$2000(7.5)
	4	18317$\pm$477(5.3)	18630 $\pm$ 197(5.7)	14414$\pm$217(3.6)	14394$\pm$221(3.4)	13735$\pm$88(1.5)	13748$\pm$101(1.5)	27743$\pm$2191(7.5)	27743$\pm$2191(7.5)
	5	20998$\pm$138(5)	22605 $\pm$ 331(6)	15932$\pm$189(3.6)	15894$\pm$230(3.4)	15333$\pm$9(1.4)	15336$\pm$11(1.6)	31064$\pm$2866(7.5)	31064$\pm$2866(7.5)
E2-A	1	12049 $\pm$ 72(6)	10679$\pm$258(4.9)	10121$\pm$127(3.6)	10070$\pm$114(3.4)	9714$\pm$60(1.5)	9720$\pm$55(1.5)	18699$\pm$1742(7.5)	18699$\pm$1742(7.5)
	2	14173 $\pm$ 246(5.7)	12743$\pm$1499(4.9)	11042$\pm$117(3.4)	11071$\pm$128(4.0)	10712$\pm$76(1.4)	10719$\pm$81(1.6)	21735$\pm$1707(7.5)	21735$\pm$1707(7.5)
	3	14358 $\pm$ 46(6.0)	12190$\pm$1489(5.0)	10688$\pm$186(3.6)	10654$\pm$150(3.4)	10287$\pm$23(1.4)	10294$\pm$30(1.6)	23440$\pm$2822(7.5)	23440$\pm$2822(7.5)
	4	10803 $\pm$ 793(5.3)	9898$\pm$122(3.2)	10132$\pm$131(4.7)	10165$\pm$150(4.8)	9471$\pm$36(1.4)	9482$\pm$63(1.6)	20649$\pm$1990(7.5)	20649$\pm$1990(7.5)
	5	11814$\pm$443(4.4)	13003 $\pm$ 1148(5.9)	11623$\pm$166(3.8)	11621$\pm$101(3.9)	11127$\pm$86(1.4)	11129$\pm$74(1.6)	23731$\pm$2454(7.5)	23731$\pm$2454(7.5)
E2-B	1	14822$\pm$24(5.3)	15626 $\pm$ 1340(5.7)	12624$\pm$157(3.6)	12617$\pm$150(3.4)	12082$\pm$76(1.5)	12092$\pm$66(1.5)	22829$\pm$1803(7.5)	22829$\pm$1803(7.5)
	2	24926$\pm$39(5)	31035 $\pm$ 138(6.2)	21828$\pm$208(3.6)	21860$\pm$218(3.4)	20966$\pm$27(1.5)	20973$\pm$32(1.5)	34624$\pm$2271(7.4)	34624$\pm$2271(7.4)
	3	22818$\pm$236(5.5)	23448$\pm$4504(5.4)	18715$\pm$212(3.6)	18681$\pm$179(3.5)	18055$\pm$80(1.6)	18064$\pm$75(1.4)	32244$\pm$2307(7.5)	32244$\pm$2307(7.5)
	4	20693 $\pm$ 23(6.0)	18843$\pm$1502(4.2)	15533$\pm$282(3.9)	18491$\pm$199(3.9)	17754$\pm$9(1.6)	17744$\pm$17(1.4)	31178$\pm$1906(7.5)	31178$\pm$1906(7.5)
	5	23918$\pm$206(5)	27676 $\pm$ 276(6)	21215$\pm$173(3.5)	21272$\pm$286(3.5)	20502$\pm$76(1.7)	20477$\pm$82(1.3)	35184$\pm$2828(7.5)	35184$\pm$2828(7.5)
E2-C	1	19158$\pm$39(5.1)	19760 $\pm$ 339(6.1)	15655$\pm$128(3.3)	15556$\pm$185(3.7)	15103$\pm$66(1.6)	15164$\pm$57(1.4)	24052$\pm$2346(7.4)	24052$\pm$2346(7.4)
	2	19099$\pm$14(5)	22019 $\pm$ 140(6.1)	16496$\pm$112(3.6)	16466$\pm$165(3.4)	15855$\pm$48(1.7)	15831$\pm$41(1.3)	25457$\pm$1845(7.5)	25457$\pm$1845(7.5)
	3	21042 $\pm$ 10(6)	20168$\pm$16(5)	17479$\pm$155(3.4)	17521$\pm$199(3.6)	17725$\pm$32(1.6)	17213$\pm$35(1.4)	27856$\pm$2381(7.5)	27856$\pm$2381(7.5)
	4	27475 $\pm$ 278(6)	24900$\pm$0(5)	21322$\pm$171(3.4)	21366$\pm$183(3.6)	20714$\pm$80(1.5)	20707$\pm$66(1.5)	32173$\pm$2287(7.5)	32173$\pm$2287(7.5)
	5	29733 $\pm$ 15(6)	27727$\pm$521(5)	24182$\pm$288(3.6)	24168$\pm$258(3.4)	23416$\pm$68(1.5)	23398$\pm$61(1.5)	36975$\pm$2467(7.5)	36975$\pm$2467(7.5)

A.1. Full Results for Section 3.3.4: Analysis of the Effects of the Virtual-Task Strategy
(Table 3.3)

Table A.2: Results of VT-RTS, VTtr-RTS, VTtr-ILMA, VTtr-MAENS, VTtr-MAENS, VT-MASDC and VTtr-MASDC on the *egl* dataset (E3-A $\sim$ E4-C). The values in each cell represent the “MEAN $\pm$ STD” with the average ranking (in the brackets) w.r.t the average total cost over 25 independent runs. The bold values are the better results under the Wilcoxon signed-rank test with the level of significance 0.05 between restart and inheriting strategies in an algorithm for a DCARP instance.

Map	Ins	VT-RTS	VTtr-RTS	VT-ILMA	VTtr-ILMA	VT-MAENS	VTtr-MAENS	VT-MASDC	VTtr-MASDC
E3-A	1	11591 $\pm$ 589(4.6)	11178$\pm$122(3.1)	11542$\pm$259(4.8)	11583$\pm$180(5.5)	10890$\pm$46(1.6)	10875$\pm$35(1.4)	22269$\pm$2386(7.5)	22225$\pm$2415(7.5)
	2	11182$\pm$290(5)	12626 $\pm$ 126(6)	10327$\pm$142(3.3)	10403 $\pm$ 130(3.7)	9903$\pm$34(1.4)	9908$\pm$43(1.6)	17680$\pm$1885(7.5)	17680$\pm$1885(7.5)
	3	13917$\pm$183(5.5)	14362$\pm$1361(5.4)	13138$\pm$210(3.4)	13190$\pm$193(3.7)	12594$\pm$53(1.6)	12578$\pm$45(1.4)	27775$\pm$2164(7.5)	27775$\pm$2164(7.5)
	4	17575 $\pm$ 195(6)	15054$\pm$894(4.1)	14612$\pm$188(4.0)	14609$\pm$198(3.9)	14241$\pm$91(1.5)	14238$\pm$14(1.5)	25524$\pm$2472(7.5)	25524$\pm$2472(7.5)
	5	14708$\pm$745(3.8)	14543$\pm$74(4.1)	14647$\pm$146(4.8)	14675$\pm$160(5.2)	14168$\pm$34(1.4)	14177$\pm$42(1.6)	25984$\pm$2326(7.5)	25984$\pm$2326(7.5)
E3-B	1	17316 $\pm$ 242(5.8)	15256$\pm$1514(4.1)	14546$\pm$211(3.8)	14557$\pm$137(4.0)	14237$\pm$125(1.9)	14181$\pm$119(1.4)	26497$\pm$2717(7.5)	26497$\pm$2717(7.5)
	2	18654$\pm$353(5.5)	18582$\pm$5(5.5)	15769$\pm$142(3.4)	15759$\pm$146(3.6)	15274$\pm$57(1.4)	15293$\pm$64(1.6)	26680$\pm$2580(7.5)	26680$\pm$2580(7.5)
	3	20429 $\pm$ 16(6)	19390$\pm$34(5)	15968$\pm$113(3.4)	16000$\pm$196(3.6)	15436$\pm$53(1.4)	15468$\pm$83(1.6)	28072$\pm$2627(7.5)	28072$\pm$2627(7.5)
	4	19481 $\pm$ 78(5.6)	18636$\pm$1287(4.4)	17747$\pm$203(3.8)	17807$\pm$187(4.2)	17194$\pm$39(1.4)	17190$\pm$19(1.6)	28606$\pm$1573(7.5)	28606$\pm$1573(7.5)
	5	20588$\pm$562(4.9)	21544 $\pm$ 95(6)	17993$\pm$166(3.5)	17959$\pm$160(3.6)	17293$\pm$52(1.4)	17318$\pm$52(1.6)	30710$\pm$1867(7.5)	30710$\pm$1867(7.5)
E3-C	1	19711$\pm$5(5)	22269 $\pm$ 254(6.1)	17980$\pm$167(3.6)	17985$\pm$187(3.4)	17274$\pm$133(1.5)	17253$\pm$113(1.3)	26340$\pm$1730(7.5)	26340$\pm$1730(7.5)
	2	18533$\pm$0(5)	22944 $\pm$ 23(6.1)	17202$\pm$313(3.6)	17067$\pm$228(3.4)	16342$\pm$65(1.5)	16336$\pm$51(1.5)	25883$\pm$1823(7.5)	25883$\pm$1823(7.5)
	3	21867$\pm$300(5)	23726 $\pm$ 249(6)	17871$\pm$176(3.5)	17913$\pm$269(3.5)	17214$\pm$49(1.5)	17211$\pm$63(1.5)	27940$\pm$2635(7.5)	27940$\pm$2635(7.5)
	4	24038$\pm$0(5)	26515 $\pm$ 787(6.1)	20688$\pm$209(3.4)	20812$\pm$231(3.6)	19695$\pm$73(1.2)	19738 $\pm$ 79(1.8)	30885$\pm$2344(7.5)	30885$\pm$2344(7.5)
	5	27518$\pm$29(5.2)	28211 $\pm$ 473(5.8)	23431$\pm$207(3.5)	23483$\pm$284(3.5)	22522$\pm$64(1.4)	22540$\pm$57(1.6)	34525$\pm$2083(7.5)	34525$\pm$2083(7.5)
E4-A	1	11718 $\pm$ 68(6)	10185$\pm$330(3.3)	10359$\pm$161(4.4)	10296$\pm$148(4.3)	9816$\pm$41(1.6)	9816$\pm$54(1.4)	23180$\pm$2486(7.5)	23180$\pm$2486(7.5)
	2	11269$\pm$98(3.1)	12037 $\pm$ 1273(5.4)	11539$\pm$219(4.7)	11566$\pm$146(4.8)	10994$\pm$17(1.6)	10990$\pm$11(1.4)	23378$\pm$2210(7.5)	23378$\pm$2210(7.5)
	3	10492$\pm$120(3.3)	12868 $\pm$ 1544(5.5)	10701$\pm$186(4.5)	10741$\pm$196(4.7)	10142$\pm$39(1.5)	10142$\pm$29(1.5)	21125$\pm$2110(7.5)	21125$\pm$2110(7.5)
	4	15092 $\pm$ 452(5.7)	14079$\pm$89(3.8)	14148$\pm$170(4.2)	14213$\pm$164(4.4)	13500$\pm$30(1.4)	13507$\pm$30(1.6)	26856$\pm$2673(7.5)	26856$\pm$2673(7.5)
	5	16224 $\pm$ 137(6)	15212$\pm$222(4.7)	14837$\pm$241(3.6)	14816$\pm$283(3.7)	13788$\pm$79(1.6)	13759$\pm$59(1.4)	28261$\pm$2526(7.5)	28261$\pm$2526(7.5)
E4-B	1	18400 $\pm$ 139(6)	15737$\pm$1004(4.8)	14654$\pm$193(3.6)	14653$\pm$206(3.6)	14007$\pm$57(1.6)	13983$\pm$46(1.4)	27267$\pm$2660(7.5)	27267$\pm$2660(7.5)
	2	16020$\pm$153(5.0)	17668 $\pm$ 900(5.9)	13886$\pm$183(3.4)	13876$\pm$174(3.6)	13202$\pm$113(1.6)	13157$\pm$84(1.4)	22915$\pm$1820(7.5)	22915$\pm$1820(7.5)
	3	20373 $\pm$ 22(6)	20342$\pm$76(5)	17946$\pm$166(3.3)	18039$\pm$216(3.7)	17347$\pm$101(1.6)	17334$\pm$89(1.4)	27483$\pm$1932(7.5)	27483$\pm$1932(7.5)
	4	22397 $\pm$ 0(5.9)	20523$\pm$1306(5)	19505$\pm$282(3.3)	19628$\pm$295(3.8)	18674$\pm$66(1.5)	18680$\pm$45(1.5)	31500$\pm$2281(7.5)	31500$\pm$2281(7.5)
	5	17591$\pm$125(3.5)	17696 $\pm$ 217(4.0)	17920$\pm$181(5.4)	17943$\pm$222(5.1)	17351$\pm$34(1.5)	17355$\pm$42(1.5)	27969$\pm$2555(7.5)	27969$\pm$2555(7.5)
E4-C	1	22459$\pm$122(5)	23201 $\pm$ 32(6)	18870$\pm$215(3.6)	18797$\pm$195(3.4)	18235$\pm$25(1.5)	18237$\pm$23(1.5)	28150$\pm$2120(7.5)	28150$\pm$2120(7.5)
	2	21901$\pm$29(5)	22251 $\pm$ 31(6)	18343$\pm$132(3.4)	18393$\pm$191(3.6)	17650$\pm$39(1.6)	17640$\pm$60(1.4)	28010$\pm$1870(7.5)	28010$\pm$1870(7.5)
	3	23802 $\pm$ 71(6)	22224$\pm$12(5)	19962$\pm$154(3.6)	19939$\pm$215(3.4)	19332$\pm$38(1.5)	19321$\pm$55(1.5)	28525$\pm$1982(7.5)	28525$\pm$1982(7.5)
	4	26380 $\pm$ 0(6)	24666$\pm$97(5)	20881$\pm$220(3.4)	20963$\pm$193(3.6)	20113$\pm$21(1.5)	20112$\pm$31(1.5)	32540$\pm$1932(7.5)	32540$\pm$1932(7.5)
	5	27446$\pm$16(5)	27703 $\pm$ 137(6)	23023$\pm$162(3.5)	22974$\pm$134(3.5)	22229$\pm$54(1.7)	22207$\pm$50(1.3)	32859$\pm$2143(7.5)	32859$\pm$2143(7.5)

Table A.3: Results of VT-RTS, VTtr-RTS, VT-ILMA, VTtr-ILMA, VT-MAENS, VTtr-MAENS, VT-MASDC and VTtr-MASDC on the *egl* dataset (S1-A ~ S2-C). The values in each cell represent the “MEAN±STD” with the average ranking (in the brackets) w.r.t the average total cost over 25 independent runs. The bold values are the better results under the Wilcoxon signed-rank test with the level of significance 0.05 between restart and inheriting strategies in an algorithm for a DCARP instance.

Map	Ins	VT-RTS	VTtr-RTS	VT-ILMA	VTtr-ILMA	VT-MAENS	VTtr-MAENS	VT-MASDC	VTtr-MASDC
S1-A	1	16244±783(3.7)	16196±171(4.2)	16484±279(5.1)	16395±265(5)	15307±52(1.4)	15324±47(1.6)	38745±3199(7.5)	38745±3199(7.5)
	2	22746±222(6)	18195±240(3.4)	18493±293(4.5)	18397±272(4.1)	17333±34(1.4)	17350±49(1.6)	43534±4294(7.5)	43534±4294(7.5)
	3	25304±1395(5.5)	23389±441(4.6)	23068±347(3.7)	23114±323(4.2)	21626±61(1.4)	21626±80(1.6)	55374±3631(7.5)	55374±3631(7.5)
	4	22606±252(4.8)	22277±200(3.4)	22573±195(4.6)	22614±239(5.2)	21731±78(1.5)	21716±103(1.5)	56181±4121(7.5)	56181±4121(7.5)
	5	33013±482(6)	26681±325(3.5)	26699±272(4.2)	26931±355(4.3)	25187±126(1.4)	25222±152(1.6)	67488±4733(7.5)	67488±4733(7.5)
S1-B	1	20982±33(5.6)	18362±224(2.4.5)	16928±200(4.0)	16842±199(3.8)	16066±52(1.5)	16052±48(1.5)	36477±2556(7.5)	36477±2556(7.5)
	2	35778±115(5)	38863±1044(6)	30554±348(3.5)	30488±309(3.5)	28701±68(1.4)	28728±78(1.6)	59367±3954(7.5)	59367±3954(7.5)
	3	40836±418(5)	44385±1323(6)	34815±588(3.5)	34814±356(3.5)	32680±81(1.3)	32730±163(1.7)	67750±4157(7.5)	67750±4157(7.5)
	4	40276±44(5.1)	46654±3691(5.9)	34164±479(3.6)	34135±365(3.4)	31611±168(1.7)	31543±115(1.3)	70649±4957(7.5)	70649±4957(7.5)
	5	45608±497(5.3)	43182±5238(5.2)	36248±402(3.7)	36172±435(3.8)	34368±142(1.6)	34322±89(1.4)	76455±4848(7.5)	76455±4848(7.5)
S1-C	1	23876±72(5)	24763±162(6)	19849±198(3.5)	19872±244(3.5)	19048±43(1.3)	19081±71(1.7)	34630±2286(7.5)	34630±2286(7.5)
	2	33366±43(5.6)	32908±1605(5.2)	27801±315(3.6)	27814±323(3.6)	26644±84(1.5)	26670±69(1.5)	49935±3144(7.5)	49935±3144(7.5)
	3	33506±34(5.8)	33463±85(5.2)	28673±244(3.5)	28704±239(3.5)	27513±118(1.5)	27534±85(1.5)	50061±2618(7.5)	50061±2618(7.5)
	4	51535±0(5)	54117±66(6)	44494±389(3.4)	44545±474(3.6)	42632±113(1.6)	42633±79(1.4)	73369±4441(7.5)	73369±4441(7.5)
	5	44011±24(5.4)	43099±551(4.9)	36751±377(4.0)	36721±420(3.7)	34788±102(1.5)	34761±124(1.5)	66642±3622(7.5)	66642±3622(7.5)
S2-A	1	24926±88(6)	19408±1013(3.0)	20440±309(4.4)	20373±285(4.4)	18868±52(1.7)	18868±49(1.5)	46849±4765(7.5)	46849±4765(7.5)
	2	26386±1958(4.5)	24743±298(3.1)	26090±375(5.0)	26243±341(5.2)	24247±92(1.6)	24218±67(1.5)	56622±3659(7.5)	56622±3659(7.5)
	3	29149±30(6)	24409±276(3)	25504±341(4.5)	25505±379(4.5)	24027±54(1.5)	24027±38(1.5)	54507±3833(7.5)	54507±3833(7.5)
	4	34263±3(6)	27398±379(3)	28384±255(4.5)	28325±380(4.5)	26336±97(1.4)	26348±100(1.6)	67998±4904(7.5)	67998±4904(7.5)
	5	23457±304(3.4)	23583±188(3.7)	24110±274(5.3)	24084±247(5.6)	22709±97(1.5)	22721±94(1.5)	54086±4122(7.5)	54086±4122(7.5)
S2-B	1	32049±1118(5.9)	27489±374(3.0)	28378±358(4.5)	28380±360(4.5)	26791±129(1.5)	26804±135(1.5)	56566±4401(7.5)	56566±4401(7.5)
	2	38856±3442(3.5)	41869±3802(5)	38737±365(4.7)	38807±463(4.8)	36461±121(1.5)	36452±96(1.5)	72236±4367(7.5)	72236±4367(7.5)
	3	42136±390(6)	35292±1778(3.4)	35498±366(4.2)	35577±405(4.4)	33781±125(1.4)	33823±131(1.6)	63547±3393(7.5)	63547±3393(7.5)
	4	38481±102(5)	39486±149(6)	34097±378(3.5)	34169±335(3.5)	32500±133(1.4)	32262±122(1.6)	70548±5522(7.5)	70548±5522(7.5)
	5	37322±169(6)	35672±51(5)	31447±294(3.5)	31486±320(3.5)	29879±144(1.4)	29931±178(1.6)	60579±4462(7.5)	60579±4462(7.5)
S2-C	1	36905±13(6)	33986±3028(4.4)	29721±296(3.7)	29744±304(3.8)	28199±118(1.6)	28141±91(1.4)	53621±2233(7.5)	53621±2233(7.5)
	2	38863±25(6)	37245±49(5)	32644±225(3.7)	32782±255(3.3)	31522±95(1.4)	31566±111(1.6)	53613±2521(7.5)	53613±2521(7.5)
	3	50854±229(5.1)	53235±2935(5.8)	44089±401(3.6)	44092±383(3.6)	44330±181(1.4)	44289±156(1.6)	77091±5542(7.5)	77091±5542(7.5)
	4	57254±71(6)	52573±35(5)	45573±364(3.3)	45883±360(3.7)	44193±102(1.4)	44221±108(1.6)	78275±3948(7.5)	78275±3948(7.5)
	5	50864±76(6)	47012±8(5)	41192±395(3.4)	41321±313(3.6)	39343±167(1.5)	39295±141(1.5)	68306±3296(7.5)	68306±3296(7.5)

A.1. Full Results for Section 3.3.4: Analysis of the Effects of the Virtual-Task Strategy (Table 3.3)

Table A.4: Results of VT-RTS, VTtr-RTS, VTtr-ILMA, VTtr-MAENS, VTtr-MAENS, VT-MASDC and VTtr-MASDC on the *egl* dataset (S3-A ~ S4-C). The values in each cell represent the “MEAN±STD” with the average ranking (in the brackets) w.r.t the average total cost over 25 independent runs. The bold values are the better results under the Wilcoxon signed-rank test with the level of significance 0.05 between restart and inheriting strategies in an algorithm for a DCARP instance.

Map	Ins	VT-RTS	VTtr-RTS	VT-ILMA	VTtr-ILMA	VT-MAENS	VTtr-MAENS	VT-MASDC	VTtr-MASDC
S3-A	1	22989±156(3.6)	22971±190(3.4)	24100±307(5.5)	24073±289(5.5)	22411±61(1.5)	22401±70(1.5)	52327±3849(7.5)	52327±3849(7.5)
	2	25249±348(2.5)	26012±192(4.0)	27134±338(5.3)	27234±405(5.6)	25013±105(1.8)	25017±103(1.8)	57870±3931(7.5)	57870±3931(7.5)
	3	25907±350(3.3)	26117±160(3.7)	27011±250(5.5)	27042±320(5.5)	25285±145(1.4)	25349±113(1.6)	59652±3792(7.5)	59652±3792(7.5)
	4	41928±177(6)	33347±287(3)	34800±398(4.5)	34840±420(4.5)	32374±119(1.4)	32423±100(1.6)	72252±4788(7.5)	72250±4790(7.5)
	5	31319±4(6)	24329±316(2.7)	25614±322(4.6)	25632±334(4.4)	23989±74(1.7)	24006±85(1.6)	62362±4014(7.5)	62362±4014(7.5)
S3-B	1	30945±70(6)	29429±141(5)	26486±282(3.4)	26493±230(3.6)	25161±123(1.5)	25165±106(1.5)	46491±3247(7.5)	46491±3247(7.5)
	2	28494±1307(3.5)	29198±1867(4.1)	29318±356(5.2)	29374±347(4.7)	27445±121(1.8)	27356±140(1.2)	55642±2518(7.5)	55642±2518(7.5)
	3	36665±2093(5.3)	35529±3448(3.8)	34378±330(4.2)	34515±347(4.7)	32584±78(1.5)	32613±112(1.5)	65559±3265(7.5)	65559±3265(7.5)
	4	36424±10(6)	29498±240(3.0)	30262±280(4.6)	30306±339(4.4)	28983±133(1.6)	28930±131(1.4)	59740±4565(7.5)	59740±4565(7.5)
	5	32715±2377(4.0)	31524±348(3.7)	32271±345(5.2)	32185±177(5.1)	30444±167(1.4)	30437±160(1.6)	61165±3506(7.5)	61165±3506(7.5)
S3-C	1	45906±0(5.0)	47919±1147(6.0)	40932±333(3.4)	40930±425(3.6)	39036±146(1.4)	39049±154(1.6)	69270±3741(7.5)	69270±3741(7.5)
	2	58082±263(5.0)	58770±195(6.0)	51538±346(3.6)	51445±366(3.4)	49131±169(1.5)	49157±207(1.5)	78938±4233(7.5)	78938±4233(7.5)
	3	45802±74(5.2)	45868±21(5.8)	38727±341(3.4)	38925±241(3.6)	36791±118(1.6)	36730±94(1.4)	67602±3362(7.5)	67602±3362(7.5)
	4	41011±1(6)	40879±115(5)	35457±319(3.5)	35339±338(3.5)	33663±89(1.7)	33653±91(1.3)	61607±2949(7.5)	61607±2949(7.5)
	5	49309±114(5.5)	46793±3817(4.8)	42843±488(3.9)	42764±568(3.7)	40153±107(1.6)	40143±132(1.4)	72621±3858(7.5)	72621±3858(7.5)
S4-A	1	24849±1470(6)	21105±170(2.9)	22336±255(4.5)	22385±332(4.5)	20730±131(1.6)	20699±85(1.5)	53513±3603(7.5)	53513±3603(7.5)
	2	24693±1073(3.1)	24947±322(4.0)	26177±477(5.4)	26175±352(5.5)	23967±134(1.6)	23920±96(1.4)	59023±3019(7.5)	59023±3023(7.5)
	3	28262±12(6)	22203±286(3)	23181±356(4.4)	23091±319(4.5)	21661±67(1.6)	21638±68(1.4)	49301±3606(7.5)	49325±3575(7.5)
	4	28294±261(4.6)	27786±653(3.1)	28551±420(5.0)	28636±379(5.4)	26885±82(1.7)	26856±103(1.3)	62334±3972(7.5)	62334±3972(7.5)
	5	28727±249(3.0)	30571±479(5.5)	30571±479(5.5)	30693±441(5.5)	28421±139(1.6)	28440±125(1.8)	64233±5147(7.5)	64242±5138(7.5)
S4-B	1	47887±173(6)	40827±2573(3.3)	41301±476(4.4)	41242±464(4.3)	38564±117(1.6)	38549±139(1.4)	70144±4029(7.5)	70144±4029(7.5)
	2	43677±0(5.6)	43024±2961(4.7)	40667±448(3.7)	40804±492(4.0)	38055±128(1.5)	38061±172(1.5)	70219±3947(7.5)	70219±3947(7.5)
	3	50458±193(3.4)	50440±239(3.2)	52701±547(5.6)	52632±519(5.4)	50091±210(1.6)	50103±151(1.8)	83768±3444(7.5)	83768±3444(7.5)
	4	34946±288(5.6)	33974±1347(3.2)	34526±317(4.4)	34478±318(4.7)	32820±95(1.7)	32864±148(1.5)	62765±3949(7.5)	62765±3949(7.5)
	5	38698±4308(4.5)	38901±3107(4.8)	36166±349(4.4)	36195±289(4.4)	34049±163(1.6)	34016±143(1.4)	65365±4152(7.5)	65369±4147(7.5)
S4-C	1	42365±10(5.7)	42128±314(5.3)	35903±393(3.5)	35922±456(3.5)	33866±168(1.5)	33892±143(1.5)	61633±3417(7.5)	61633±3417(7.5)
	2	49614±26(5)	49797±74(6)	44052±394(3.4)	44087±340(3.6)	41842±170(1.4)	41878±204(1.6)	71240±3779(7.5)	71240±3779(7.5)
	3	50103±0(6)	49263±282(5)	41426±369(3.6)	41299±442(3.4)	39367±173(1.4)	39386±133(1.6)	71625±2776(7.5)	71625±2776(7.5)
	4	51311±0(6)	50781±48(5)	44940±369(3.4)	45099±410(3.6)	42895±168(1.4)	42939±179(1.6)	72347±3396(7.5)	72347±3396(7.5)
	5	46067±54(6)	45495±105(5)	40747±390(3.6)	40751±340(3.4)	38984±120(1.4)	38994±79(1.6)	66381±3802(7.5)	66381±3802(7.5)

A.2 The Pseudo-Code of the Instance Generator for Fitness Landscape Analysis

Algorithm 10: Pseudo code of the instance generator.

Input: Static instance, initial solution;

The full capacity of vehicles: Q ;

A configuration of dynamic event: C_{Event} ;

Factors' set for the selected event: Set_{factor} ;

Configurations of state factors: C_{OV} , C_{RQ} ;

Output: A DCARP instance I_1 with selected configuration.

```

1 if  $C_{OV} == few$  then
2   | Uniformly randomly select one vehicle as the outside vehicle;
3 else
4   | Select all vehicles as outside vehicles;
5 if  $C_{RQ} == small$  then
6   | Set  $RQ = 30\%$ 
7 else
8   | Set  $RQ = 90\%$ 
9 Determine positions for all outside vehicles that each vehicle's remaining capacities
   are about  $RQ$  percents of full capacity.
10 Set current instance as  $I_0$  if  $C_{Event} == CI$  then
11   | Call Function:  $I_1 = Apply\_Cost\_Increase\_Event(I_0, Set_{factor})$ ;
12 if  $C_{Event} == CD$  then
13   | Call Function:  $I_1 = Apply\_Cost\_Decrease\_Event(I_0, Set_{factor})$ ;
14 if  $C_{Event} == ND$  then
15   | Call Function:  $I_1 = Apply\_New\_Demand\_Event(I_0, Set_{factor}, Q)$ ;
16 if  $C_{Event} == NT$  then
17   | Call Function:  $I_1 = Apply\_New\_Tasks\_Event(I_0, Set_{factor}, Q)$ ;

```

Function: Apply *Cost Increase* Event

Input: DCARP instance; Configuration set: CI-N;

Output: DCARP instance applying with dynamic event.

- 1 **if** $CI-N == few$ **then**
 - 2 $p = 20\%$
 - 3 **else**
 - 4 $p = 80\%$
 - 5 Uniformly random select p of edges in the solution's path and make their costs be doubled.
-

Function: Apply *Cost Decrease* Event

Input: DCARP instance; Configuration set: CD-N;

Output: DCARP instance applying with dynamic event.

- 1 **if** $CD-N == few$ **then**
 - 2 $p = 20\%$
 - 3 **else**
 - 4 $p = 80\%$
 - 5 Uniformly random select p of edges in the solution's path and make their costs be halved.
-

Function: Apply *New Demand* Event

Input: DCARP instance;

Configuration set: (ND-N, ND-V);

The full capacity of vehicles: Q ;

Output: DCARP instance applying with dynamic event.

- 1 **if** $ND-N == few$ **then**
 - 2 $p = 20\%$
 - 3 **else**
 - 4 $p = 80\%$
 - 5 Uniformly random select p of tasks in the remaining tasks and save them into a set Set_{ND} . **if** $ND-V == small$ **then**
 - 6 $dm = \frac{Q}{|Set_{ND}|}$
 - 7 **else**
 - 8 $dm = \frac{4Q}{|Set_{ND}|}$
 - 9 Add demand dm to each task in Set_{ND} .
-

Function: Apply *New Tasks* Event

Input: DCARP instance;

Configuration set: (NT-P, NT-N, NT-V);

The full capacity of vehicles: Q ;

Output: DCARP instance applying with dynamic event.

- 1 Save all available edges (not task) in a list $List_{NT}$;
 - 2 **if** $NT-P == close$ **then**
 - 3 Sort $List_{NT}$ in ascending order according to the max distance of two nodes to the depot.
 - 4 **else**
 - 5 Sort $List_{NT}$ in descending order according to the max distance of two nodes to the depot.
 - 6 **if** $NT-N == few$ **then**
 - 7 $p = 20\%$
 - 8 **else**
 - 9 $p = 80\%$
 - 10 Select the front p of edges from $List_{NT}$ as the new tasks and save into Set_{NT} . **if**
 $NT-V == small$ **then**
 - 11 $dm = \frac{Q}{|Set_{ND}|}$
 - 12 **else**
 - 13 $dm = \frac{4Q}{|Set_{ND}|}$
 - 14 Add demand dm to each task in Set_{NT} .
-

References

- [1] Rafael Kendy Arakaki and Fábio Luiz Usberti. “An efficiency-based path-scanning heuristic for the capacitated arc routing problem”. In: *Computers & Operations Research* 103 (2019), pp. 288–295.
- [2] Mazhar Ansari Ardeh, Yi Mei, and Mengjie Zhang. “Genetic programming with knowledge transfer and guided search for uncertain capacitated arc routing problem”. In: *IEEE Transactions on Evolutionary Computation* 26.4 (2021), pp. 765–779.
- [3] Mazhar Ansari Ardeh, Yi Mei, Mengjie Zhang, and Xin Yao. “Knowledge transfer genetic programming with auxiliary population for solving uncertain capacitated arc routing problem”. In: *IEEE Transactions on Evolutionary Computation* 27.2 (2023), pp. 311–325.
- [4] Arjang A. Assad, Wen-Lea Pearn, and Bruce L. Golden. “The capacitated chinese postman problem: Lower bounds and solvable cases”. In: *American Journal of Mathematical and Management Sciences* 7.1-2 (1987), pp. 63–88.
- [5] Enrico Bartolini, Jean-François Cordeau, and Gilbert Laporte. “Improved lower bounds and exact algorithm for the capacitated arc routing problem”. In: *Mathematical Programming* 137.1 (2013), pp. 409–452.

- [6] Joaquín Bautista, Elena Fernández, and Jordi Pereira. “Solving an urban waste collection problem using ants heuristics”. In: *Computers & Operations Research* 35.9 (2008), pp. 3020–3033.
- [7] José-Manuel Belenguer, Enrique Benavent, Nacima Labadi, Christian Prins, and Mohamed Reghioui. “Split-delivery capacitated arc routing problem: Lower bound and metaheuristic”. In: *Transportation Science* 44.2 (2010), pp. 206–220.
- [8] José-Manuel Belenguer, Enrique Benavent, Philippe Lacomme, and Christian Prins. “Lower and upper bounds for the mixed capacitated arc routing problem”. In: *Computers & Operations Research* 33.12 (2006), pp. 3363–3383.
- [9] Enrique Benavent, Vicente Campos, Angel Corberán, and Enrique Mota. “The capacitated arc routing problem: Lower bounds”. In: *Networks* 22.7 (1992), pp. 669–690.
- [10] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. “Machine learning for combinatorial optimization: A methodological tour d’horizon”. In: *European Journal of Operational Research* 290.2 (2021), pp. 405–421.
- [11] Patrick Beullens, Luc Muyldermans, Dirk Cattrysse, and Dirk Van Oudheusden. “A guided local search heuristic for the capacitated arc routing problem”. In: *European Journal of Operational Research* 147.3 (2003), pp. 629–643.
- [12] Tim Blackwell and Jürgen Branke. “Multi-swarm optimization in dynamic environments”. In: *Workshops on Applications of Evolutionary Computation*. Springer. 2004, pp. 489–500.
- [13] Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuysse. “The vehicle routing problem: State of the art classification and review”. In: *Computers & Industrial Engineering* 99 (2016), pp. 300–313.

-
- [14] José Brandão and Richard Eglese. “A deterministic tabu search algorithm for the capacitated arc routing problem”. In: *Computers & Operations Research* 35.4 (2008), pp. 1112–1126.
- [15] Jürgen Branke. *Evolutionary optimization in dynamic environments*. Vol. 3. Springer Science & Business Media, 2012.
- [16] Jürgen Branke. “Memory enhanced evolutionary algorithms for changing optimization problems”. In: *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. Vol. 3. IEEE. 1999, pp. 1875–1882.
- [17] Yuning Chen, Jin-Kao Hao, and Fred Glover. “A hybrid meta-heuristic approach for the capacitated arc routing problem”. In: *European Journal of Operational Research* 253.1 (2016), pp. 25–39.
- [18] Feng Chu, Nacima Labadi, and Christian Prins. “A scatter search for the periodic capacitated arc routing problem”. In: *European Journal of Operational Research* 169.2 (2006), pp. 586–605.
- [19] Angel Corberán and Christian Prins. “Recent results on arc routing problems: An annotated bibliography”. In: *Networks* 56.1 (2010), pp. 50–69.
- [20] Ángel Corberán, Richard Eglese, Geir Hasle, Isaac Plana, and José María Sanchis. “Arc routing problems: A review of the past, present, and future”. In: *Networks* 77.1 (2021), pp. 88–115.
- [21] George Dantzig, Ray Fulkerson, and Selmer Johnson. “Solution of a large-scale traveling-salesman problem”. In: *Journal of the Operations Research Society of America* 2.4 (1954), pp. 393–410.
- [22] Fabio Daolio, Sébastien Verel, Gabriela Ochoa, and Marco Tomassini. “Local optima networks of the permutation flow-shop problem”. In: *International Conference on Artificial Evolution (Evolution Artificielle)*. Springer. 2013, pp. 41–52.

- [23] Janez Demšar. “Statistical comparisons of classifiers over multiple data sets”. In: *Journal of Machine Learning Research* 7. Jan (2006), pp. 1–30.
- [24] Sachin Desale, Akhtar Rasool, Sushil Andhale, and Priti Rane. “Heuristic and meta-heuristic algorithms and their relevance to the real world: A survey”. In: *Int. J. Comput. Eng. Res. Trends* 351.5 (2015), pp. 2349–7084.
- [25] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. “Ant colony optimization”. In: *IEEE Computational Intelligence Magazine* 1.4 (2006), pp. 28–39.
- [26] Richard W Eglese. “Routeing winter gritting vehicles”. In: *Discrete Applied Mathematics* 48.3 (1994), pp. 231–244.
- [27] Horst A Eiselt, Michel Gendreau, and Gilbert Laporte. “Arc routing problems, part II: The rural postman problem”. In: *Operations Research* 43.3 (1995), pp. 399–414.
- [28] Burak Eksioglu, Arif Volkan Vural, and Arnold Reisman. “The vehicle routing problem: A taxonomic review”. In: *Computers & Industrial Engineering* 57.4 (2009), pp. 1472–1483.
- [29] Casper Joost Eyckelhof and Marko Snoek. “Ant systems for a dynamic TSP”. In: *International Workshop on Ant Algorithms*. Springer. 2002, pp. 88–99.
- [30] Liang Feng, Yew-Soon Ong, Meng-Hiot Lim, and Ivor W Tsang. “Memetic search with interdomain learning: A realization between CVRP and CARP”. In: *IEEE Transactions on Evolutionary Computation* 19.5 (2014), pp. 644–658.
- [31] Liang Feng, Yew-Soon Ong, Quang Huy Nguyen, and Ah-Hwee Tan. “Towards probabilistic memetic algorithm: An initial study on capacitated arc routing problem”. In: *IEEE Congress on Evolutionary Computation*. IEEE. 2010, pp. 1–7.
- [32] Liang Feng, Yew-Soon Ong, Ah-Hwee Tan, and Ivor W Tsang. “Memes as building blocks: A case study on evolutionary optimization+ transfer learning for routing problems”. In: *Memetic Computing* 7 (2015), pp. 159–180.

-
- [33] Les Foulds, Humberto Longo, and Jean Martins. “A compact transformation of arc routing problems into node routing problems”. In: *Annals of Operations Research* 226.1 (2014), pp. 177–200.
- [34] Greg N Frederickson, Matthew S Hecht, and Chul E Kim. “Approximation algorithms for some routing problems”. In: *17th Annual Symposium on Foundations of Computer Science (SFCS 1976)*. IEEE. 1976, pp. 216–227.
- [35] Zvi Galil. “Efficient algorithms for finding maximum matching in graphs”. In: *ACM Computing Surveys (CSUR)* 18.1 (1986), pp. 23–38.
- [36] Bruce L Golden, James S DeArmon, and Edward K Baker. “Computational experiments with algorithms for a class of routing problems”. In: *Computers & Operations Research* 10.1 (1983), pp. 47–59.
- [37] Bruce L Golden and Richard T Wong. “Capacitated arc routing problems”. In: *Networks* 11.3 (1981), pp. 305–315.
- [38] Luís Gouveia, Maria Cândida Mourão, and Leonor Santiago Pinto. “Lower bounds for the mixed capacitated arc routing problem”. In: *Computers & Operations Research* 37.4 (2010), pp. 692–699.
- [39] Peter Greistorfer. “A tabu scatter search metaheuristic for the arc routing problem”. In: *Computers & Industrial Engineering* 44.2 (2003), pp. 249–266.
- [40] Maxime Guériau and Ivana Dusparic. “Quantifying the impact of connected and autonomous vehicles on traffic efficiency and safety in mixed traffic”. In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2020, pp. 1–8.
- [41] Michael Guntsch and Martin Middendorf. “Pheromone modification strategies for ant algorithms applied to dynamic TSP”. In: *Workshops on Applications of Evolutionary Computation*. Springer. 2001, pp. 213–222.

- [42] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. “Exploring network structure, dynamics, and function using NetworkX”. In: *Proceedings of the 7th Python in Science Conference*. 2008, pp. 11–15.
- [43] Hisashi Handa, Lee Chapman, and Xin Yao. “Dynamic salting route optimisation using evolutionary computation”. In: *2005 IEEE Congress on Evolutionary Computation*. Vol. 1. IEEE. 2005, pp. 158–165.
- [44] Hisashi Handa, Lee Chapman, and Xin Yao. “Robust route optimization for gritting/salting trucks: A CERCIA experience”. In: *IEEE Computational Intelligence Magazine* 1.1 (2006), pp. 6–9.
- [45] Franklin T. Hanshar and Beatrice M. Ombuki-Berman. “Dynamic vehicle routing using genetic algorithms”. In: *Applied Intelligence* 27.1 (2007), pp. 89–99.
- [46] Alain Hertz, Gilbert Laporte, and Michel Mittaz. “A tabu search heuristic for the capacitated arc routing problem”. In: *Operations Research* 48.1 (2000), pp. 129–135.
- [47] Alain Hertz and Michel Mittaz. “A variable neighborhood descent algorithm for the undirected capacitated arc routing problem”. In: *Transportation Science* 35.4 (2001), pp. 425–434.
- [48] Klaus Jansen. “Bounds for the general capacitated routing problem”. In: *Networks* 23.3 (1993), pp. 165–173.
- [49] Ya-Hui Jia, Yi Mei, and Mengjie Zhang. “A bilevel ant colony optimization algorithm for capacitated electric vehicle routing problem”. In: *IEEE Transactions on Cybernetics* 52.10 (2021), pp. 10855–10868.
- [50] Terry Jones. “Evolutionary algorithms, fitness landscapes and search”. PhD thesis. Citeseer, 1995.

-
- [51] Terry Jones and Stephanie Forrest. “Fitness distance correlation as a measure of problem difficulty for genetic algorithms”. In: *Proceedings of the 6th International Conference on Genetic Algorithms*. 1995, pp. 184–192.
 - [52] Philippe Lacomme, Christian Prins, and Wahiba Ramdane-Cherif. “Competitive memetic algorithms for arc routing problems”. In: *Annals of Operations Research* 131.1-4 (2004), pp. 159–185.
 - [53] Philippe Lacomme, Christian Prins, and Wahiba Ramdane-Chérif. “A genetic algorithm for the capacitated arc routing problem and its extensions”. In: *Workshops on Applications of Evolutionary Computation*. Springer. 2001, pp. 473–483.
 - [54] Philippe Lacomme, Christian Prins, and Wahiba Ramdane-Chérif. “Evolutionary algorithms for periodic arc routing problems”. In: *European Journal of Operational Research* 165.2 (2005), pp. 535–553.
 - [55] Philippe Lacomme, Christian Prins, and Alain Tanguy. “First competitive ant colony scheme for the CARP”. In: *International Workshop on Ant Colony Optimization and Swarm Intelligence*. Springer. 2004, pp. 426–427.
 - [56] Leon YO Li and Richard W Eglese. “An interactive algorithm for vehicle routeing for winter—gritting”. In: *Journal of the Operational Research Society* 47.2 (1996), pp. 217–228.
 - [57] Rui Li, Xinchao Zhao, Xingquan Zuo, Jianmei Yuan, and Xin Yao. “Memetic algorithm with non-smooth penalty for capacitated arc routing problem”. In: *Knowledge-Based Systems* 220 (2021), p. 106957.
 - [58] Jialin Liu, Ke Tang, and Xin Yao. “Robust optimization in uncertain capacitated arc routing problems: Progresses and perspectives”. In: *IEEE Computational Intelligence Magazine* 16.1 (2021), pp. 63–82.

- [59] Min Liu, Hemant Kumar Singh, and Tapabrata Ray. “A benchmark generator for dynamic capacitated arc routing problems”. In: *2014 IEEE Congress on Evolutionary Computation*. IEEE. 2014, pp. 579–586.
- [60] Min Liu, Hemant Kumar Singh, and Tapabrata Ray. “A memetic algorithm with a new split scheme for solving dynamic capacitated arc routing problems”. In: *2014 IEEE Congress on Evolutionary Computation*. IEEE. 2014, pp. 595–602.
- [61] Min Liu, Hemant Kumar Singh, and Tapabrata Ray. “Application specific instance generator and a memetic algorithm for capacitated arc routing problems”. In: *Transportation Research Part C: Emerging Technologies* 43 (2014), pp. 249–266.
- [62] Tiantang Liu, Zhibin Jiang, and Na Geng. “A memetic algorithm with iterated local search for the capacitated arc routing problem”. In: *International Journal of Production Research* 51.10 (2013), pp. 3075–3084.
- [63] Xiaofang Liu, Zhihui Zhan, Tianlong Gu, Sam Kwong, Zhenyu Lu, Henry Been-Lirn Duh, and Jun Zhang. “Neural network-based information transfer for dynamic optimization”. In: *IEEE Transactions on Neural Networks and Learning Systems* 31.5 (2020), pp. 1557–1570.
- [64] Yuxin Liu, Yi Mei, Mengjie Zhang, and Zili Zhang. “Automated heuristic design using genetic programming hyper-heuristic for uncertain capacitated arc routing problem”. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2017, pp. 290–297.
- [65] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. “Microscopic traffic simulation using SUMO”. In: *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018, pp. 2575–2582.

-
- [66] Eduyn Ramiro López-Santana, Germán Andrés Méndez-Giraldo, and Carlos Alberto Franco-Franco. “A hybrid scatter search algorithm to solve the capacitated arc routing problem with refill points”. In: *Intelligent Computing Theories and Application*. Springer International Publishing, 2016, pp. 3–15.
- [67] Katherine M Malan and Andries P Engelbrecht. “A survey of techniques for characterising fitness landscapes and some possible ways forward”. In: *Information Sciences* 241 (2013), pp. 148–163.
- [68] Katherine Mary Malan. “A survey of advances in landscape analysis for optimisation”. In: *Algorithms* 14.2 (2021), p. 40.
- [69] Vittorio Maniezzo and Matteo Roffilli. “Algorithms for large directed capacitated arc routing problem instances: Urban solid waste collection operational support”. In: *Recent Advances in Evolutionary Computation for Combinatorial Optimization* (2008), pp. 259–274.
- [70] Rafael Martinelli, Diego Pecin, Marcus Poggi, and Humberto Longo. “A branch-cut-and-price algorithm for the capacitated arc routing problem”. In: *Experimental Algorithms*. Springer Berlin Heidelberg, 2011, pp. 315–326.
- [71] Cristian Martinez, Irene Loiseau, Mauricio GC Resende, and S Rodriguez. “BRKGA algorithm for the capacitated arc routing problem”. In: *Electronic Notes in Theoretical Computer Science* 281 (2011), pp. 69–83.
- [72] Michalis Mavrovouniotis, Changhe Li, and Shengxiang Yang. “A survey of swarm intelligence for dynamic optimization: Algorithms and applications”. In: *Swarm and Evolutionary Computation* 33 (2017), pp. 1–17.
- [73] Michalis Mavrovouniotis and Shengxiang Yang. “Adapting the pheromone evaporation rate in dynamic routing problems”. In: *European Conference on the Applications of Evolutionary Computation*. Springer. 2013, pp. 606–615.

- [74] Michalis Mavrovouniotis, Shengxiang Yang, Mien Van, Changhe Li, and Marios Polycarpou. “Ant colony optimization algorithms for dynamic optimization: A case study of the dynamic travelling salesperson problem [research frontier]”. In: *IEEE Computational Intelligence Magazine* 15.1 (2020), pp. 52–63.
- [75] Yi Mei, Xiaodong Li, and Xin Yao. “Cooperative Coevolution With Route Distance Grouping for Large-Scale Capacitated Arc Routing Problems”. In: *IEEE Transactions on Evolutionary Computation* 18.3 (2014), pp. 435–449.
- [76] Yi Mei, Xiaodong Li, and Xin Yao. “Decomposing large-scale capacitated arc routing problems using a random route grouping method”. In: *2013 IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 1013–1020.
- [77] Yi Mei, Xiaodong Li, and Xin Yao. “Variable neighborhood decomposition for large scale capacitated arc routing problem”. In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 1313–1320.
- [78] Yi Mei, Ke Tang, and Xin Yao. “A global repair operator for capacitated arc routing problem”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.3 (2009), pp. 723–734.
- [79] Yi Mei, Ke Tang, and Xin Yao. “A memetic algorithm for periodic capacitated arc routing problem”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 41.6 (2011), pp. 1654–1667.
- [80] Yi Mei, Ke Tang, and Xin Yao. “Decomposition-based memetic algorithm for multi-objective capacitated arc routing problem”. In: *IEEE Transactions on Evolutionary Computation* 15.2 (2011), pp. 151–165.
- [81] Yi Mei, Ke Tang, and Xin Yao. “Evolutionary computation for dynamic capacitated arc routing problem”. In: *Evolutionary Computation for Dynamic Optimization Problems*. Springer, 2013, pp. 377–401.

-
- [82] Yi Mei, Ke Tang, and Xin Yao. “Improved memetic algorithm for capacitated arc routing problem”. In: *2009 IEEE Congress on Evolutionary Computation*. IEEE. 2009, pp. 1699–1706.
- [83] Kwan (1960) Mei-Ko. “Graphic programming using odd or even points”. In: *Chinese Math* 1 (1962), pp. 237–277.
- [84] Marcela Monroy-Licht, Ciro Alberto Amaya, André Langevin, and Louis-Martin Rousseau. “The rescheduling arc routing problem”. In: *International Transactions in Operational Research* 24.6 (2017), pp. 1325–1346.
- [85] R. Montemanni, L. M. Gambardella, A. E. Rizzoli, and A. V. Donati. “Ant colony system for a dynamic vehicle routing problem”. In: *Journal of Combinatorial Optimization* 10.4 (2005), pp. 327–343.
- [86] M Cândida Mourão and Leonor S Pinto. “An updated annotated bibliography on arc routing problems”. In: *Networks* 70.3 (2017), pp. 144–194.
- [87] Zsuzsanna Nagy, Ágnes Werner-Stark, and Tibor Dulai. “A data-driven solution for the dynamic capacitated arc routing problem”. In: *Proceedings of IAC in Budapest 2021* (2021), pp. 64–83.
- [88] Zsuzsanna Nagy, Ágnes Werner-Stark, and Tibor Dulai. “An artificial bee colony algorithm for static and dynamic capacitated arc routing problems”. In: *Mathematics* 10.13 (2022), p. 2205.
- [89] Trung Thanh Nguyen, Shengxiang Yang, and Juergen Branke. “Evolutionary dynamic optimization: A survey of the state of the art”. In: *Swarm and Evolutionary Computation* 6 (2012), pp. 1–24.
- [90] Shams K Nseef, Salwani Abdullah, Ayad Turkey, and Graham Kendall. “An adaptive multi-population artificial bee colony algorithm for dynamic optimisation problems”. In: *Knowledge-Based Systems* 104 (2016), pp. 14–23.

- [91] Gabriela Ochoa, Marco Tomassini, Sébastien Vérel, and Christian Darabos. “A study of NK landscapes’ basins and local optima networks”. In: *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*. 2008, pp. 555–562.
- [92] Gabriela Ochoa, Sébastien Verel, Fabio Daolio, and Marco Tomassini. “Local optima networks: A new model of combinatorial fitness landscapes”. In: *Recent Advances in the Theory and Application of Fitness Landscapes*. Springer, 2014, pp. 233–262.
- [93] Clifford S Orloff. “A fundamental problem in vehicle routing”. In: *Networks* 4.1 (1974), pp. 35–64.
- [94] Wasin Padungwech, Jonathan Thompson, and Rhyd Lewis. “Effects of update frequencies in a dynamic capacitated arc routing problem”. In: *Networks* 76.4 (2020), pp. 522–538.
- [95] Wen Lea Pearn. “Approximate solutions for the capacitated arc routing problem”. In: *Computers & Operations Research* 16.6 (1989), pp. 589–600.
- [96] Wen-Lea Pearn, Arjang Assad, and Bruce L Golden. “Transforming arc routing into node routing problems”. In: *Computers & Operations Research* 14.4 (1987), pp. 285–288.
- [97] Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L. Medaglia. “A review of dynamic vehicle routing problems”. In: *European Journal of Operational Research* 225.1 (2013), pp. 1–11.
- [98] Michael Polacek, Karl F Doerner, Richard F Hartl, and Vittorio Maniezzo. “A variable neighborhood search for the capacitated arc routing problem with intermediate facilities”. In: *Journal of Heuristics* 14 (2008), pp. 405–423.
- [99] Daniel Porumbel, Gilles Goncalves, Hamid Allaoui, and Tienté Hsu. “Iterated local search and column generation to solve Arc-Routing as a permutation set-covering problem”. In: *European Journal of Operational Research* 256.2 (2017), pp. 349–367.

-
- [100] C. Prins, N. Labadi, and M. Reghioui. “Tour splitting algorithms for vehicle routing problems”. In: *International Journal of Production Research* 47.2 (2008), pp. 507–535.
- [101] Samuel Raff. “Routing and scheduling of vehicles and crews: The state of the art”. In: *Computers & Operations Research* 10.2 (1983), pp. 63–211.
- [102] Marius Riekert, Katherine M Malan, and AP Engelbrecht. “Adaptive genetic programming for dynamic classification problems”. In: *2009 IEEE Congress on Evolutionary Computation*. IEEE. 2009, pp. 674–681.
- [103] Brenner Humberto Ojeda Rios, Eduardo C. Xavier, Flávio K. Miyazawa, Pedro Amorim, Eduardo Curcio, and Maria João Santos. “Recent dynamic vehicle routing problems: A survey”. In: *Computers & Industrial Engineering* 160 (2021), p. 107604.
- [104] Nasser R Sabar, Ashish Bhaskar, Edward Chung, Ayad Turkey, and Andy Song. “A self-adaptive evolutionary algorithm for dynamic vehicle routing problems with traffic congestion”. In: *Swarm and Evolutionary Computation* 44 (2019), pp. 1018–1027.
- [105] Horst Sachs, Michael Stiebitz, and Robin J Wilson. “An historical note: Euler’s Königsberg letters”. In: *Journal of Graph Theory* 12.1 (1988), pp. 133–139.
- [106] Luís Santos, João Coutinho-Rodrigues, and John R Current. “An improved ant colony optimization based algorithm for the capacitated arc routing problem”. In: *Transportation Research Part B: Methodological* 44.2 (2010), pp. 246–266.
- [107] Luís Santos, João Coutinho-Rodrigues, and John R. Current. “An improved heuristic for the capacitated arc routing problem”. In: *Computers & Operations Research* 36.9 (2009), pp. 2632–2637.
- [108] Yasufumi Saruwatari, Ryuichi Hirabayashi, and Naonori Nishida. “Node duplication lower bounds for the capacitated arc routing problem”. In: *Journal of the Operations Research Society of Japan* 35.2 (1992), pp. 119–133.

- [109] Ronghua Shang, Hongna Ma, Jia Wang, Licheng Jiao, and Rustam Stolkin. “Immune clonal selection algorithm for capacitated arc routing problem”. In: *Soft Computing* 20.6 (2015), pp. 2177–2204.
- [110] Aydin Sipahioglu, Gokhan Kirlik, Osman Parlaktuna, and Ahmet Yazici. “Energy constrained multi-robot sensor-based coverage path planning using capacitated arc routing approach”. In: *Robotics and Autonomous Systems* 58.5 (2010), pp. 529–538.
- [111] Peter F. Stadler and Santa Fe Institute. “Towards a theory of landscapes”. In: *Complex Systems and Binary Networks*. Springer, 1995, pp. 78–163.
- [112] Mariam Tagmouti, Michel Gendreau, and Jean-Yves Potvin. “A dynamic capacitated arc routing problem with time-dependent service costs”. In: *Transportation Research Part C: Emerging Technologies* 19.1 (2011), pp. 20–28.
- [113] Ke Tang, Yi Mei, and Xin Yao. “Memetic algorithm with extended neighborhood search for capacitated arc routing problems”. In: *IEEE Transactions on Evolutionary Computation* 13.5 (2009), pp. 1151–1166.
- [114] Ke Tang, Juan Wang, Xiaodong Li, and Xin Yao. “A scalable approach to capacitated arc routing problems based on hierarchical decomposition”. In: *IEEE Transactions on Cybernetics* 47.11 (2016), pp. 3928–3940.
- [115] Mohammad-H Tayarani-N and Adam Prügel-Bennett. “An analysis of the fitness landscape of travelling salesman problem”. In: *Evolutionary Computation* 24.2 (2016), pp. 347–384.
- [116] Mohammad-H Tayarani-N and Adam Prügel-Bennett. “On the landscape of combinatorial optimization problems”. In: *IEEE Transactions on Evolutionary Computation* 18.3 (2013), pp. 420–434.

-
- [117] Sarah L Thomson, Gabriela Ochoa, Sébastien Verel, and Nadarajen Veerapen. “Inferring future landscapes: Sampling the local optima level”. In: *Evolutionary Computation* 28.4 (2020), pp. 621–641.
 - [118] Hao Tong, Leandro L Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao. “A hybrid local search framework for the dynamic capacitated arc routing problem”. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 2021, pp. 139–140.
 - [119] Hao Tong, Leandro L Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao. “Towards novel meta-heuristic algorithms for dynamic capacitated arc routing problems”. In: *International Conference on Parallel Problem Solving from Nature*. Springer. 2020, pp. 428–440.
 - [120] Gündüz Ulusoy. “The fleet size and mix problem for capacitated arc routing”. In: *European Journal of Operational Research* 22.3 (1985), pp. 329–337.
 - [121] Pieter Vansteenwegen, Wouter Souffriau, and Kenneth Sörensen. “Solving the mobile mapping van problem: A hybrid metaheuristic for capacitated arc routing with soft time windows”. In: *Computers & Operations Research* 37.11 (2010), pp. 1870–1876.
 - [122] Sébastien Verel, Fabio Daolio, Gabriela Ochoa, and Marco Tomassini. “Local optima networks with escape edges”. In: *International Conference on Artificial Evolution (Evolution Artificielle)*. Springer. 2011, pp. 49–60.
 - [123] Sébastien Verel, Gabriela Ochoa, and Marco Tomassini. “Local optima networks of NK landscapes with neutrality”. In: *IEEE Transactions on Evolutionary Computation* 15.6 (2010), pp. 783–797.
 - [124] Juan Wang, Ke Tang, Jose A Lozano, and Xin Yao. “Estimation of the distribution algorithm with a stochastic local search for uncertain capacitated arc routing problems”. In: *IEEE Transactions on Evolutionary Computation* 20.1 (2015), pp. 96–109.

- [125] Shaolin Wang, Yi Mei, Mengjie Zhang, and Xin Yao. “Genetic programming with niching for uncertain capacitated arc routing problem”. In: *IEEE Transactions on Evolutionary Computation* 26.1 (2021), pp. 73–87.
- [126] Zhenzhong Wang, Min Jiang, Xing Gao, Liang Feng, Weizhen Hu, and Kay Chen Tan. “Evolutionary dynamic multi-objective optimization via regression transfer learning”. In: *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2019, pp. 2375–2381.
- [127] Zhurong Wang, Haiyan Jin, and Manman Tian. “Rank-based memetic algorithm for capacitated arc routing problems”. In: *Applied Soft Computing* 37 (2015), pp. 572–584.
- [128] Edward Weinberger. “Correlated and uncorrelated fitness landscapes and how to tell the difference”. In: *Biological Cybernetics* 63.5 (1990), pp. 325–336.
- [129] Thomas Weise, Alexandre Devert, and Ke Tang. “A developmental solution to (dynamic) capacitated arc routing problems using genetic programming”. In: *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*. ACM, 2012, pp. 831–838.
- [130] Sanne Wöhlk. *Contributions to arc routing*. Citeseer, 2006.
- [131] Sanne Wöhlk. “New lower bound for the capacitated arc routing problem”. In: *Computers & Operations Research* 33.12 (2006), pp. 3458–3472.
- [132] Xiaoshu Xiang, Ye Tian, Xingyi Zhang, Jianhua Xiao, and Yaochu Jin. “A pairwise proximity learning-based ant colony algorithm for dynamic vehicle routing problems”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.6 (2022), pp. 5275–5286.

-
- [133] Lining Xing, Philipp Rohlfshagen, Yingwu Chen, and Xin Yao. “An evolutionary approach to the multidepot capacitated arc routing problem”. In: *IEEE Transactions on Evolutionary Computation* 14.3 (2009), pp. 356–374.
- [134] Li-Ning Xing, P. Rohlfshagen, Ying-Wu Chen, and Xin Yao. “A Hybrid Ant Colony Optimization Algorithm for the Extended Capacitated Arc Routing Problem”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 41.4 (2011), pp. 1110–1123.
- [135] Mohamed El Yafrani, Marcella SR Martins, Mehdi El Krari, Markus Wagner, Myriam RBS Delgado, Belaid Ahiod, and Ricardo Lüders. “A fitness landscape analysis of the travelling thief problem”. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. 2018, pp. 277–284.
- [136] Shengxiang Yang, Yong Jiang, and Trung Thanh Nguyen. “Metaheuristics for dynamic combinatorial optimization problems”. In: *IMA Journal of Management Mathematics* 24.4 (2013), pp. 451–480.
- [137] Danial Yazdani, Ran Cheng, Cheng He, and Jürgen Branke. “Adaptive control of subpopulations in evolutionary dynamic optimization”. In: *IEEE Transactions on Cybernetics* 52.7 (2020), pp. 6476–6489.
- [138] Danial Yazdani, Ran Cheng, Donya Yazdani, Jürgen Branke, Yaochu Jin, and Xin Yao. “A survey of evolutionary continuous dynamic optimization over two decades—Part A”. In: *IEEE Transactions on Evolutionary Computation* 25.4 (2021), pp. 609–629.
- [139] Qi Zhao, Bai Yan, Yuhui Shi, and Martin Middendorf. “Evolutionary dynamic multiobjective optimization via learning from historical search process”. In: *IEEE Transactions on Cybernetics* 52.7 (2021), pp. 6119–6130.

References
