Transfer Optimisation in Genetic Programming for Solving Uncertain Capacitated Arc Routing Problem

by

Mazhar Ansari Ardeh

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Computer Science.

Victoria University of Wellington
2022

Abstract

Uncertain Capacitated Arc Routing Problem (UCARP) is a combinatorial optimisation problem with many important real-world applications. Genetic programming (GP) is a powerful machine learning technique that has been successfully used to automatically evolve routing policies for UCARP. Reusability is an open issue in the field of UCARP and in this direction, an open challenge is the case of scenario changes, e.g. change in the number of vehicles and probability distributions of random demands, which typically requires new training procedures to be initiated. Considering the expensive training cost of evolving routing policies for UCARP, a promising strategy is to learn and reuse knowledge from a previous problem-solving process to improve the effectiveness and efficiency of solving a new related problem, i.e. transfer learning.

The overall goal of this thesis is to develop novel knowledge transfer algorithms for GP for solving UCARP to handle environment changes more effectively and efficiently. To fulfil this goal, a plethora of machine learning techniques, i.e. surrogate models, feature selection, searching and specialised genetic operators, are utilised in this thesis.

First, this thesis explores the effectiveness of the existing transfer optimisation methods for solving UCARP. Accordingly, one of the main directions of this thesis is towards identifying the nature of transferable knowledge, which can impact the quality of knowledge transfer for GP to solve UCARP. For this purpose, a collection of the state-of-the-art transfer optimisation GP algorithms are evaluated for UCARP. After identifying some potential gaps in the literature, a number of preliminary transfer optimisation algorithms are proposed that supplement the literature. To evaluate

the algorithms, a large set of knowledge transfer scenarios with various source and target problems were designed based on real-world datasets. According to the results, none of the methods showed significant improvement in the effectiveness of the trained UCARP routing policies. These results revealed the need for more effective transfer optimisation methods specifically designed for UCARP. Furthermore, our investigations revealed that the presence of duplicates in knowledge sources is one of the main challenges for effective transfer optimisation in solving UCARP.

Second, we propose approaches to handling the presence of duplicates in the transferred knowledge. The first approach increases population diversity after knowledge transfer to counteract the loss of diversity that is introduced by the presence of duplicates in the transferred knowledge. In the second approach, the duplicates are removed from the transferred knowledge. Then, the transferred knowledge is utilised to create a diverse initial GP population of high-quality individuals. Both approaches are investigated through detailed experimental studies. The results indicate that, while the first approach did not perform better than GP with knowledge transfer, the second can improve the effectiveness of training routing policies with GP significantly.

Third, this thesis proposes a novel algorithm that transfers the phenotypic characteristics of the routing policies for solving the source problem. In the new algorithm, the most fit and unique source routing policies are utilised for initialising GP for solving the target problem. Then, a tabu list is placed on the source routing policies and the GP process is prohibited from recreating any of the source routing policies. The motivation for this approach is that, due to the existence of similarity between the source and target problems, source routing policies are unlikely to have a good performance for the target problem. Our experimental studies confirmed that by prohibiting GP from recreating source policies, and the computational resources will be spent on searching and evaluating new regions of the search space, which can lead to discovering better solutions.

Fourth, this thesis proposes a novel knowledge transfer algorithm based on the idea of maintaining the transferred knowledge as an auxiliary population. In this approach, first, the best individuals of the duplicate-free knowledge source are used to initialise GP. Additionally, these transferred individuals are also maintained as an auxiliary population and are evolved alongside the main population. To save the computational cost, the auxiliary population is evolved with a surrogate method. Additionally, an elaborate knowledge exchange mechanism between the two populations is devised that emphasises transferring high-quality and unique individuals, the transfer of which can improve the diversity of the receiving population. This allows GP to overcome the problem of losing its population diversity during the evolutionary process. Our detailed experimental results confirmed the superior performance of the proposed algorithm and confirmed that the proposed method improved the phenotypic diversity of GP population.

Acknowledgments

Earning a PhD degree is similar to completing an arduous journey, one that cannot be finished without the fellowship of the good souls that would help overcome all the obstacles on the way. I feel obliged to admit that I've been blessed with such fellowship to whom I would like to express my deepest gratitude. Above all, I thank Allah for blessings, wellness, and abilities, that He has given me during this work and in all my life.

First and foremost, I would like to express my great appreciation to my supervisors Dr. Yi Mei and Prof. Mengjie Zhang. Every discussion that we had was a great learning opportunity that helped me grow as a person, and not just as an academic.

Mahnaz Ebrahimi, Rose O'Connor and Gordon Campbell, I would like to dedicate this work to you. The love I have for you is immeasurable. You stood by me unconditionally during this period and I will be forever indebted to you. I would also like to thank you Valerie, my good old furry friend, for all the nice walks that we had and all the nights that you patiently listened to my complaints on why or how my experiments did not work well. Also, thank you Omid Khazaeian for your invaluable friendship.

I would like to express my thanks to the members of the Evolutionary Computation Research Group (ECRG); I will not forget all the fun that *I* had in our Friday meetings any time soon. Special thanks go to my dear friends Fangfang Zhang (for being a fantastic office mate and a good friend, I hope she will continue citing my papers), Baligh Al-helali (for be-

ing my *best friend* and for all the songs that he sang while we were trying to study), Mahdi Abdollahi (for all the valuable life lessons that he recited from Mashall Aryan (thank you Mashall too)), Harisu Abdullahi Shehu, Joao Costa, Hiroshika Hinduramage, Duleeka Munasinghe, Ghassem Narimani, Shabbir Abbasi, Victoria Huang, Ke Chen, Muru Raj Odiathevar, Apsara Wimalasiri, Shima Afzali, Samaneh Azari, Junaid Haseeb, Atefeh Talebian, Kirita-Rose Escott, Isabella Pimentel Pincelli, Ramya Anasseriyil Viswambaran, Ying Bi, Binzi Xu, and so many unlisted, who had been of a great help to ease my life during stressful times.

I also need to thank Victoria University of Wellington and its supportive staff. Particularly, I'd like to acknowledge that I would not be able to reach this final stage without the kind help of Patricia Stein, Diana Siwiak, Kristen Sharma and Suzie Paima. Also, I cannot deny that I wouldn't be able to run so many experiments for my research without the kind support (and the patience) of Mark Davies.

During the last year of my studies, I was fortunate to join the data analytics team of PikPok™ and fulfil a life-long dream of participating in the development of video games. I sincerely thank Laura White, Stella Ramirez, Erdem Alpkaya and Brett Ross for all their support, friendship, banters and valuable lessons.

Last, but most important, I would like to appreciate all the effort and hardship that my parents, Mansoor and Khadijeh, went through for bringing up and educating me. I wouldn't have achieved this great feat if it weren't for all their support.

Although this work is a pebble in the vast sky of science, I sincerely wish that it will contribute, however insignificantly, to the body of future works on which giants could stand.

تقدیم به وطنم ایران

تقدیم به مادرانم ناهید شیرپیشه، شهناز اکلی، کوهر عشقی، بهیه ناجو

و تمام مادرانی که دیدند که خون جگرگوشه را بر خاک دیدند و از مرگ شب گفتند

List of Publications

1. Ardeh, M. A., Mei, Y., Zhang, M. & Yao, X. "Genetic Programming with Auxiliary Population Transfer Optimisation for Solving Uncertain Capacitated Arc Routing Problem" IEEE Transactions on Evolutionary Computation, 2022.
   DOI: 10.1109/TEVC.2022.3169289

2. Ardeh, M. A., Mei, Y., & Zhang, M. (2021). "Genetic Programming with Knowledge Transfer and Guided Search for Uncertain Capacitated Arc Routing Problem". IEEE Transactions on Evolutionary Computation.
   DOI: 10.1109/TEVC.2021.3129278

3. Ardeh, M. A., Mei, Y., & Zhang, M. (2021) "A Novel Multi-Task Genetic Programming Approach to Uncertain Capacitated Arc Routing Problem". Proceedings of the ACM Genetic and Evolutionary Computation Conference (GECCO), 759-767.
   DOI: 10.1145/3449639.3459322

4. M. A. Ardeh, Y. Mei and M. Zhang, "Surrogate-Assisted Genetic Programming with Diverse Transfer for the Uncertain Capacitated Arc Routing Problem", 2021 IEEE Congress on Evolutionary Computation (CEC), 2021, pp. 628-635.
   DOI: 10.1109/CEC45853.2021.9504817

5. M. A. Ardeh, Y. Mei and M. Zhang, "A GPHH with Surrogate-assisted

Knowledge Transfer for Uncertain Capacitated Arc Routing Problem", 2020 IEEE Symposium Series on Computational Intelligence (SSCI), 2020, pp. 2786-2793.
DOI: 10.1109/SSCI47803.2020.9308398

6. M. A. Ardeh, Y. Mei and M. Zhangz, "Diversity-driven Knowledge Transfer for GPHH to Solve Uncertain Capacitated Arc Routing Problem", 2020 IEEE Symposium Series on Computational Intelligence (SSCI), 2020, pp. 2407-2414.
DOI: 10.1109/SSCI47803.2020.9308501

7. Ardeh, M. A., Mei, Y., & Zhang, M. (2020). "A Parametric Framework for Genetic Programming with Transfer Learning for Uncertain Capacitated Arc Routing Problem". AI 2020: Advances in Artificial Intelligence, 150–162.
DOI: https://DOI.org/10.1007/978-3-030-64984-5_12

8. Ardeh, M. A., Mei, Y., & Zhang, M. (2020). "Genetic Programming Hyper-Heuristics with Probabilistic Prototype Tree Knowledge Transfer for Uncertain Capacitated Arc Routing Problems". IEEE Congress on Evolutionary Computation (CEC), 1–8.
DOI: https://DOI.org/10.1109/CEC48606.2020.9185714

9. Ardeh, M. A., Mei, Y., & Zhang, M. (2019). "Transfer Learning in Genetic Programming Hyper-heuristic for Solving Uncertain Capacitated Arc Routing Problem". 2019 IEEE Congress on Evolutionary Computation (CEC) , 49-56
DOI: 10.1109/CEC.2019.8789920

10. Ardeh, M. A., Mei, Y., & Zhang, M. (2019). "A Novel Genetic Programming Algorithm with Knowledge Transfer for Uncertain Capacitated Arc Routing Problem". PRICAI 2019: Trends in Artificial Intelligence , 196–200.
DOI: 10.1007/978-3-030-29908-8_16

11. Ardeh, M. A., Mei, Y., & Zhang, M. (2019). "Genetic Programming Hyper-heuristic With Knowledge Transfer for Uncertain Capacitated Arc Routing Problem". GECCO '19: Proceedings of the Genetic and Evolutionary Computation Conference Companion , 334–335. DOI: 10.1145/3319619.3321988

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Capacitated Arc Routing Problem (CARP) is a well-known optimisation problem that is modelled on a connected undirected graph $G(V, E)$. Given a number of homogeneous vehicles each with a capacity, a CARP aims to determine a set of routes to serve tasks with minimal total costs and satisfy some predefined constraints [238]. Many real-world applications can be modelled as CARP, such as street watering, snow removal [38], waste collection [8], and scheduling meter reading [239]. Many variants of CARP have been proposed by scholars [238], [174]. However, most existing studies focused only on static CARP. It is difficult to apply the approaches to the static CARP to real-world and practical cases, in which problem parameters are often stochastic and unknown before edges are served or traversed. To overcome this limitation, Mei et al. proposed an uncertain model of CARP (denoted as UCARP) with four stochastic parameters, i.e., presence of tasks, demands of tasks, presence of paths and traversal costs of paths [162, 144].

One of the characteristics of UCARP is that many existing methods for solving the static CARP, e.g. mathematical programming [106, 147, 23], heuristic [90, 207, 91, 191] and meta-heuristic algorithms [66, 26, 93, 133, 29, 104], are not applicable for solving it. This is mostly attributed to the uncertain nature of this problem that when the environment changes so-

lutions for the problem also change and consequently, there are several methods that are specific to solving UCARP. One of the most flexible methods is the use of routing policy. A routing policy is a priority function that vehicles use to determine the priority of the available tasks to select and serve next. It has been shown that routing policy is a powerful tool for solving UCARP that can handle the dynamic and uncertain nature of this problem very effectively and is considered the state-of-the-art method for solving UCARP. Designing routing policies for UCARP manually requires extensive domain expertise and is time consuming. On the other hand, Genetic Programming (GP) can be utilised to automatically design effective routing policies through a machine learning paradigm. After a routing policy has been trained by GP from the training data, vehicles can use it to guide their serving tasks for solving the unseen UCARP test instances. The advantage of routing policy is that it can be applied to a set of problem instances rather than being restricted within a single problem instance.

A challenge of the GP-based routing policy learning approach, despite being more flexible than the traditional methods, is that when the characteristics of the problem, such as number of vehicles or nodes, probability distribution of stochastic variables or capacity of vehicles, change, the performance of the originally trained policy can deteriorate greatly. On the other hand, retraining a routing policy for the new problem from scratch can be time consuming. However, since the new problem has similarities to a previously solved problem, it is reasonable to believe that the knowledge in solving the old problem can be exploited for training effective policies for the new problem. This goal falls into the realm of transfer optimisation. By salvaging the knowledge in a solved problem, transfer optimisation techniques promise an easier path to better solutions for related but unsolved solution. Consequently, transfer optimisation is an ideal candidate for tackling the mentioned reusability problem.

Despite the fact that there are a plethora of knowledge transfer techniques available in the machine learning literature [186, 151] and transfer

optimisation [96], none of them has been considered as a hyper-heuristic approach in evolving routing policies of UCARP. Although some works have been done for similar problems [73], [157], to the best of our knowledge, the amount of effort in addressing the reusability issue of UCARP with transfer optimisation is very limited. Ergo, this thesis aims to propose new GP transfer optimisation methods and improve the efficiency and effectiveness of retraining routing policies for UCARP. The majority of current GP transfer methods mainly transfer subtrees and to the best of our knowledge, no GP transfer optimisation method has been proposed or verified in the context of GP hyper-heuristic. Therefore, an additional contribution of this thesis is to use UCARP as a benchmark problem and explore the potentials and shortcomings of various GP transfer optimisation methods for solving UCARP.

1.1 Motivation

As was mentioned in the introduction, one challenge in the applicability of the GPHH approach to solving UCARP is that when a change in characteristics of the problem happens for any reason, the routing policies cannot perform well for the new problem and therefore, they are not reusable [163]. Additionally, to the best of our knowledge, general routing policies, which are applicable to all problems with good performance, do not exist yet.

There is a practical aspect to this definition of reusability for UCARP as it has strong real-world implications. In many real-world applications of UCARP, such as transportation systems or delivery systems, it is very likely that problem properties change over time. For example, in transportation systems, it is very common to increase (or decrease) the number of vehicles in response to increase (or decrease) in number of passengers. On the other hands, it is also very routine for vehicles to be removed from the fleet due to break-downs or maintenance reasons. Similar situations

may occur for edges too. For example, it is not unusual to add new stations or stops to existing ones in response to increase in demand or to add more links between stations that are different from the existing ones. All the same, probability distribution of deadheading cost and demand also may depend on season, time of day, etc.

For many problems, training a solution from scratch, in response to changes in problem, may be a viable option but it is not the case for UCARP. There are two prominent reasons for this issue. The first reason is that UCARP is an NP-hard problem [197] and therefore, finding a solution for it is inherently a non-trivial task and is usually a time consuming process. This difficulty is corroborated with the fact that GP, as a method of evolving routing policies for UCARP, is also a time consuming approach too [**?**, 98]. This fact becomes more important when the second reason is considered; that is, the real-world applications of UCARP are considered in which usually hundreds of vehicles are involved and changes in aspects of the problem are frequent. Therefore, immediate and effective responses to the changes are desired. Furthermore, in such cases, the changes in the problem are not drastic and often, the new problem is similar and related to the problem it originated from. Additionally, even if the computational cost of solving a new problem is not an issue, the fact that the problems are related motivates utilising the similarity of the problems for finding better solutions for new problems.

Consequently, efficient and powerful methods are required to manage and handle the non-reusability property of UCARP. In this thesis, we aim to handle this issue and search for methods that can help overcome the computational cost of handling changes in problem aspects, and even, find more effective solutions, to the point that can be more suitable for real-world applications of UCARP.

The concept of reusing old solutions as transferable knowledge is an important field of research in the area of machine learning and artificial intelligence [186, 235, 151]. However, there are key differences that pre-

vent the applicability of the majority of these methods for UCARP. Among them, the most important factor is that transfer learning methods are generally tailored to the learning techniques that are utilised for solving a particular problem [186, 213, 142, 110, 215]. Because of this, and considering that GPHH is currently the state-of-the-art method for solving UCARP, only knowledge transfer methods that are developed for GP and Evolutionary Computation (EC) algorithms are viable candidates for handling the changes in problem aspects of UCARP.

To the best of our knowledge, the majority of transfer optimisation for EC algorithms is based on the transfer of whole or parts of solutions. However, we believe this approach has several limitations that we will endeavour to address. Firstly, they are mostly focused on creation of initial population of EC algorithms for solving the target problem and by doing so, the majority of them do not utilise the extracted knowledge any more [96, 11, 10], especially in the context of UCARP. Secondly, the improvement obtained from creating better-than-random initial population is usually lost after a few generations for the case of UCARP and therefore, it does not result into a better final performance [11]. Thirdly, current transfer optimisation methods based on transfer of (sub-)trees do not take any precautions against code bloats and redundancies and their performance will be affected negatively if the source problem contains these issues [96, 220].

To overcome the issues mentioned above, different GP-based transfer optimisation approaches will be proposed and investigated in this thesis.

1.2 Research Goals

The overall goal of this thesis is to develop novel GP-based transfer optimisation methods that can benefit from the knowledge that exists in a solved instance of UCARP to improve effectiveness and efficiency of evolving new routing policies for UCARP when a change in problem prop-

erties is occurred. In this direction, the effort will be put to take advantage of GP features and characteristics but, considering the fact that the ultimate goal is to handle scenario changes of UCARP, a special attention will be paid to the characteristic features of UCARP and the ways that the domain knowledge from UCARP can be exploited to facilitate knowledge transfer. Specifically, this thesis is focused on identifying and confirming the challenges that can deteriorate the performance of existing transfer optimisation algorithms for UCARP and propose novel methods for extracting reusable knowledge from a solved UCARP problem. This work will utilise the extracted knowledge effectively for initialising GP and during the search for solving the target problem. Finally, this thesis will devise a novel method for using the transferred knowledge for preventing the GP process for solving the target problem from suffering from the same issues that afflicted the search for solving the source problem. The details of each objective are as follows.

The first objective of this thesis is to investigate the applicability of the existing transfer optimisation algorithms for solving UCARP. For this purpose, first a set of state-of-the-art existing transfer optimisation algorithms are considered for handling the problem changes of UCARP in Chapter 3. Upon investigating these algorithms, a set of shortcomings are also identified in the literature of transfer optimisation and a set of novel algorithms are also proposed in the same chapter. In this regard, first this thesis proposes two algorithms for evaluating the importance of sub-trees for transfer. Second, this thesis proposes a novel algorithm for learning and transferring probability distributions of the routing policies. Third, by proposing these methods, one important goal in this thesis is to evaluate the performance of these methods, and compare them with a set of existing GP transfer methods for solving UCARP problems. Last but not least, we aim to identify the important challenges that may exist in the way of performing transfer optimisation successfully for UCARP. According to this goal, in this thesis we have identified the lack of diversity and the pres-

ence of duplicates in the knowledge source as one of the main challenges against the effectiveness of the transfer optimisation for solving UCARP.

The second objective of this thesis is to propose novel transfer optimisation methods for creating high-quality initial population for GP for solving the target problem. In this thesis, we will propose two approaches to handling the lack of diversity in the knowledge source. In the first approach, it is tried to handle the issue after the GP initialisation with the transferred knowledge. In the second approach, the duplicates in the knowledge source are removed before initialising GP for solving the target problem. Since the removal of the duplicates may remove a significant amount of high-quality individuals, two algorithms are proposed for compensating the loss in the high-quality source individuals through utilising surrogate models that are learned from the transferred knowledge. Surrogate models are a category of machine learning techniques that allow approximating the fitness of routing policies with a very low computational overhead. It should be pointed out that, after the initialisation, the surrogate models can also be utilized to improve algorithm efficiency. However, this is not investigated in this thesis mainly because the focus of this work is on devising knowledge transfer methods that are suitable for solving UCARP and also, because the effectiveness of surrogate methods for GP has been investigated previously [105]. The set of individuals that have been evaluated for solving the source problem provide a rich training set for constructing surrogate models and the low computational cost of the model allows estimating the quality of a large number of individuals. It is important to remember that the suitability of using a surrogate derived from the source problem on the target problem may depend critically on the similarity between the source and target problems. As a result, when there is good similarity between the problems, by using the surrogate model, it is possible to augment the set of available transferred knowledge by adding more diverse and high-quality solutions, whose quality has been evaluated with the surrogate model. We conduct experiments

to investigate the efficiency of the proposed methods. In this regard, this work will also investigate how the proposed methods will address the challenges that were identified when fulfilling the first objective.

The third objective of this thesis is to propose a new transfer optimisation method that utilises the transferred knowledge after initialisation in a novel way and during the GP search for solving the target problem. Consequently, this thesis will first propose that the phenotypic information about the source problem has the potential for being reused during the evolutionary process for solving the target problem. In this regard, the phenotypic information pertain to the information about the behaviour of the routing policies which are the priorities that they calculate for tasks and the decisions that they make. This is in contrast with the genotypic information which is the information about the GP functions, terminals and subtrees that constitute the routing policies. Second, the thesis will propose novel crossover and mutation operators that will use the transferred knowledge in a sophisticated way. Finally, through extensive experimental studies, we will demonstrate how the phenotypic approach to transfer optimisation can outperform the genotypic methods.

The fourth objective of this thesis is to propose a novel method for evolving the transferred knowledge during the evolutionary search and using it for helping GP overcome the loss of diversity issue during the search process. In this new approach, GP is equipped with two populations. The main population is an standard GP population that is initialised with the transferred knowledge and is evolved with the main fitness evaluation. The second population is also initialised with the transferred knowledge but unlike the main population, a surrogate model is utilised to evaluate it. Second, this thesis will devise an elaborate method for knowledge exchange between the main and the auxiliary populations. In this regard, the knowledge exchange mechanism is designed to prevent sending redundant knowledge from one population to another. The mechanism also uses the exchange of knowledge as an opportunity to re-

move redundant individuals from the populations and by doing so, increase the population diversity and final performance. Finally, this thesis will demonstrate how the auxiliary population and the exchange mechanism can increase effectiveness of GP for solving UCARP.

1.3 Major Contributions

This thesis makes the following contributions:

- This thesis has investigated the potential of transfer optimisation for UCARP and identified potential pitfalls that one may face when applying EC-based knowledge transfer. This is achieved through investigating the transfer of GP subtrees and probability distributions of good solutions for solving UCARP. In this regard, we developed two measures for assessing the importance of subtrees and transfer. Additionally, we developed a method for learning the probability distributions of high-quality source solutions. The learned probability distributions are then utilised for generating high-quality solutions. The developed methods and a set of state-of-the-art transfer optimisation algorithms were then investigated through solving a large number of transfer optimisation scenarios. Our investigations revealed that the presence of duplicates in the knowledge source and corresponding lack of diversity plays a significant negative role in the effectiveness of the knowledge transfer for UCARP.

  Parts of this contribution have been published in:

  Mazhar Ansari Ardeh, Yi Mei, Mengjie Zhang, "Transfer Learning in Genetic Programming Hyper-heuristic for Solving Uncertain Capacitated Arc Routing Problem", IEEE Congress on Evolutionary Computation (IEEE CEC 2019), 49–56

  Mazhar Ansari Ardeh, Yi Mei, Mengjie Zhang, "A Novel Genetic Programming Algorithm with Knowledge Transfer for Uncertain Ca-

pacitated Arc Routing Problem", The 16th Pacific Rim International Conference on Artificial Intelligence (PRICAI 2019), 196–200

Mazhar Ansari Ardeh, Yi Mei, Mengjie Zhang, "Genetic Programming Hyper-Heuristics with Probabilistic Prototype Tree Knowledge Transfer for Uncertain Capacitated Arc Routing Problems", IEEE Congress on Evolutionary Computation (IEEE CEC 2020), 1–8

- This thesis has proposed three effective transfer optimisation methods for initialising GP population for solving the target task. This is achieved through addressing the problem that is caused by the presence of duplicates in the knowledge source. The focus of the proposed knowledge transfer methods was placed on transferring high-quality source materials without introducing redundancy and duplicates into the target population. To achieve this, we proposed increasing the mutation rate of GP temporarily after performing knowledge transfer. This could let GP mutate its population more freely and increase its diversity. Additionally, we proposed two novel algorithms that, instead of increasing the diversity after transfer, focused on creating an initial population that already has a high degree of diversity and does not contain redundancies. To achieve this, we proposed employing the set of individuals that were found for solving the source problem for training surrogate models. The low computational cost of surrogate models allows searching for high-quality, yet unique solutions that could create a good initial population for GP. Our experimental studies confirmed that the surrogate-based methods can outperform the existing algorithms and verify that the lack of diversity in the knowledge source can reduce the effectiveness of knowledge transfer significantly.

  Parts of this contribution have been published in:

  Mazhar Ansari Ardeh, Yi Mei, Mengjie Zhang, "Diversity-driven Knowledge Transfer for GPHH to Solve Uncertain Capacitated Arc

Routing Problem" MA Ardeh, Y Mei, M Zhang 2020 IEEE Symposium Series on Computational Intelligence (SSCI), 2407–2414

Mazhar Ansari Ardeh, Yi Mei, Mengjie Zhang, "A GPHH with Surrogate-assisted Knowledge Transfer for Uncertain Capacitated Arc Routing Problem" 2020 IEEE Symposium Series on Computational Intelligence (SSCI), 2786–2793

Mazhar Ansari Ardeh, Yi Mei, Mengjie Zhang, "Surrogate-Assisted Genetic Programming with Diverse Transfer for the Uncertain Capacitated Arc Routing Problem" 2021 IEEE Congress on Evolutionary Computation (IEEE CEC 2021), 628–635

- This thesis has proposed an effective and novel method for capturing the phenotypic behaviour of routing policies as the transferable knowledge and utilising it effectively during the search process for solving a target UCARP instance. To reuse the transferred phenotypic knowledge, modified crossover and mutation operators are proposed. Whenever these operators create an offspring, they check the phenotypic behaviour of the offspring against the set of transferred phenotypic knowledge and if the offspring has the same behaviour as any item in the transferred set, it is discarded and the operators try to create another offspring whose behaviour is not observed when solving the source problem. The motivation behind this seemingly counter-intuitive knowledge transfer approach is based on the speculation that if the phenotypic behaviour of an individual has been seen during the search for solving the source problem, then it is unlikely for the individual to have a good performance for the target problem. This is because if the individual had a good performance, then it is used for initialising GP and there is no need to examine it again during the evolutionary process. On the other hand, if the performance of that individual was bad for the source problem, then its performance is likely to be bad for solving the target prob-

lem due to the existence of similarity between the source and target problems. Our experimental studies confirmed that the proposed approach is indeed very effective and the computational cost that is prevented from being wasted on such individuals can be allocated for searching new solutions.

Parts of this contribution have been published in:

Mazhar Ansari Ardeh, Yi Mei, Mengjie Zhang "Genetic Programming with Knowledge Transfer and Guided Search for Uncertain Capacitated Arc Routing Problem", 2021, IEEE Transactions on Evolutionary Computation (DOI: 10.1109/TEVC.2021.3129278)

- This thesis has proposed an effective method for adapting the transferred knowledge to the target problem and using it for preventing GP from losing its population diversity. As will be demonstrated in Chapter 3, the UCARP process for solving the source problem is prone to losing its population diversity, which in turn, will affect the quality of search for solving the source problem. However, the issue is that the GP process for solving the target problem may have the same problem and may lose its population diversity. To overcome this issue, this thesis proposes an algorithm that maintains the transferred knowledge in a separate auxiliary population that is evolved along side the main population. The main goal of the auxiliary population is to increase the exploration capability of GP. For this, an elaborate knowledge exchange mechanism between the main and the auxiliary population is proposed that strives to increase both the quality and diversity of the populations through replacing the phenotypic duplicates with high-quality individuals that are not in the population. Through our experimental studies, we verified that the proposed approach is capable of helping GP maintain its population diversity, which in turn, leads to its overall increased performance.

  Parts of this contribution have been published in:

Mazhar Ansari Ardeh, Yi Mei, Mengjie Zhang, Xin Yao "Genetic Programming with Auxiliary Population Transfer Optimisation for Solving Uncertain Capacitated Arc Routing Problem" 2022 IEEE Transactions on Evolutionary Computation (DOI: 10.1109/TEVC.2022.3169289)

1.4 Organisation of the Thesis

This thesis is organised as follows. Chapter 2 presents a literature review of uncertain capacitated arc routing problem and the methodologies for solving them. The chapter also gives a detailed overview of genetic programming and the way it can be used for solving UCARP. A comprehensive overview of transfer optimisation, especially EC-based transfer optimisation is also given in Chapter 2. Then, our achievements on the four research objectives are presented in four chapters from Chapter 3 to Chapter 7. Chapter 8 concludes this thesis. An overview of each chapter is shown as follows.

Chapter 2 presents the literature review, including vehicle routing, evolutionary computation, GP, and hyper-heuristics. The detailed descriptions of the CARP problem with a focus on UCARP are given in Section 2.1 and it also reviews how to use routing policies for solving UCARP. This chapter also provides details on how to use GPHH to evolve routing policies for UCARP in Section 2.5 automatically. In addition, existing studies related to transfer optimisation and other topics that are related to the research objectives of this thesis are discussed in details in Chapter 2.

Chapter 3 is focused on investigating the applicability of transfer optimisation for solving UCARP. Accordingly, it first describes some of the shortcomings of the existing methods and proposes a set of transfer optimisation algorithms with measuring the importance of subtrees and learning probability distributions of the good source solutions to address the identified shortcomings. Details of the proposed algorithms are given in this chapter. The efficiency and effectiveness of the proposed algorithms,

as well as some of the state-of-the-art existing algorithms, are verified along with further analyses. In this chapter, the lack of diversity in the knowledge source is identified as one of the main pitfalls to the effectiveness of knowledge transfer.

Chapter 4 presents the three proposed novel diversity-driven transfer optimisation methods for solving the target problem with GP. Based on the proposed algorithms, this chapter further describes two new GPHH initialisation algorithms that utilise surrogate models. Also, this chapter describes several mutation adaptation strategies to increase the population diversity after knowledge transfer. The performance of the proposed algorithms are investigated in this chapter and the effect of the proposed algorithms on the diversity of GP population is evaluated. The surrogate models, mutation adaptation strategies and phenotypic characterisation are the main techniques in this chapter.

Chapter 5 introduces the newly proposed transfer optimisation algorithm based on the transfer of phenotypic behaviours for generating offspring that have not been created for solving the source problem. This chapter describes the proposed modifications to the initialisation, crossover and mutation operators based on the transferred phenotypic behaviour. This chapter also investigates the effect and contribution of each of the proposed operators to the overall performance of GP. In addition, the act of preventing the recreation of source individual will be investigated to measure how it can save the computational cost of solving the target problem.

Chapter 6 introduces the proposed transfer optimisation algorithm with auxiliary population for solving UCARP. This chapter describes how the main and the auxiliary populations are initialised, how the auxiliary population is evolved alongside the main population, how a surrogate model is utilised for evolving the auxiliary population and how the surrogate model of the auxiliary population is updated during the evolutionary process. Furthermore, the knowledge exchange mechanism between the main

and the auxiliary populations, which is an essential component of the algorithm, is also explained with details. The chapter concludes with a thorough investigation of performance of the proposed algorithm in which the contribution of each novel component of the algorithm to its final performance will be discussed. Last, the sizes and semantic insight of the evolved scheduling heuristics are further studied.

Chapter 7 summaries the achieved objectives and the main conclusions of this thesis. Some discussions and future research directions are also presented in this chapter.

Chapter 2

Literature Survey

2.1 UCARP

UCARP [162] is an extension of the Capacitated Arc Routing Problem (CARP), first proposed by Golden and Wong [91]. It is an optimisation problem that is modelled on a connected undirected graph $G(V, E)$ in which nodes represent locations and edges represent routes between locations with non-negative costs and demands on edges [91]. Given a number of homogeneous vehicles, each with a limited capacity, CARP aims to determine a set of routes to serve the required edges with minimal total costs and satisfying some predefined constraints [238, 145].

Current research trend mostly focuses on static CARP [161, 45]. In case of static CARP, demands and deadheading costs along edges are static and deterministic. Despite its appeals, it is difficult to apply the static CARP to real-world and practical cases in which problem parameters are often stochastic and unknown before edges are served or traversed [162]. For example, in waste collection problem, the amount of garbage on streets may change weekly and only can be known exactly when serving them [223]. Stochastic CARP (SCARP), first devised by Fleury et al. [78], is one of the attempts to overcome the aforementioned shortcoming of static CARP. In SCARP, the demand is stochastic and this provides a more prac-

tical and realistic view of the CARP. Fleury et al. [78] analysed robustness of this model and utilized a Memetic Algorithm to solve it. Despite its effectiveness, SCARP assumes that all parameters of CARP are deterministic except demand of task and this property limits its real-world applicability. To overcome this limitation, Mei et al. [162] proposed an uncertain model of CARP in (denoted as UCARP) with four stochastic parameters, i.e., presence of tasks, demands of tasks, presence of paths and traversal costs of paths and later, Wang et. al [223, 224] and Liu et. al [145] devised a few novel evolutionary methods for solving UCARP.

A UCARP problem is a graph $G(V, E)$ in which V is the set of nodes of the graph and E is the set of edges. The set of edges is comprised of two subsets $E = E_T \cup E_U$. E_T is the set of required tasks that must be served while on the other hand, members of E_U are not needed to be served. In UCARP, each vehicle has a capacity Q and at the beginning, all vehicles are located at the depot $v_0 \in V$. Each edge $e \in E$ has two cost values associated: (1) a positive stochastic deadheading cost $dc(e)$ and, (2) a non-negative deterministic serving cost $sc(e)$. Each edge also has a non-negative stochastic demand $d(e)$. For non-required edges $e \in E_U$ demand value is zero, $d(e) = 0$. An edge with $d(e) > 0$ is called a task and should be served. To solve a UCARP means to find routes for vehicles to serve all tasks so that the sum of all serving and deadheading costs in each routes is minimised. In solving a UCARP, following constraints must be respected:

- All vehicles need to start their routes from the depot and end it at depot. Vehicles are allowed to visit the depot multiple times to replenish their capacity.
- Total amount of demand that each vehicle serves cannot be more than its capacity between two visits of the depot.

A sampled UCARP instance is a CARP instance in which stochastic variables are sampled from their distributions. To be specific, considering a UCARP instance I, a sampled instance I_ξ for I is obtained by sam-

pling each random demand $d_\xi(e), \forall e \in E_T$ and each deadheading cost $dc_\xi(e), \forall e \in E$ under the environment (e.g random seed) represented by ξ. As a result, a sampled UCARP is different from a CARP instance in the following aspects:

- For demand of tasks of sampled UCARP, the actual value is revealed after the vehicle finishes serving it while for the case of CARP, it is available when starting the task.
- For deadheading costs, the sampled value is revealed after the vehicle has finished traversing it while for the case of CARP, it is available in advance.

Because of the aforementioned features, two failures may happen in a solution to a sampled UCARP instance:

- *Edge failure*: This case happens when the edge to be served is inaccessible. This can be denoted as $dc(e) = \infty$.
- *Route failure*: This case happens when the demand of the task is larger than the remaining capacity of the vehicle trying to serve it.

In case of an edge failure, the vehicle needs to find a detour by finding the shortest path (e.g. Dijkstra's algorithm). In case of an route failure, the vehicle needs to return to the depot in the middle of the service to replenish its capacity.

A solution to a sampled UCARP can be represented with $S = (X, Y)$ in which $S.X = \{S.X^(1), ..., S.X^(m)\}$ is a set of vertex sequences $S.X^{(k)} = (S.x_1^{(k)}, ..., S.x_{L_k}^{(k)})$ that represents the k^{th} route and $S.Y$ is a set of continuous vectors $\{S.Y^{(1)}, ..., S.Y^{(k)}\}$ with $(S.y_1^{(k)}, ..., S.y_{L_{K-1}^{(k)}}$ $(S.y_i^{(k} \in [0, 1])$ that denotes the fraction of demand that is served along the route $S.X^{(k)}$. With this representation, UCARP can be formulated as [162, 145]:

$$\min E_{\xi \in \Xi}[C(S_\xi)] \tag{2.1}$$

$$s.t.: S_\xi.x_1^{(k)} = S_\xi.x_{L_k}^{(k)} = v_0, \forall k = 1, 2, ..., m \tag{2.2}$$

$$\sum_{k=1}^{m} \sum_{i=1}^{L_k - 1} S_\xi.y_i^{(k)} \times S_\xi.z_i^{(k)}(e) = 1, \forall e : d_\xi(e) > 0, \tag{2.3}$$

$$\sum_{k=1}^{m} \sum_{i=1}^{L_k - 1} S_\xi.y_i^{(k)} \times S_\xi.z_i^{(k)}(e) = 0, \forall e : d_\xi(e) = 0, \tag{2.4}$$

$$\sum_{i=1}^{L_k - 1} d_\xi \left(S_\xi.x_i^{(k)}, S_\xi.x_{i+1}^{(k)} \right) \times S_\xi.y_i^{(k)} \leq Q, \forall k = 1, 2, ..., m \tag{2.5}$$

$$\left(S_\xi.x_i^{(k)}, S_\xi.x_{i+1}^{(k)} \right) \in E \tag{2.6}$$

$$S_\xi.y_i^{(k)} \in [0, 1] \tag{2.7}$$

In equation 2.3, $S_\xi.z_i^{(k)}(e)$ is defined as:

$$S_\xi.z_i^{(k)}(e) = \begin{cases} 1 & \text{if } \left(S_\xi.x_i^{(k)}, S_\xi.x_{i+1}^{(k)} \right) = e \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

For UCARP, the problem objective is to find a solution with the best expected cost under all possible uncertain environments which expresses solution robustness. Equation 2.1 defines the objective function with this consideration in which the expected total cost $C(S_\xi)$ of the solution S_ξ is minimised for all possible environments $\xi \in \Xi$. Equation 2.2 expresses the constraint that all routes need to start and end with the depot. Equations 2.2 and 2.3 formulate the constraint that each task must be served exactly once and each non-required edge is not served. The capacity constraint is formulated with Equation 2.5. It should be noted that in general, the solution S_ξ may be different for each environment ξ.

UCARP has two types of challenges, one from the NP-hardness of CARP [197], and the other from the uncertain environment, which requires adjusting/making decisions in real time when the environment is

detected to be different from expected. Existing studies have taken into account different real-world conditions into the problem model. However, most of them focused on static CARP, where the environment is known in advance. As a result, their solutions cannot be directly applied to UCARP due to the *route failures*. That is, the actual demand of an edge can be greater than expected, and the vehicle capacity is expired in the middle of the service.

2.2 Basic Concepts

This section presents a basic overview of machine learning and evolutionary computation.

2.2.1 Machine Learning

Machine learning is the computational approach of using experience (represented in the form of data) to perform tasks more efficiently and recognize patterns [169]. As a sub-field of artificial intelligence [204], machine learning is focused on enabling computers to learn from experience automatically and without being explicitly programmed.

From a general point of view, machine learning is comprised of three main tasks: (1) supervised learning, (2) unsupervised learning, and (3) reinforcement learning. In supervised learning problems (e.g. regression and classification), the desired outcome or outputs of the problem are known in the available data (i.e. the data is labelled). Conversely, the desired outputs are not known (i.e. the data is not labelled) in unsupervised problems (e.g. clustering). In reinforcement learning, the learning agent interacts with an environment and based on the feedback it receives from the environment, learns to optimise its performance in the environment.

In this thesis, makes use of two important machine learning techniques which are transfer learning and surrogate models. Additionally, the con-

cepts in this thesis have a close relationship with multi-task learning. Transfer learning is the approach of gaining knowledge from some solved source problem(s) and reusing the knowledge to solve related new target problems more effectively and efficiently [235]. Multi-task learning is closely related to transfer learning in the sense that it is also focused on gaining knowledge from solving a problem to improve the process of finding solutions for related problems [39]. However, in multi-task learning, all the tasks are solved simultaneously and the gained knowledge is shared as each problem is being solved. A detailed overview of transfer learning is given in Section 2.6.

Surrogate models are a set of techniques that allow constructing cheap models for estimating the outcome of some functions of interest that are computationally expensive [120]. In the context of evolutionary algorithms, surrogate models have proven to be of great importance for optimising computationally expensive problems [120]. Surrogate models are described with more details in Section 2.7.5.

2.2.2 Evolutionary Computation

Within the area of artificial intelligence, Evolutionary Computation (EC) [18] is a computational intelligence technique that is inspired from natural population-based evolution. The fundamental idea in evolutionary computation is that individuals in a population are improved generation by generation. EC algorithms can be divided into two categories of (1) evolutionary algorithms and (2) swarm intelligence. Genetic Algorithm (GA) [135], GP, evolutionary strategy [137] and evolutionary programming [80] are some of the important evolutionary algorithms. Particle swarm optimisation [67], ant colony optimisation [40] and artificial bee colony [2] are some of the widely-used swarm intelligence algorithms. Evolutionary algorithms, and GP in particular, are the focus of this thesis.

Typically, an evolutionary algorithms starts with creating a population

of individuals. The individuals represent potential solutions to the problem that is intended to be solved and initially, they are usually generated randomly. The individuals are then evaluated with a fitness function that measures how good, or *fit*, they are for solving the problem. Then, a new population is generated from the current population with genetic operators, such as crossover, mutation and reproduction. For this, the individuals with better fitness values are given higher chances to be selected for producing offspring. The process of creating a new population from the current one is then continued until some stopping criteria is met. Finally, the individual with the best fitness value is selected as the output of the algorithm.

GP is a famous evolutionary algorithm which also the focus of this thesis and a detailed overview of this method is given is Section 2.3.

2.3 Genetic Programming

GP is derived from Genetic Algorithm in which the members of the algorithm's population are computer programs. Popularised by Koza, it utilises evolutionary operations to evolve computer programs [130, 59] and has been used extensively for solving a vast variety of problems [249], [3], [16]. As is depicted in Algorithm 2.1, GP for evolving routing policies has four general steps:

1. Create an initial population of programs.
2. Evaluate the fitness of each individual in the population.
3. Select individuals based on fitness value.
4. Use the genetic operators such as crossover, mutation and reproduction to create new individual(s) from the selected individual(s).
5. Repeat the steps (2) through (4) until the stopping criteria are met.

In the rest of this section, GP and its features are described in more detail.

Algorithm 2.1: Pseudo-code of standard GP

Input : Training instances
Output: The best evolved function ind^*

```
while N_ind < Popsize do
    Initialisation: Randomly initialise each individual
end
set ind* ← null and fitness(ind*) ← +∞
gen ← 0
while gen < maxGen do
    Evaluation: Evaluate the individuals
    for i = 1 to |Individual| do
        if fitness_i < fitness_ind* then
            ind* ← ind_i
        end
    end
    Apply selection
    Apply crossover
    Apply mutation
    gen ← gen + 1
end
return ind*
```

2.3.1 Representation

A widely used method for representing individuals in a GP population is to use trees to represent computer programs. An example of a tree-based representation of a program is given in Figure 2.1. The leaf nodes, also called terminals, in this program are the variables x and y and constant 5.

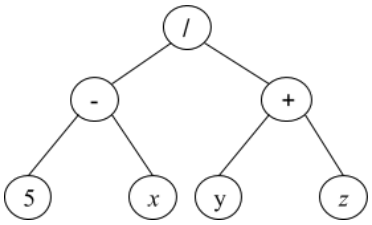


Fig. 2.1 *An example of tree-based GP program of* $(5 - x)/(y + z)$.

In this tree, the operators $\{+, -, /\}$ are the functions that operate on terminals and preside in non-leaf nodes. The set of all possible terminals and functions that can be defined for a program are called a *terminal set* and a *function* set and GP individuals are created by selecting a combination of elements from these sets. A GP individual is not required to have exactly one tree and if needed, it can contain two or more trees to facilitate representing and solving different problems. In this case, each tree can be utilised to represent and solve different aspects of a problem which will be combined in a problem-specific manner to give the final solution. Trees are not the only representation methods for GP individuals. In Linear Genetic Programming (LGP) [28], an imperative programming language like C or Java is represented in specific linear representation. In Cartesian Genetic Programming (CGP) [167, 156], similar to the tree-based representation, a graph of terminals and functions is used to represent a solution to a problem to be solved but in this case, nodes are allowed to be connected with more than one edge. In Grammar-based GP (GGP) [158] uses grammars to pose restrictions on the way that population individuals are represented.

The terminal and function sets are used to represent solutions for a problem that GP tries to solve and therefore, the correct selection of elements of these sets is crucial for success of GP. A correct selection of the sets conforms to the *sufficiency* and *closure* requirements. To be more specific, the collection of GP functions and terminals need to be comprehensive enough to be *sufficient* for solving the problem. Also, these the function set needs to be *closed* over the set of terminals and the output of the

functions so that nodes can be mixed and joined freely and arbitrarily.

2.3.2 Initialisation

The first step of a GP process, as a population-based method, is to generate an initial population of potential solutions to be evolved. Two of the most popular methods for initialising a GP population are the *full* and *grow* approaches that were proposed by Koza in the early days of GP research [130]. In both methods, the depth of GP trees are limited within a preset threshold to restrict program size and reduce the chance of creating bloated large trees that contain redundant sub-trees. In the *full* method, the depth of a tree is exactly the same as the maximum allowed depth and only functions are permitted to appear at the depth that are smaller than the maximum depth. Both terminals and functions are selected randomly. In the *grow* method, the restriction of choosing the terminals only at the maximum depth is lifted and terminals are allowed to be selected at smaller depths too. The two methods can be combined to improve diversity in the initial population. In the combined method, called *ramped half-and-half*, half of the initial population is generated with the *full* method while the rest is created with the *grow* technique.

2.3.3 Evaluation

In an evolutionary algorithm, it is important to measure how fit or capable each individual is for solving the problem since the individuals' fitness plays an important role in giving the algorithm directions for finding good programs. The definition of the fitness value for an individual is dependent on the problem to be solved but in any case, it should be a good indicator of the performance and efficiency of the program for solving the problem. For a particular problem, different fitness measures can be defined. For instance, when a classification problem is solved with GP, each individual is a classifier and the fitness can be defined as the rate of suc-

cessful predictions over the instances of a training dataset. However, this is not necessarily the only possible option and fitness can be defined differently, e.g. as the aggregate error between target class and the decision of the classifier over all instances of the training dataset. The fitness function of GP individuals play an important role in their chance of being selected for evolution, as is described in Subsection 2.3.4.

2.3.4 Selection

In an evolutionary system, fitness of population members plays as a deciding factor in its chance to be selected to generate new individuals. Consequently, the better the fitness of an individual, the more likely that the individual will be selected as a parent to produce new individuals. Roulette wheel [187], tournament selection and fitness ranking [187] are three of the most popular selection methods for GP. In the roulette wheel method, a probability distribution is constructed based on fitness value of population member so that individuals with better fitness have a higher chance of being selected and individuals are selected based on that distribution. The tournament selection method has two steps: (1) a number of individuals (determined with a tournament size parameter) are sampled randomly with uniform distribution and then (2) the individual with best fitness is selected. An advantage of this method is that bad individuals also have a chance of being selected. In fitness ranking method, the population is sorted based on fitness value and then, the individuals are selected randomly based on their rank in the sorted population.

2.3.5 Evolution

As the name suggests, evolution is the main component of an evolutionary algorithm like GP. The overall aim of this process is to improve the pool of potential solutions by inheriting and mixing genetic materials of a population and creating a new and better population. This process con-

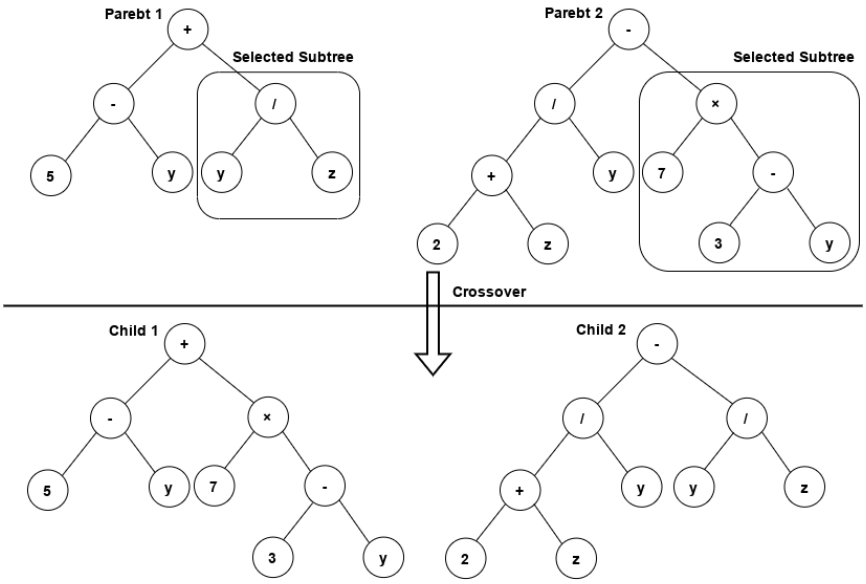


Fig. 2.2 *The crossover operator.*

sists of the operators *crossover*, *mutation* and *reproduction*. These operators are applied to GP population with a user-defined probability.

The crossover operator mimics the sexual interaction of biological entities in which two individuals are selected with a selection method to contribute genetic material for creating new individuals. *Subtree crossover* is one of the earliest and most frequently used crossover methods methods for GP [130]. In this method, first, one node in each parent is selected randomly and then, the subtrees located at the selected nodes are swapped between parents to generate two new offsprings. An example of the crossover operator, that is applied to two GP trees, is depicted in Figure 2.2. Since the majority of nodes in a GP tree are terminals, it is recommended by Koza to discriminate between the terminals and functions in selecting crossover points by selecting functions and terminals with a probability of 90% and 10% respectively [130].

Mutation is another evolutionary operator. The prevalent mutation method for GP is *subtree mutation*. In this method, an individual is se-

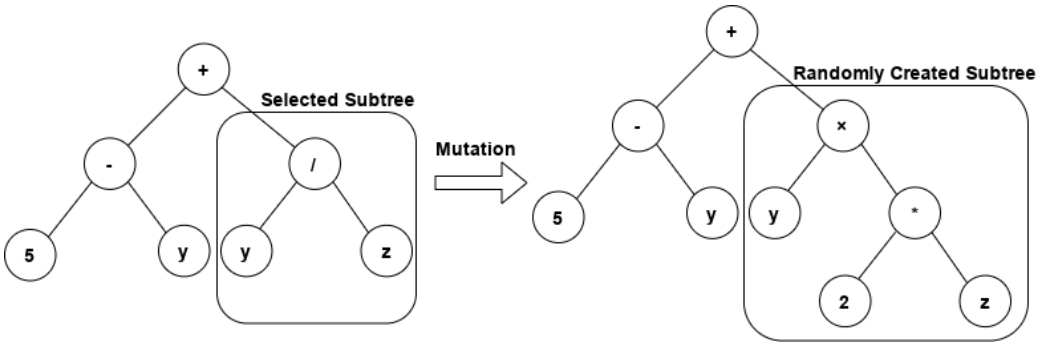


Fig. 2.3 *The mutation operator.*

lected and one of its nodes is selected randomly and the subtree rooted at the selected node is replaced with a randomly generated new tree. An example of the mutation operator, that is applied to a GP tree, is depicted in Figure 2.3.

The reproduction first selects an individual by the selection method and copies it into the next population as it is and without modifying it. It should be noted that *elitism* is a special case of reproduction that picks the best individuals in the population and inserts copies them into the next population. This mechanism allows GP to maintain good candidate solutions found so far and preserve them in the population.

2.4 Hyper-heuristics

In general, the term *heuristic* refers to any technique or approach that can be used for solving a problem. In this regard, such a technique may employ practical methods that may not be able to guarantee a perfect or optimal solution [113, 190]. Instead, the heuristic can achieve solutions that are sufficiently good in context of the problem at hand. Heuristics for solving a particular problem are often devised from previous experiences in overcoming similar problems [113], [190]. The concept of heuristics has its origin from the outside of Computer Science as the earliest contemporary

studies of the concept was based on the study of human decision-making in 1970s and 1980s by psychologists Armos Tversky and Daniel Kahneman [122].

Despite their power and applicability, heuristics have their shortcomings too. One of the major problems with heuristics is that generally, heuristics developed for a particular problem are not applicable to other problem domains. Additionally, heuristics are usually expensive to develop because they require substantial expertise and knowledge in problem domain [52].

To overcome these challenges, the concept of *hyper-heuristics* was devised. Denzinger et al. [60] are considered the first to use the term hyper-heuristics [63] to denote a system that chooses and combines from a collection of artificial intelligence methods. The term hyper-heuristic was coined first in this paper but the idea of operating on the search space of heuristics can be traced back to as early as 1960s to Fisher and Thompson [77] who demonstrated that better results can be achieved by combining scheduling rules rather than using single rules.

Cowling et. al [52] introduced *Hyper-heuristics* into the field of combinatorial optimisation [63] who believe that hyper-heuristics operate at a higher level of abstraction than heuristics. In their term, a hyper-heuristic "operates at a higher lever of abstraction than current metaheuristic approaches. The hyperheuristic manages the choice of which lower-level heuristic method should be applied at any given time, depending upon the characteristics of the region of the solution space currently under exploration ".

After defining the concept, Cowling considered a scheduling problem and defined several hyper-heuristics for solving it. By analysing the performance hyper-heuristics that they defined, the authors showed that their novelty is indeed effective and suggested hyper-heuristics may have wider applicability to other problems.

A more recent definition was provided by Burke et al. [121], [31] to

include hyper-heuristics which generate new heuristics from components of existing heuristics:

> A hyper-heuristic is a search method or learning mechanism for selecting or generating heuristics to solve computational search problems.

Burke et al. [121] outline two main categories of hyper heuristics: heuristic selection methodologies and heuristic generation methodologies. Heuristic selection methodologies select a low-level heuristic to apply at a given point in the search space. Heuristic generation methodologies automatically generate new heuristics from a set of low-level components or building blocks. In either case, the set of low-level heuristics being selected or generated can either be further split to distinguish between those which construct solutions from scratch (constructive) and those which modify an existing solution (perturbative) [185]. As well as the nature of the search space, hyper-heuristics can learn from feedback concerning heuristic performance throughout the search process. Hyper-heuristics which utilise online learning continuously adapt throughout the search process based on the feedback they receive. Hyper-heuristics using offline learning train a heuristic on a subset of instances before being applied to a larger set of unseen instances.

2.5 GPHH for UCARP

GP can be utilised as a hyper-heuristic approach to solving UCARP instances. The GPHH approach consists of a key element, a *meta-algorithm*, that is required for evaluating the heuristic functions that are GP individuals [176]. A meta-algorithm for UCARP takes a routing policy and a UCARP instance as input and returns a feasible solution for the instance. Considering the input, output and functionality of the meta-algorithm, it can be designed in various fashions. For the case of single-vehicle UCARPs,

one meta-algorithm is described in [145]. This meta-algorithm employs a constructive heuristic-like approach. The algorithm starts with an empty route and step by step, it adds tasks to the end of the route in a sequential approach. As a result, this meta-algorithm will not open any new route as long as the current route is open. The algorithm may return to depot to refill its capacity when it is fully-loaded.

The meta-algorithm initialises the vehicle's route to start from the depot. Naturally, in the beginning, all tasks are unserved and the capacity of the vehicle is intact. Then the meta-algorithm uses the routing policy to find the next task that needs to be served (or go back to the depot to replenish its capacity) until all tasks are served. To decide the next task that the vehicle should serve, first all available tasks are considered and then a subset of them is selected with a *filtering* method. If the filtered subset is empty, the vehicle needs to go back to depot to refill, close the current route and open a new one. If the subset is not empty, the routing policy is applied to the tasks to calculate a heuristic value of each task, and the task with the least heuristic value is selected to be served.

Because of the dynamic nature of the problem, it is likely that selected routes lead to failure. As it was mentioned earlier, demand of tasks have a stochastic nature and therefore, when a vehicle arrives at a task, it may find the actual demand greater than available capacity of the vehicle. This situation leads to a *route failure*. In case of a *route failure*, the vehicle returns to the depot to refill. Also, again, because of the random nature of the problem, it is likely that a vehicle arrives at an edge e to find it inaccessible (i.e. $dc(e) = \infty$) which leads to an *edge failure* during a vehicle's course of action. It is because of this nature of UCARP that traditional and exact optimisation methods that try to find robust solutions will fail to provide a solution that can address these failures effectively. Edge failures are handled by taking a detour around the failed edge. Otherwise, if no failure is occurred, the task is served and removed from the list of unserved tasks. This process continues on until all tasks are served. When

the meta-algorithm is finished, a collection of routes are created that serve all the tasks in the UCARP instance.

The described meta-algorithm is designed for the case of UCARP with one vehicle but for the case of multiple vehicles, a new meta-algorithm is described in [163]. This new algorithm is essentially a discrete event simulation system that is comprised of a priority queue of events and a system state. The system state contains the information about unserved tasks, the remaining demand fraction of tasks, the current partial routes and the served tasks. In the begining, the system is initialised with all the unserved tasks and the remaining demand fraction of each task is set to 1. Additionally, the partial routes are initialised so that all vehicles are at the depot, and the served demand of all routes are set to zero. The event queue of the meta-algorithm is a priority queue that is originally initialised with *refill* events for each route. A refill event is triggerred when a vehicle is at a node on its way back to the depot to refill. Additionally, the queue can have *serve* and *Refill-and-serve* events. A *serve* event is triggerred when a vehicle arrives at a node to serve the next task. The *Refill-and-serve* event is triggerred after a route failure, which occurs when a vehicle arrives at a node on its way back to the depot to refill, and then go back to serve the failed task. At each step of the simulation, an event is pulled out of the queue and executed, after which the state is updated. The simulation is considered finished when all the tasks are served and the vehicles are stationed back back at the depot.

By having a meta-algorithm and a training set, a typical GPHH approach to solving a UCARP instance can be described briefly as:

Step 1: Initialisation: Create an initial GP population of routing policies randomly.

Step 2: Evaluation: For each individual in the population, find its fitness value by applying the meta-algorithm to instances in the training dataset and averaging over the total cost of the obtained solutions for the training instances.

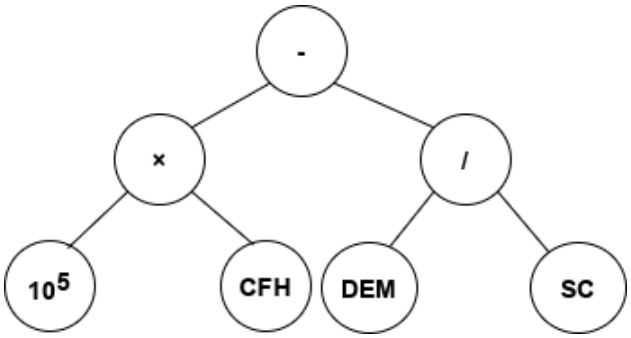


Fig. 2.4 *The tree representation of the* 10^5 * *CFH - DEM / SC routing policy*

Step 3: If the stopping criteria are met, return the routing policy that has the best fitness value. Otherwise, go to Step 4.

Step 4: Create a new population using the genetic operators of crossover and mutation. Then, go to Step 2.

Routing policies can be represented with the typical tree-based representation. With this representation, a traditional tree-based crossover and mutation can be utilised. Figure 2.4 presents the tree representation of a well-known path-scanning heuristic [125] as 10^5 * *CFH - DEM / SC*. In this policy, *CFH* indicates the cost from the vehicle's current location to the candidate task, *DEM* is the demand of the task, and *SC* is its serving cost (the coefficient 10^5 of *CFH* is a sufficiently large number which ensures that tasks with smaller *CFH* are always preferred)

Compared with the manual design of routing policies, which can be time expensive and requires high level of domain knowledge, GPHH is a very viable alternative that does not require extensive domain knowledge or any preplanned solutions, and studies have shown it to be the state-of-the-art for tackling this problem [153], [145], [163]. However, a shortcoming of this method is that, because of the nature of GP, it requires model training. The training phase, though done automatically, can be non-trivial and time consuming. On the other hand, the problem scenario can change over time in reality. In this case, the performance of the previ-

ously trained routing policies deteriorates greatly [163] and it is needed to retrain new routing policies. Intuitively, we can train new routing policies from scratch. However, due to the high computational cost of the training process, it is desirable to have methods that alleviate the training cost by reusing the knowledge gained from previous problem solving to retrain the routing policies for the new problem more efficiently. In the literature of machine learning, this scenario is a prime candidate for transfer learning [151], [186].

2.6 Transfer Learning and Optimisation

Machine Learning algorithms are some of the most widely used and successful artificial intelligence algorithms that have found applications in a wide variety of both academic and industrial problems such as regression, image processing and text mining [186, 151]. Learning is an inherent capability of human beings and is a source of inspiration and motivation for empowering machines with learning capabilities. An interesting feature of the learning ability in humans is the capability of transferring the knowledge learned in one domain to another related or similar new domains which can boost the learning efficiency in the new domains. This feature, as useful as it is, is not present in traditional machine learning techniques and consequently, they need to learn new problems from scratch even if, similar and related problems are already solved. As a result, they do not benefit from any potential boost in performance or reduction in training and computational efforts from learned knowledge in previous experiences [100, 129].

Transfer learning tries to address this shortcoming. In machine learning, transfer learning can be described as "the ability of a system to recognise and apply knowledge and skills learned in previous tasks to novel tasks" [182]. As mentioned earlier, the application of knowledge from previous tasks, among other benefits, has the potential of increasing perfor-

mance and/or reducing training costs in new problems. Additionally, it can be of great use in cases that there are no or little labelled data in target domain [182, 100]. Transfer learning algorithms have been devised for many learning methods like neural networks [173, 148], Markov logic networks [165, 164] and many learning problems such as text categorisation [97], image classification [202, 255] and web page classification [143].

The same line of reasoning that motivates devising and applying transfer learning methods, i.e. reuse of the common knowledge can improve performance and effectiveness, is also valid for other areas of artificial intelligence. Particularly, recent efforts have demonstrated that extraction and salvaging of common knowledge has a great potential for boosting the performance of evolutionary optimisation algorithms [96]. In the context of EC algorithms, the act of reusing the experience that is learned from solving a *source* problem to improve the effectiveness of solving a *target* problem is referred to as *transfer optimisation* [96]. In general, transfer optimisation for EC algorithms can lead to better initial populations, faster the convergence, and fitter final solutions [100, 218].

Due to the existence similarities between the transfer learning and transfer optimisation concepts, in this section, a definition and a brief overview of transfer learning is given first in Section 2.6.1. However, since the focus of this thesis is on transfer optimisation, this literature review will concentrate on transfer optimisation. Hence, a definition of transfer optimisation is given in Section 2.6.2 and a detailed review of the recent advancements in this research area are given in Section 2.7.3.

2.6.1 Definition of Transfer Learning

To give a formal definition of transfer learning, we need to define *domain* and *tasks*.

Definition 1. *[Domain] A domain* $D = \{\chi, P(X)\}$ *is comprised of a feature space* χ *and a marginal probability distribution* $P(X)$ *so that* $X = \{x_1, ..., x_n\} \in$

χ.

Definition 2. *[Task] A task $T = \{Y, f(.)\}$ is comprised of a label space $Y = \{y_1, ..., y_m\}$ and function $f(.)$, called an objective predictive function, which can be learned from training data $\{x_i, y_i\}$*

The objective function can be utilised to predict the label of an instance x_i as $f(x_i)$.

Definition 3. *[Transfer Learning] Let D_s and T_s be a source domain and a learning task on source domain respectively. Let D_t and T_t be a target domain and learning task on target domain respectively. Transfer learning is the effort to use the knowledge in D_s and T_s to improve or facilitate the learning of the target task, i.e. the target predictive function $f_t(.)$ given that $D_s \neq D_t$ or $T_s \neq T_t$.*

Based on this definition, the condition $D_s \neq D_t$ means that either $\chi_s \neq \chi_t$ or $P_s(X) \neq P_t(X)$. A similar meaning can be implied from the condition $T_s \neq T_t$. [151]

Types of Transfer Learning

In transfer learning, there are many types of methods for transferring knowledge from source domains to target domains [186, 235]. Categorising transfer learning techniques mainly focuses on the difference between the source domain and the target domain. In a transfer learning task, when the input features in the source and target domains are different, the task is considered as heterogeneous transfer learning; otherwise it is homogeneous transfer learning [82].

There are three important research issues in transfer learning: 1) what to transfer, 2) how to transfer, and 3) when to transfer [62, 100].

"What to transfer" asks which part of knowledge can be transferred from the source to the target problem. Some knowledge is specific for the source task and should not be re-used. Other knowledge may be common

between different domains and potentially help improve performance for the target domain [62].

After transferred knowledge has been discovered, a learning algorithm needs to be devised to extract [117] and then transfer the knowledge to the target problem, which corresponds to the issue "How to transfer"[62]. It is also in this step that should be decided how the knowledge should be used [117].

"When to transfer" asks in which situations, transferring skills will be beneficial and when knowledge should not be reused. In some situations, when the source problem and target problem are entirely different, brute-force transfer may not be successful. In the worst case, it may even deteriorate the performance of learning in the target domain. This situation is referred to as *negative transfer* [62].

After giving a brief history of transfer learning, Pan et al. [186] gave a rigorous mathematical definition of transfer learning. They then move on to categorise transfer learning into inductive transfer learning, transductive transfer learning and unsupervised transfer learning categories and provide a mathematical definition for each of the mentioned categories. The paper, based on "what to transfer", identifies four approaches to transfer learning: Instance transfer, feature-representation transfer, transfer of parameters and relational-knowledge transfer. By identifying the aforementioned approaches, the paper gives an overview of applications of each approach in each category of transfer learning [186].

Although the concept of transferring learning is an important field of research in the area of machine learning and artificial intelligence [186, 235, 151], there are key differences that prevent the applicability of the majority of these methods for UCARP. Among them, the most important factor is that transfer learning methods are generally tailored to the learning techniques that are utilised for solving a particular problem [186, 213, 142, 110, 215]. Because of this, and considering that GPHH is currently the state-of-the-art method for solving UCARP, only knowledge transfer methods that

are developed for GP and Evolutionary Computation (EC) algorithms are viable candidates for handling the changes in problem aspects of UCARP. This falls into the category of transfer optimisation algorithms.

2.6.2 Transfer Optimisation

Let's consider K optimisation problems $T_1, T_2, ..., T_K$. Each problem $T_k, k = 1, ..., K$ consists of a search space X_k and an objective function f_k and a set of inequality and equality constraints $\boldsymbol{g}_k$ and $\boldsymbol{h}_k$. Accordingly, the optimisation problem T_k can be stated as

$$\min_{\boldsymbol{x}} f_k(\boldsymbol{x}_k) \tag{2.9}$$

$$\text{subject to } g_{ki}(\boldsymbol{x}) \leq 0, \text{for } i = 1, ..., |\boldsymbol{h}_k| \tag{2.10}$$

$$\text{and, } h_{ki}(\boldsymbol{x}) = 0, \text{for } i = 1, ..., |\boldsymbol{h}_k| \tag{2.11}$$

For the sake of this thesis, we consider f_k to be scalar as the UCARP optimisation problem in this work is a single-objective problem. In these equations, $|\boldsymbol{g}_k|$ $|\boldsymbol{h}_k|$ are the number of inequality and equality constraints.

In the context of evolutionary optimisation algorithms, the output of the algorithm is the solution that the algorithm finds for the problem. Let's consider the scalar value Q quantifies the quality of the solutions. Accordingly, we denote the effectiveness of a search algorithm on problem T_k as $Q_k(T_k)$, which quantifies the quality of the solutions achieved with regard to f_k in t time steps. Particularly, if X_k^t is the set of candidate solutions that are evaluated over t time steps, then we denote the quality of an algorithm for solving a problem T_k as

$$Q_t(T_k) = f_k(\boldsymbol{x}^*) : \boldsymbol{x}^* \in X_k^t \wedge (\nexists \boldsymbol{x}^* \in X_k^t : f_k(\boldsymbol{x}) < f_k(\boldsymbol{x}^*)) \tag{2.12}$$

Let $\mathcal{M}$ denote the knowledge base of the information that was learned or extracted from different problem solving exercises. That is, if the knowledge building block m_k is extracted from solving T_k, the knowledge base can be considered as $\mathcal{M} = \cup_{\forall k} m_k$.

Definition 4. *Given a knowledge base $\mathcal{M}$ that is extracted from solving K previous problems $T_1, ..., T_K$ and a newly presented optimisation task $\mathcal{T}$, transfer optimisation is the act of incorporating $\mathcal{M}$ into the dynamics of the optimisation algorithm so that $Q_t(\mathcal{T}|\mathcal{M}) - Q_t(\mathcal{T}) \geq 0$, where $Q_t(\mathcal{T}|\mathcal{M})$ is the algorithmic quality conditioned on the knowledge embedded in $\mathcal{M}$. In this context, $T_1, ..., T_K$ are referred to as source problems and $\mathcal{T}$ is called the target problem.*

This definition basically states that *transfer optimisation* is the act of increasing the quality of an optimisation through incorporation of the knowledge $\mathcal{M}$ that is learned from solving previous problems. It should be noted that this definition does not specify any structure for the knowledge base $\mathcal{M}$ and also, it does not mention how the $\mathcal{M}$ should be extracted and reused. Additionally, the availability and incorporation of the knowledge base does not necessarily increase the quality and performance of optimisation algorithms. Conversely, if $\mathcal{M}$ extracted from the problems that are not related to $\mathcal{T}$, incorporating $\mathcal{M}$ into the search process of solving $\mathcal{T}$ may even decrease the performance of the algorithm. Similar to the case of transfer learning, this phenomenon is also referred to as *negative transfer*.

2.7 Related Work

2.7.1 Existing CARP and UCARP Methods

CARP

CARP originally originated from real-world scenarios and as a result, it has many applications in real-world situations. For example, Campbell et al. and Amponsah et al. respectively modelled street watering and snow

removal [38] and waste collection [8] as CARP and Wunderlich et al. used a modified version of CARP to model routing meter readers for the Southern California Gas Company from Los Angles, USA [239]. Accordingly, researches on CARP are helpful for improving the work efficiency in related service industries. CARP is a NP-hard problem. Hence, although there are some exact algorithms [20, 23] that can solve small-size instances of CARP, finding the optimal solution for all cases of CARP is not always possible [221]. Because of this fact, alongside exact methods, many inexact methods, ranging from heuristics [90, 196, 236] and meta-heuristics [45, 196, 208], have been devised to tackle CARP. Most of these methods show promising performance in both solution quality and running time [161], even for large-scale CARP instances [159].

UCARP

In CARP, it is assumed that all the problem parameters are static. However, this assumption does not always match with the specifications of the real-world problems [162]. For this reason, several extensions of CARP have been proposed. Stochastic CARP (SCARP), or CARP with stochastic demand (CARPSD) as is referred in some papers [46, 134], was first proposed by Fleury et al. [78, 79] and assumed that the task demands are stochastic. In a different approach, Eydi et al. [70] and Babaee et al. [216] used fuzzy numbers for modelling the uncertainty in task demands. Chen et al. [42, 41] took a different approach to handling uncertainty in which service time and travel time are considered stochastic. However, the most comprehensive approach to handling uncertainty in CARP was proposed in the uncertain CARP (UCARP) model that was proposed by Mei et al. [162] in which the demand, the travelling cost, the existence of tasks and the existence of routes were considered to be stochastic.

Solving UCARP

Generally speaking, there are two approaches to solving UCARP: proactive methods that take a robust solution and optimise it, and reactive methods that utilise Genetic Programming Hyper-Heuristic (GPHH) to search for routing policies that generate UCARP solution for the problem. The work of Wang et al. in [224] and [223] and the work of Mei et al. in [162] are interesting techniques in the category of proactive methods. In [224], Wang et al. developed a memetic algorithm based on the genetic algorithm and later, they developed new a new Estimation of Distribution Algorithm with stochastic Local Search in [223].

Although proactive approaches have been proposed for solving UCARP, they are not flexible enough for real-time and real-world scenarios [145] , mainly because the changes in the problem, such as machine breakdown, are very difficult to predict. To overcome this shortcoming, Weise et al. [234] founded the reactive approach to solving UCARP by proposing the idea of of evolving routing policies for UCARP with GP and in a simulated environment. Simulation [58] is an important approach to the study of complex problems, such as vehicle routing [170] and crane scheduling [179]. In the reactive approach, the experiments are based on the simulated model for UCARP to evaluate the effectiveness. In this approach, in order to evaluate the fitness of a potential solution, it is needed to run a simulation to assess the performance of the solution in the simulated environment. As a result, the GP training process can be computationally more expensive for solving UCARP since it requires a substantial number of priority calculations with the population members for making decisions during UCARP simulation runs. On the other hand, this approach provides greater flexibility for handling the uncertainties of the environment.

Building on the idea by Weise et al. [234], Liu et al. proposed an improved Genetic Programming-based Hyper Heuristic method for solving UCARP [145] and evaluated the performance of the proposed method on benchmark instances described in [162]. In this work, a routing policy

is utilised to model a decision making process for helping vehicles select their next tasks after serving the current one. The optimal routing policies are evolved with GP. Mei et al. extended the approach in [145] to a general scenario that can process multiple vehicles for serving tasks [163]. MacLachlan et al. took advantage of GPHH for solving a modified and improved UCARP by adding a new feature into the feature set of UCARP [153]. MacLachlan et al. [154] also showed that the performance of vehicles will increase if they are allowed to collaborate. Liu et al. also showed that [146] proposed a novel proactive-reactive approach, which represents a solution as a baseline task sequence and a recourse policy. The two components are optimised under a cooperative coevolution framework, in which the baseline task sequence is evolved by an estimation of distribution algorithm, and the recourse policy is evolved by genetic programming.

The routing policies evolved by GPHH are usually difficult to interpret. Hence, Wang et al. [226, 225] proposed ensemble approaches to evolving smaller but more interpretable routing policies. Wang et al. also proposed a set of multi-objective approaches [228, 227, 229] to training routing policies that are both fit and have small sizes. In a later work, Wang et al. [230] partitioned the GP population into distinct niches based on the fitness value, selected the smallest individual of each niche into an archive and modified the breeding operators to utilise the archive for creating smaller and more interpretable policies. A detailed survey of the various variants of CARP can be found in [144].

2.7.2 GPHH for Solving Combinatorial Optimisation Problems

Since its inception, hyper-heuristics, and particularly the automated design of heuristics, have seen considerable interest from scholars for designing heuristics for hard computational problems [176]. GP is a popular

method for automatic heuristic design that is described shortly. Burke et al. [35] outlined the suitability of genetic programming as a hyper-heuristic to generate new heuristics and survey previous work attempting to create heuristics using genetic programming. One of the advantages that Burke et al. [35] highlighted is that genetic programming relies on expert knowledge to define its terminal and function sets. As human expert knowledge is necessary, domain specific information can be incorporated into the fundamental components of the system. A second advantage is that other methods (such as genetic algorithms) may restrict the length of an encoded solution in order to facilitate simple genetic operators, genetic programming trees have variable length representation. This can be useful if the best length encoding for heuristic representation is not known. Finally, genetic programming can be used to evolve trees as executable programs allowing low-level heuristics to be generated directly.

Applications of GPHH for Solving Combinatorial Optimisation Problems

Genetic programming has successfully been used to evolve new constructive heuristics comparable to human designed heuristics for a number of problem domains. Burke et al. [34], [36] showed that stand-alone heuristics generated using genetic programming could outperform the human designed "bestfit" heuristic from the literature on unseen instances of the same class of one dimensional bin packing problems. This work was extended to three dimensional bin packing by Allen at al. [5] and generalised by Burke et al. [37] to include one, two and three dimensional bin packing problems, again obtaining human competitive results. A similar method was presented by Burke et al. [33] for two dimensional strip packing problems. Bader-El-Den and Poli [19] used genetic programming to quickly generate "disposable" heuristics to solve the satisfiability problem. Again, this work generated heuristics comparable to those which were human designed. However, only a limited search space of heuristics was cov-

ered. Kumar et al. [131] used genetic programming as a hyper-heuristic to evolve heuristics for the bi-objective 0-1 knapsack problem. This system successfully created 'reusable' heuristics able to produce a set of Pareto-optimal solutions. The Pareto fronts generated using this approach are indistinguishable from those obtained using the human-designed profit-to-weight ratio heuristic. Hauptman et al. [101] employed genetic programming to generate solvers for two common puzzles including the NP-Complete Freecell. Genetic programming has also been used as a hyper-heuristic by Keller and Poli [123] for the travelling salesman problem, by Fukunaga [83] for generating local search heuristics for satisfiability and by Geiger et al. [85] for creating dispatching rules for the job shop problem. At a higher level of abstraction, Hyde et al. [111] evolved the acceptance criteria component of a selection hyper-heuristic. The evolved acceptance criteria performed well when compared to standard acceptance criteria from the literature on instances of both bin packing and MAX-SAT. Tan et al. [212] developed a novel cooperative GPHH approach for on-line resource allocation.

Dynamic Job Shop Scheduling, DJSS, [205, 64, 168] and its variant Dynamic Flexible Job Shop Scheduling, DFJSS, [240, 244, 250, 245] are also two other difficult combinatorial problems that that GPHH excels at solving. To mention some of the most recent publications in this regard, Zhang et al. [247] proposed a new approach to improve the effectiveness of the crossover operator. In [243, 160], the authors proposed a feature selection methods for improving the performance of evolving scheduling rules for solving DFJSS. Nguyen et al. [177, 178] proposed a method for using a dimensionality reduction technique and growing neural gas to find an optimal representation of phenotypic characteristics of programs evolved by genetic programming for solving DFJSS.

2.7.3 Transfer Optimisation Methods

The literature of transfer learning in the broad context of machine learning is vast and it is not possible to effectively cover it in this text [186]. Since the focus of this thesis is solely on an evolutionary computing work, a review of EC algorithms with knowledge transfer capabilities is presented in this subsection.

Most of the early transfer optimisation efforts were focused on *injection of genetic materials* from a solved source problem into the population of the algorithm that solves a target problem. One of the earliest efforts in this direction can be attributed to Cunningham et al. [54] who proposed starting an optimisation process for a new problem with solutions that were either transferred as whole or were constructed from components of several solutions. Louis et al. [150] proposed to inject periodically into the population the transferred solutions that are most similar to the best solutions of the current population. Taylor et al. [214] performed knowledge transfer through initialising Genetic Algorithm (GA) with the final population of the source problem.

Another one of the initial works in applying concepts of knowledge transfer to an evolutionary algorithm was given by Koçer et al. [129]. The paper implements transfer learning as transferring individuals from population of the algorithm solving the source problem to the initial population of the algorithm solving the target problem. In their implementation, the authors selected two sample individuals I_1 and I_2 at each iteration of the source problem, one of which is the best individual in the population and the other is the median individual in terms of fitness, and saved into an "ndividual pool". For initialising the population of the target problem, 30% of the initial population of the target problem were chosen randomly from this pool [129].

Dinh et al. [62] present three new algorithms for knowledge transfer in GP. In *FullTree* method, k% best of individuals in the final generation source problem are used as initial individuals. In their second algorithm,

named *SubTree*, they investigate the idea that there is a good subtree structure that is shared between related problems. In their implementation of this idea, the authors focused on last generation of source problem. To transfer, a random subtree of each individual of final population is selected and added to a pool and then moved to the target problem as initial individuals. The third method, *BestGen*, it is assumed that good knowledge to transfer may appear, not only in the final generation but also during the evolutionary process of source problem. Consequently, in each generation of GP on source problem, k best individuals are sampled into a pool that are finally used as initial population on target. . Al-helali et al. [4] proposed a multi-tree genetic programming method with new genetic operators based on transferred knowledge and utilised it for solving symbolic regression problems with incomplete data. Muñoz et al. [171] have given a comprehensive review of transfer optimisation methods for GP in the context of constructive induction.

Haslam et al. proposed an update to Dinh's work in [62] by adjusting the parameter k adaptively. For this purpose, they defined two methods that they collectively called *adaptive-k*. In their first method, called *k-throttle*, first trees transferred from source domain are stored in a separate population and their performance on target domain is measured. The same number of trees are generated randomly and their performance is also measured on target problem and k is calculated with

$$k = \frac{P_r}{P_s + P_r} \tag{2.13}$$

wherein P_r is average fitness of 50% of individuals in the store that have a performance on target domain better than other individuals which were generated randomly and P_s has the same meaning but for the transferred population. After computing this parameter, $k\%$ of the worst-performing randomly generated initial population will be substituted with $k\%$ of best-performing transferred individuals and form initial population of GP on new problem. In their second approach, called *Tournament Transfer*, Haslam

et al. make the same assumption as before but combine the random and transferred initial population and $k\%$ of individuals are selected with tournament selection [100].

Subtrees of GP individuals contain (partial) domain knowledge about a problem and if a subtree is repeated among individuals, it has the potential of containing useful and transferable knowledge. This idea was explored by O'Neil et. al. In their method, named common subtrees from related problems (CSRP), they first discovered common subtrees among individuals and then, considered them as functions. These functions replace leaf terminals in the extracted common subtree with variables and are used to augment the function set of GP on new target problem [182].

Iqbal et al. modified the initialisation and mutation phase of GP to utilise transfer learning. The authors considered children of the root node of each individual in the population as a transferable segment of knowledge and called them "code fragment". On source domain, average fitness of the population in the final generation is calculated and then code fragments are extracted from individuals that are better than the average. These code fragments are then stored to be reused on target problem. On the target problem, for initialisation, children of a root node are either selected from the stored code fragments with a probability of 0.5 or from terminal or function set of GP. For mutation, a subtree of individual to be mutated will be selected and and replaced with a tree that is created the same way described for initialisation [119].

Considering n source domains, the Fu et al. first train GP populations to learn knowledge from the source domains. To use the learned knowledge on a target domain, for each source domain i, p individuals are selected randomly from the final population of the trained GP. For each instance in the target domain, each of the selected individuals are used on the instance to find its class. Output of the individuals are then summed and if the sum is greater than $0.5 * p * n$ it is classified as class 1 else it is classified as class 0 [82].

Iqbal and et al introduced the concept of code blocks in [114]. A code block is a tree of depth two or less. At first, a random population of code fragments is created. This population is kept static during the learning process and are chosen from during the covering phase of classifiers. The authors defined a fitness measure for code fragments and CFs with better fitness on smaller problems are utilised as terminals to seed the population of code fragments in a more difficult problem. In this work, a terminal is either a condition bit or a code fragment from a previous level code fragment with a probability of 0.5 [114].

Inspired by the concept of code fragments, the Iqbal et al. used a novel design of Learning Classifier Systems for solving large scale Boolean problems in which the population of code fragments is not static. By solving simple Boolean problems, they extracted code fragments of depth two from the conditions of final population from individuals whose fitness value is greater than the average fitness and reused them in solving higher level and more complex problems. The extracted code fragments are used as terminals in rule conditions for solving higher-level problem. A terminal in a higher-level problem is a code fragment with a probability of 0.5. The authors have limited the depth of code fragments to two [115].

Iqbal and co-authors introduced the concept of ADFs, instead of ternary conditions, into XCS [118]. In their work, ADFs are manually designed and are not extracted automatically from any problem domain and, all condition bits are filled with ADFs. Alvarez et al advanced this novelty. Built furthermore on a later work of Iqbal et al [115], Alvarez et al used the idea of code fragments in the condition section of rule-sets. However, in addition to this feature, the paper considers rule sets from source problem and allows them to act as functions in target domain that can appear at non-leaf nodes of conditions [6].

Use of code fragments can increase the scalability of system to some degree by reducing problem search space. However, CFs impose a strain on computing resources because as the learned knowledge increases, the long

chain of CFs increase which eventually prohibits them from scaling up to larger and more complex problems. The authors observe that CFs are susceptible to redundancy as different CF trees can have the same value and hypothesized that this downside to efficiency can be alleviated by using compacting techniques on the final code fragment rules to remove redundancy. Consequently, the authors built on their work in [6] and devised a compaction scheme to reduce redundancy in CFs and improve their scalability [7].

In a more recent paper, Iqbal considered the direct children of the root node of individuals in the final generation of a GP run and named them as 'code fragments'. When solving a new problem, these code fragments are used as the transferable knowledge and utilised in the initialisation and mutation phase of GP. In initialisation phase, each child of the root node is either generated randomly or selected from code fragments with a probability of 0.5. In the mutation phase, a subtree of an individual is selected and replaced with either a randomly generated tree or a code fragment selected with the same probability [117]. In a later work, Igbal used the same technique but with probabilities μ_I and μ_M for initialisation and mutation respectively for image classification [116].

Based on the messy coding representation of classifier conditions in LCSs proposed by Lanzi [43], the authors use the hash table data structure to implement Lanzo's sensory tags and extended Iqbal's [115], [114] framework with this innovation.

Transfer optimisation can also be achieved through learning a model of the good solutions of the source problem. In these *model-based* approaches, the model is then used for biasing the search process when solving the target problem [55]. Feng et al. first established that there is a common knowledge between CVRP and CARP and hence, it would be more efficient to transfer knowledge learned from one domain to the other. After this, the paper proposes a common representation for CVRP and CARP. For a given instance of CARP, I_a, first the shortest distance matrix, SD,

among all the arcs of I_a is calculated with Dijkstra's algorithm. Then, MDS is applied on SD and finally, a common problem representation is reached by applying manifold alignment between CVRP and the MDS approximated CARP (Algorithm 1 in the paper). In the paper a knowledge meme in evolutionary optimisation serves as an instruction to guide the search toward the near-optimal solution. The paper proceeds with which it extracts meme knowledge from a solved instance of a problem. For a given problem instance p and its solution s^* on the constructed common feature space, a knowledge meme M is formulated as maximisation of statistical dependency between p and s^* with distance as constraints. For each solved instance of a CARP or CVRP one knowledge meme can be found and the paper uses a weighted sum of multiple memes from similar problems as a method of selecting memes. The knowledge meme selection process is formulated as to identify the weight of each knowledge meme. Subsequently, the knowledge meme M_t generalised from past experiences is then assimilated for enhancing evolutionary search on another problem domain via the generation of meme-biased solutions [73].

Expanding on his previous work [73], Feng et al. proposed a new approach for transferring knowledge that consisted of four operators: Learning, Selection, Variation and Imitation. Defining meme as a distance matrix that maximally aligns a problem instance to its solution, the Learning operator is modelled as finding this mathematical mapping between a solved problem and its solution by maximising their statistical dependence. The Selection operator, which aims to find a high-quality meme, is modelled as finding a weight coefficient for each meme in a pool of memes. The variation operator, that is meant to introduce new knowledge into the pool, is implemented as weighted sum of memes. The imitation operator projects the variated meme into the search space of new problem and is implemented as product of new problem instance to L which is derived by singular value decomposition of the variated meme. The result of these operators will be used to initialise an evolutionary algorithm

that finds the final solution for the new problem. In this approach, if the knowledge pool is empty or the selection operator does not find a meme that is similar to a new problem, an ordinary initialisation procedure of evolutionary algorithm will be used to initialise population [74]. Feng et al. [71] also learned customer representation as transferable knowledge for solving VRPs.

Feng et al. performed knowledge transfer through a single layer denoising autoencoder. A similar idea proved to be useful for knowledge sharing in multi-task scenarios [76]. Zhou et al. [256] captured the structured knowledge from optimised vehicle routing problem (VRP) and reused it for biasing the search when the problem changes.

Probabilistic models are also popular for modelling the knowledge and transferring it between problems. For example, it is shown that previously learned/evolved probabilistic models can be used as transferable knowledge. In order to achieve this, a similarity metric can be designed on problem variables to indicate the strength of dependencies between them. The metric can then be used for collecting statistics on the transferred models and biasing the search process when similar problems are solved in future [193, 192, 103, 102].

2.7.4 Multi-Task Learning for Evolutionary Algorithms

Transfer of useful knowledge is not specific to transfer optimisation but is also an important concept in the context of Evolutionary Multi-Task (EMT) learning [94, 183, 95, 22]. The main idea in multi-task learning is to detect and extract the common knowledge available in related tasks and through sharing them between the evolutionary processes, improve the performance of the search for solving each task [183]. Accordingly, one of the main differences between mutli-task learning and transfer optimisation is that in transfer optimisation, the source problem is solved first and the act of knowledge extraction and searching for the solution of the tar-

get problem starts after finishing solving the source problem. On the other hand, in multi-task learning algorithms, no task is solved before hand and all tasks are solved simultaneously and the act of knowledge sharing takes place during the search for the solving the problems [96].

Although the idea of learning several tasks together has long history in machine learning [39, 252, 53], the notion has gained attraction from researchers recently [96]. In the context of EMT, existing algorithms can be broadly categorised into two classes of implicit and explicit EMT methods [72]. The algorithms in the implicit EMT class solve multiple tasks via utilising a single population with a unified representation of solutions of all tasks. The act of knowledge transfer in this category of EMT methods is achieved *implicitly* and through the crossover operator between two individuals that have two different skill factors, that is, they are more suited to solve two different tasks.

The multi-factorial evolutionary algorithm, MFEA, by Gupta et al. [95] and its multi-objective version by Feng et al. [72] are two prime examples of implicit EMT algorithms. Ding et al. [61] indicate that MFEA is not very effective when the global optima of tasks are separate. Additionally, the efficiency decreases further when decision variables of the tasks have different dimensions. To address the first issue, the paper proposes a decision variable transformation strategy that transforms the decision variable into a unified space in which all tasks have the same global optimum. Bali et al. [21] noted that if the tasks are uncorrelated, MFEA is vulnerable to negative transfer of knowledge and they proposed a linearised domain adaptation to handle this issue. In a later work, Bali et al. [22] proved a set of mathematical theorems that provided the conditions on which an EMT algorithm can converge to a solution. Liaw et al. [138] improved the performance of MFEA for solving many tasks, i.e. more than two or three tasks. Zhou et al. [257] performed knowledge transfer that is self-adaptive based on the information obtained during the search process. Feng et al. [75] also proposed a novel EMT algorithm to solve a generalised VRP with

occasional drivers. Zhong et al. [254] proposed a way that it encoded GP individuals so that a unified space is defined so that individuals from multiple problem domains can be evolved in a single population. Zheng et al. [253] proposed an extension to the MFEA method in which a way of measuring the similarity of tasks is provided that allows knowledge transfer based on the degree of similarity. Wei et al. [233] proposed an EMT-based classification method using Gene Expression Programming with several knowledge transfer strategies between tasks. Zhang et al. [242, 241] proposed a set of novel approaches for implicit knowledge sharing for solving DFJSS.

In contrast to the implicit approach, the explicit category of EMT methods dedicate one population for each optimisation task to be solved. In this paradigm, the act of knowledge sharing between populations is performed *explicitly* by specific operators that are designed for this purpose. The explicit approach to EMT optimisation has the advantage over the implicit methods that it allowing incorporating multiple search mechanisms for solving different optimisation tasks, which could improve the performance of EMT algorithms because different search mechanisms can have different problem-specific operators and biases [72].

Feng et al. [76] proposed an EMT algorithm for solving continuous problems that utilised different search mechanisms. Since each search paradigm of their proposed algorithm could potentially utilise different representations, the authors employed denoising autoencoders for converting the candidate individuals from the representation of their source population to the representation of the receiving population. Realising that autoencoders are designed for continuous problems, Feng et al. [72] proposed a novel method for mapping the representation of the discrete problems such as CARP into a continuous representation that allowed proposing an explicit EMT that utilised autoencoders during knowledge transfer for solving CARP. Gong et al. [92] argued that in many multi-task learning scenarios, some of the tasks are easier and require less computa-

tional resources. As a result, better results can be achieved if less resources are allocated to easier tasks and more to harder ones and therefore, the authors proposed a method for allocating computational resources dynamically based on the difficulty of tasks. Lin et al. [140] proposed an explicit EMT method for multi-objective optimisation in which if a transferred solution from one population to another is non-dominated in the receiving population, then the neighbours of this solution are selected as the transferred solution in the next generation. Noting that finding useful knowledge for transfer between populations is of crucial importance, Lin et al. [141] proposed utilising a Naïve Bayes binary classifier to predict if a candidate solution is likely to be helpful for the receiving population. In a work by Zhang et al. [248], a surrogate model is trained to estimate the fitness of transfer candidates in their receiving populations and based the selection of candidates on the estimated value.

2.7.5 Surrogate Models for Evolutionary Algorithms

For most EC systems, the most expensive operation during the evolution is fitness evaluation of individuals. Despite the fact that fitness value is the guiding principle of EC systems, because of the evaluation expense, it is not possible to use fitness evaluation liberally. In order to overcome this, surrogate models [120, 180, 219, 30, 107, 149, 203] are proposed that can reduce the cost of fitness evaluation at the cost of fitness accuracy and since they are cheaper, they make it possible to take advantage of fitness guidance more frequently. Different surrogate models can be considered, e.g. binary model that either accepts or rejects an individual as a good or bad one, the models that give a ranking of the population members based on their quality without producing their exact fitness value or, fitness estimation models that give an estimate of individual fitness. Some of the common algorithms for surrogate models include kriging models [48] and radial basis function networks [189].

Although there exist a plethora of surrogate techniques for EC methods [217, 120], the majority of these methods are not applicable for solving UCARP because, the training instances of UCARP are inherently different from conventional machine learning tasks such as regression [126] and numerical optimisation [139, 184]. In UCARP, the fitness evaluation is based on simulations that are run on datasets that are generated dynamically while in traditional machine learning tasks, the training dataset are available in advance. Theoretically, it is feasible to collect some data during simulation processes and adopt machine learning tools to learn the relationship between the collected information and the fitness value. However, it is not trivial to select what information to collect.

To the best of our knowledge, there exist no research about surrogate-assisted GP for solving UCARP. However, the similarity between the nature of UCARP routing policies and DFJSS routing and sequencing rules allows utilising some of the surrogate techniques that have been proposed for DFJSS. In this regard, the way that decision situations are collected is the minimal needed change to the original algorithm. Particularly, Hildebrandt et al. [105] proposed a KNN-based approach [200, 27] to estimating the fitness of an individual. In their approach, the authors saved two most recent GP populations as pool of data. Then, to estimate the fitness of an individual ρ, its similarity to the individual in the pool are measured. Finally, the average fitness of k individuals in the pool is considered to be the approximate fitness of ρ. Accordingly, in this approach, the important consideration is about how to measure the similarity of two individuals. In this context, the genotypic similarity is a straightforward approach to measuring the similarity of two individuals ρ_1 and ρ_2, which is the approach of measuring how much genetic materials ρ_1 and ρ_2 share. However, Hildebrandt et al. [105] demonstrated that in case of a using GPHH for solving an uncertain problem like DFJSS, the genotypic similarity measure is not effective at capturing the actual similarity of the ρ_1 and ρ_2. Instead, the author devised a phenotypic approach for measuring how similar ρ_1

and ρ_2 are. The phenotypic approach measures how similar the individual behave in a given decision situation. For example, In the context of DFJSS, a sequencing situation is a situation in which a machine is idle and a sequencing rule such as ρ_1 needs to calculate the priority value of each operation in the work queue of the machine to select the operation with the highest priority to complete next. In this sense, two routing rules are identical in a given situation if they both make the same decision for that situation. Hildebrandt et al. [105] showed that the phenotypic similarity based on comparing how similar the rules behave in set of decision situations is very good at measuring how similar the rules actually are. Based on this description, it can be seen that there exist a great degree of resemblance between the sequencing rules of DFJSS and the routing policies of UCARP as they both calculate priority of unserved operations/tasks for machines/vehicles. This indicates that the same approach can be utilised for measuring the phenotypic similarity of UCARP routing policies.

The work by Hildebrandt et al. [105] is based on the KNN technique that requires a similarity measure. Nguyen et al. [180] proposed a surrogate model based on using a simplified model of the DFJSS problem that does not require a similarity measure. Similar approaches were proposed in [250, 127]. Zhang et al. [246] also proposed a multi-fidelity approach based on the idea of incorporating multiple surrogate models based on simplified DFJSS simulations with different degree of accuracy.

2.8 Summary

In this chapter, the basic concepts of arc routing, evolutionary algorithms, genetic programming, heuristics and hyper-heuristics are introduced, which form the underlying foundations of this thesis. This chapter investigated the details of subject problem, i.e. UCARP and explained how routing heuristics are employed for constructing problem solutions. In doing so, the proactive and reactive categories methods are explained as the main

class of approaches for solving UCARP, and the most recent efforts in each category are reviews. Furthermore, the reactive approach of using GP as a Hyper Heuristic, GPHH, for evolving vehicle routing policies is described. Accordingly, the GPHH approach and some of its applications are also explained.

After studying the literature of UCARP, it becomes clear that the GPHH approach is the state of the art for solving UCARP. Nevertheless, this approach suffer from an important shortcoming which is the reusability issue. More specifically, the routing policies that are evolved with GPHH are optimal for the solved UCARP instance that they were trained for as long as the problem does not change. However, many real-world applications of UCARP are subject to change frequently, e.g. number of vehicles, graph topology or probability distribution of stochastic variables. When such changes happen, the routing policies lose their optimal performance and hence, new routing policies are needed to be retrained. Considering the fact that training of routing policies is time-consuming and expensive, there is a need for effective and efficient retraining of these policies. Considering the fact that in many real-world scenarios, the change in problem instance creates new problems that are related to the old one, transfer learning is the ideal technique for handling this issue.

Therefore, we reviewed the concept of transfer learning in machine learning and its counterpart in evolutionary algorithms which is called transfer optimisation. Through our review, we realised that the body of existing transfer optimisation methods for GP is not dense and also, none of the existing methods are designed for the case of evolving hyper-heuristics. Since the act of knowledge extraction in transfer optimisation methods shares similarities with the same operations in multi-task learning, a broad review of these algorithms are provided in this chapter too.

The content of this chapter gives an overview of the basic concepts needed for reading this thesis. The following four contribution chapters will provide the details of the challenges in performing transfer optimi-

sation for solving UCARP and the proposed approaches to overcoming them. More specifically,

- We will investigate the applicability of transfer optimisation for solving UCARP to identify (and proposing solutions for) some of the shortcomings of the existing methods. Through extensive experimental studies we will discover the lack of diversity in the knowledge source as one of the main challenges that may arise when applying transfer applying transfer optimisation to solving UCARP.
- We will propose a set of algorithms for initialising GP through transfer optimisation so that the issue of lack of diversity in the knowledge source will not affect the quality of transfer.
- We will capture the phenotypic behaviour of routing policies as transferable knowledge and utilise it during the search process for solving target problems.
- We will propose an effective method for adapting the transferred knowledge to the target problem and use it for preventing GP from losing its population diversity.

Chapter 3

Transfer Optimisation for Solving UCARP

As was mentioned in Chapter 1, the problem changes of UCARP lead to new but similar problems. Due to the existence of similarities between the old and new problems, transfer optimisation is a viable approach for solving the new problem more efficiently with the common knowledge that is extracted from the old problem. In this chapter, we will investigate the applicability of the existing transfer optimisation methods for solving UCARP, and identify some of the shortcomings in the existing literature and the challenges that may exist in the way of applying transfer optimisation methods successfully for solving UCARP.

3.1 Introduction

As was mentioned in Chapter 1, the problem changes of UCARP, which is a common phenomenon in the real-world, lead to new, but related problems. An interesting feature of the routing policy approach is that a trained routing policy is not tied to the characteristics of the problem that it is trained for. That is, even if some change happens to the characteristics of the problem, it is still possible to use the policy for the new problem. How-

ever, although the new problem has similarities with the old problem it is originated from, Mei et al. [163] showed that the old solution will not perform optimally for the new problem, even if all aspects of the old and new problems are the same except that the number of vehicles is changed by one and, even for small problem changes, it is needed to train a new routing policy. Consequently, it is needed to find a new solution for the new problem. Since the old and new problems are related, it is possible to extract the knowledge common to both problems from the solved problem to improve the search performance for solving the new problem, which motivates the application of transfer optimisation algorithms.

As the investigation of the literature in Chapter 2 pointed out, to the best of our knowledge, there is no previous work on utilising and/or devising transfer optimisation algorithms for solving UCARP. Although there exist a number of algorithms that perform transfer optimisation specifically for GP, to the best of our knowledge, no GP transfer optimisation method has been proposed or verified in the context of utilising GP as a hyper-heuristic for solving UCARP. Additionally, although there exist a plethora of algorithms for GP and GA, it is not clear how these algorithms will perform for solving UCARP because most, if not all, of the existing algorithms were devised for static problems. As a result, the focus of this chapter will be placed on investigating the applicability of the existing transfer optimisation methods for solving UCARP. By doing so, we aim to identify the challenges that may exist on the way of the successful extraction and reuse of knowledge for solving UCARP. Accordingly, the main research questions and goals to be answered through this investigation are presented in Section 3.2.

As the literature review in Chapter 2 indicated, a commonly considered approach for performing transfer optimisation for GP is the approach of selecting/extracting the (sub-)trees of individuals that were found for solving a source problem and reuse them for solving a target problem. Upon investigating the examined algorithms, we discovered a few poten-

tial shortcomings in these methods. Therefore, we also propose our solutions to these shortcoming in this chapter and investigate their potential for solving UCARP too.

To be more specific, in the context of (sub-)tree-based transfer optimisation, the majority of works rely on the fitness of individuals as the only criterion for selecting extracting sub-trees from individuals and do not consider any other indicators for the goodness of the sub-trees to be selected. Additionally, any potential approaches for transferring knowledge for GP, other than the (sub-)tree-based approach have also been neglected. Therefore, we will also propose the probability distribution of high-quality source solutions as the transferable knowledge which allows creating arbitrary number of good individuals that can be reused for solving the target problem.

3.2 Chapter Goals

The goal of this chapter is to *investigate the potential of the existing transfer optimisation algorithms for solving UCARP. In this direction, we also aim to discover some of the shortcomings of the existing algorithms, propose our solutions to them and evaluate their performance for solving UCARP too*. For this purpose, we evaluate the performance of the proposed algorithms in comparison against the existing methods and also the vanilla GPHH without any knowledge transfer. More specifically, in this chapter we have the following research objectives:

- Investigate the applicability and potential of the existing transfer optimisation algorithms to handling problem changes in UCARP.
- Develop multiple criteria for assessing the transferability sub-trees from source solutions.
- Develop a probabilistic approach to transfer optimisation in which the probability distribution of high-quality source individuals is learned.

- Analyse the efficacy of the proposed algorithms on different transfer scenarios.

3.3 Chapter Organisation

The rest of this chapter is organised as follows. Detailed descriptions of the proposed algorithm are given in Section 3.4. The experiment design is shown in Section 3.5, followed by results and discussions in Sections 3.5.3 – 3.5.7. Finally, Section 3.6 concludes this chapter.

3.4 Proposed Algorithms

In this section, three transfer optimisation algorithms are described in details. Two of the proposed algorithms are based on the detection and transfer of useful (sub-)trees and are presented in Sections 3.4.1 and 3.4.2. In Section 3.4.3, a probabilistic approach to knowledge transfer is proposed in which the distribution of the good source GP individuals is learned as the transferable knowledge.

3.4.1 Frequent Sub-Trees as Transferable Knowledge

As was reviewed in Chapter 2, a large body of work has been dedicated to implementing transfer optimisation for GP as the transfer of subtrees. However, in the majority of these works, the selection of sub-trees is based of the fitness of their individuals but they do not measure how important or useful a sub-tree could be for solving the source problem. Consequently, in this section, we also propose a set of guidelines for selecting the sub-trees that play an important role for solving the source problem.

It is reasonable to speculate that if an individual has a good fitness for source problem, it tends to be a good candidate for extracting good subtrees to be transferred. Additionally, if a subtree is repeated frequently

in good solutions that were found for the source problem, it must contain important genetic materials that GP has reused it frequently. Therefore, it is logical to speculate that such sub-trees could also be more likely to be important for solving the target problem. For this reason, we consider the final GP population during the training process of the source problem, and extract subtrees from the top individuals in terms of their test performance using the following three criteria for measuring how frequently they were reused in the top individuals:

1. **All**: All the possible subtrees of an individual (dubbed as *FreqSub-all*).

2. **Root Subtree**: Immediate children of the root of the considered individuals (dubbed as *FreqSub-sub*).

3. **Entire**: The entire tree is transferred (dubbed as *FreqSub-root*).

After all subtrees are extracted from the best individuals of the final source population, they are sorted based on the frequency that they were repeated in the top individuals and then, *all* the most frequent subtrees were transferred as individuals until 100% of the initial population of GP on target problem is filled.

Furthermore, our preliminary studies show that the size of transferred subtrees is an important factor, that is, large subtrees usually had a better fitness for both source and target problems but also they tended to afflict GP with code bloats and also, reduce population diversity. On the other hand, small subtrees were also not beneficial as they usually had lower fitness values and disappeared in early generations. Therefore, we consider a limit on minimum and maximum size of subtrees and only transfer the ones that are within the size limit.

It should be noted that our *FreqSub* algorithm has some similarities with the work by O'Neill et al. [182]. In that work, sub-trees are extracted from the middle of their tree and do not include the terminals and then,

they are converted to functions that augment the GP function set for solving the target problem. Although the authors confirmed the effectiveness of their approach on some non-dynamic problems, in our experiments, we noticed that this approach, in the context of UCARP, will lead to hidden code bloat and unacceptable computational cost, because such a function node in a GP tree is indeed a subtree of its own, disguised as single functional node that when is needed to be evaluated, the whole subtree needs to be evaluated. It is can be rather trivial to optimize this procedure for static problems but doing so for a dynamic problem like UCARP can be more challenging. Consequently, we decided to include the terminals of the frequent subtrees.

3.4.2 Contributive Sub-Trees as Transferable knowledge

In Section 3.4.1, we considered the frequency by which sub-trees appear in the good solutions of the source problem to measure how important the sub-trees are for transfer. One downfall to that approach is that if the source individuals contain introns [47, 152, 166, 237], that is they may contain sub-trees that do not make any contribution to the fitness of their tree and hence, are redundant.

To handle this issue, we propose a new method for filtering good transferable sub-trees by evaluating subtrees to distinguish their potential for transfer. In our approach, we focus on the final source GP population as the knowledge source because the final population contains the most evolved set of individuals. Furthermore, it is natural to assume that individuals with good fitness value are better sources for knowledge extraction. Therefore, we consider the subtrees of the top 50% individuals in the final source population in terms of their test performance.

To extract the sub-trees and form the pool of the candidate transferrable sub-trees, we adopt the following two strategies:

1. **All**: All the possible sub-trees of the considered source individuals

are extracted and included in the pool.

2. **Root Subtrees**: The immediate root sub-trees of the considered individuals are included in the pool.

Needless to say, the first pool is more comprehensive and contains the second one. However, the second pool can contain potentially larger subtrees that may contain more genetic materials.

Not all the sub-trees in the extracted pool are important and they may have different reusability values. As was discussed in Section 3.4.1, frequency is an intuitive measure because a sub-tree that has appeared more frequently in the GP population may contain important building blocks that GP had reused it multiple times. Thereore, it is very likely that those building blocks are also important for solving a related target problem.

However, there is one obvious pitfall to the frequency measure that can make the measure misleading which is the potential presence of introns in the final GP trees; that is, the trees may contain redundant branches and these redundant branches increase the frequency rate of the sub-trees unnecessarily. The contribution measure that is proposed in this section can address this issue.

In the literature of genetic programming, it has been discussed that feature selection can improve performance of GP because, in a GP system, it is very likely that some features are redundant and do not contribute to the performance of the system. There are also cases that features have negative contribution to the fitness. It is for this reason that feature selection and feature weighting methods are shown to improve the performance of genetic programming systems [160, 44]. With the same line of thought, we can similarly suspect that considering all possible subtrees in a tree to be candidates for transfer may contain redundant and even harmful structures that their transfer may degrade the performance of knowledge transfer. Therefore, to detect the useful sub-trees that are potentially beneficial for transfer, we propose a second reusability measure: contribution

of the sub-tree to the fitness of its individual [160].

Given a GP tree x, we define the contribution $\xi(x, \tau)$ of its sub-tree τ as the change that happens to the fitness of the tree if the sub-tree is replaced with a constant 1:

$$\xi(x, \tau) = fit(x|\tau = 1) - fit(x). \tag{3.1}$$

After measuring the contributions of a sub-tree to each individual that contains it, a weighted cumulative contribution value can be calculated for each sub-tree with the following equation:

$$w(\tau) = \sum_{x \in \Omega} \xi(x, \tau) pow(x). \tag{3.2}$$

In this equation, Ω represents the set of all the individuals that include subtree τ, and $pow(x)$ is the power of individual x, which is the normalised fitness of individual x. The design of Equation 3.2 allows good subtrees of good individuals to have better weights. The power of an individual is defined as:

$$pow(x) = \frac{g(x) - g_{min}}{g_{max} - g_{min}} \tag{3.3}$$

in which $g(x)$ is calculated as below:

$$g(x) = \frac{1}{1 + fit(x)} \tag{3.4}$$

wherein $fit(x)$ is the fitness of individual x and Ψ is the set of all individuals that were evaluated for the source problem (note that $\Omega \subset \Psi$) and

$$g_{min} = min\{g(x)|x \in \Psi\} \tag{3.5}$$

$$g_{max} = max\{g(x)|x \in \Psi\} \tag{3.6}$$

For solving the target problem, the subtrees in the pool ("All" or "RootSubTree") are first sorted by the cumulative contribution value, and the top subtrees are selected and imported to the initial population to form 50% of the population. The corresponding algorithms are respectively named (1) *ContribSub-all* and (2) *ContribSub-subtree*.

3.4.3 Probabilistic Prototype Trees as Transferable Knowledge

In Sections 3.4.1 and 3.4.2, we proposed two new methods for the transfer of reusable trees and sub-trees. It can be argued that the subtrees that are extracted from the individuals of the knowledge source may not have the comprehensive information about the search space of the source problem and they just capture partial manifestation of that information. Although full trees have more genetic materials than subtrees, it can still be argued that good full trees are just good points in the search space of the source problem and therefore, direct transfer of them might not transfer enough helpful knowledge for solving the target problem. In this section, we investigate this argument. For this purpose, we consider the underlying probability distribution of good source individuals as the transferrable knowledge. By possessing a the probability distribution, it is possible to sample new individuals from the distribution and reuse the resultant individuals for solving the target problem. One benefit of this approach is that it may lead to the creation of individuals that were not present in the original knowledge source nevertheless, and by doing so, they can perform well for the target problem.

To capture the probability distribution of the good source individuals, we define that the probability distribution of the good source individuals should specify the probability that a GP terminal or function may occur at a given node of GP programs. Therefore, by specifying an arbitrary indexing I over nodes of GP trees, we let the random variable X_i be the GP terminal or function in node $i \in I$. Therefore, $X_i \in T \cup F$ in which T and F are the GP terminal and function sets respectively. We denote the probability that X_i has the value $r \in T \cup F$ as $P(X_i = r) = p_{i,r}$. The given definition of X_i has the categorical distribution since it can have $\rho = |T \cup F|$ exclusive outcomes and the probability of each outcome does not change over time in a standard GP [172]. Accordingly, if we let $Y_{i,r}$ be the random

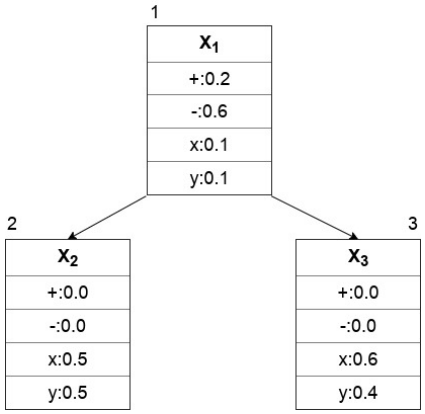


Fig. 3.1 *An example of a PPT*

variable of the number of times that $r \in T \cup F$ appeared in node $i \in I$ in a GP population, we note that $Y_{i,r}$ has a multinomial distribution [81].

With the given definition of the X_i and $Y_{i,r}$ random variables, we define a Probabilistic Prototype Tree, PPT, as a complete binary (or q-ary with q being the maximum arity of GP functions) tree in which each node $i \in I$ holds the probability vector $P_i = (p_{i,r})_{r \in T \cup F}$. Each component of the vector P_i presents the probability of selecting an item r from either the terminal or the function set and represents the underlying categorical distribution of the item r appearing at i. As an example, a simple PPT of depth 2 is shown in Figure 3.1 for $T = \{x, y\}$ and $F = \{+, -\}$.

By defining the transferrable knowledge as PPT, the next step in performing transfer optimisation is to learn a PPT from a population of source GP individuals. Since the random variable $Y_{i,r}$, that specifies the number of times that item r appeared in location i, follows the multinomial distribution, we can approaximate the value of $p_{i,r}$ based on the relationship that exists between X_i and Y_i and with the application of maximum likelihood estimation over the existing population [68]. Consequently, it is straightforward to show that $p_{i,r} \approx \overline{p_{i,r}}$ where $\overline{p_{i,r}}$ is the average number of times that the item r has appeared in location i in the population.

It should be noted that the PPT structure has some similarities with

the PIPE tree that is presented in [206] by Saustowicz et al.. However, our work does not consider the constants $R_{d,w}$ in the PPT that Saustowicz et al. use. Additionally, we defines a more rigorous modelling of the underlying probability distribution of the GP individuals that is absent in that work. Furthermore, that work takes a completely different learning approach from the one that we propose. Finally, that work is not in the context of GP transfer learning.

3.5 Experimental Studies

To evaluate the effectiveness of the proposed algorithms, a wide range of experimental source and target problems settings are designed. Table 3.1 presents the source and target UCARP instances that are used in the experiments. These UCARP instances are the extensions of the static CARP instances in which the deterministic CARP variables are converted into random ones with normal distribution. These datasets are based on real-world road network from Lancashire, UK [154]. This fact clearly highlights the real importance of transfer optimization, and hence the proposed algorithms, in the real-world. For the sake of consistency, all the experiments in the rest of this thesis will be performed on the scenarios in Table 3.1. These instances have been previously experimented on in previous studies [145, 163]. In Table 3.1, the datasets whose name ends with the *dm1* (*dm2*) suffix, e.g. Uval5Adm1, highlights that the mean of the random demands were increased by 1 (2) from the original dataset. In the table, each row is called a *transfer scenario* which is comprised of a *source instance*, that is assumed to be already solved, and a *target* instance that is needed to be solved with the knowledge transferred from the source instance. The column "Sim." (value between -1 and 1) indicates the degree of similarity of the source and target instances. To measure the similarity, 1024 unique routing policies were generated randomly, and evaluated on the source and target instances. Accordingly, we define the similarity

between the source and target instances as the Kendall's rank correlation coefficient [124] between the fitness of the 1024 random routing policies on the source and target instances. Naturally, the closer similarity value is to 1, the more similar and related are the source and target instances. It should be pointed out that all UCARP instances have the same objective, i.e. minimise the total cost of serving all tasks and hence common desirable routing decisions such as selecting the closest task have better prefences in the decision making process of the vehicles. Therefore, in practice, most UCARP instances are likely to be related to each other. As a result, having negative or very close to zero similarity values are unlikely and hence, the scenarios in Table 3.1 show decent similarity values of at least 0.46.

As given in Table 3.1, the experiments include source and target instances with varying degree of similarities (from 0.46 to 0.99). Note that we have included a large set of source and target instances in which most of scenarios show decent similarities with each other. A similarity of 0.46 is already a weak relatedness between different UCARP instances. In this section, first the parameter settings for the experiments are given in Section 3.5.1. Throughout this thesis, these settings will also be used for all the other experiments in the later chapters to maintain consistency between results. In Section 3.5.2 some of the state-of-the-art transfer optimisation methods for GP and GA are selected. These algorithms are utilised for solving UCARP and their performance is compared against GPHH without any knowledge transfer. The main purpose of Section 3.5.2 is to gain an insight on the potential of existing methods for solving UCARP and identify the best available algorithms against which our proposed algorithms will be compared. After selecting the best existing methods, their performance will be compared against the performance of FreqSub, PPTGP and ContribSub in Sections 3.5.4, 3.5.5 and 3.5.6 respectively.

Table 3.1: The transfer scenarios used in the experiments

Scn.	Source Instance	Target Instance	Sim.
1	Uval4A with 2 vehicles	Ugdb17 with 3 vehicles	0.46
2	Ugdb17 with 5 vehicles	Ugdb12 with 5 vehicles	0.46
3	Ugdb17 with 5 vehicles	Ugdb12 with 7 vehicles	0.48
4	Ugdb17 with 5 vehicles	Ugdb12 with 8 vehicles	0.49
5	Uval6B with 5 vehicles	Ugdb15 with 3 vehicles	0.56
6	Ugdb17 with 5 vehicles	Ugdb11 with 3 vehicles	0.57
7	Ugdb17 with 5 vehicles	Ugdb11 with 4 vehicles	0.59
8	Ugdb17 with 5 vehicles	Ugdb11 with 5 vehicles	0.61
9	Uegl-e1-C with 8 vehicles	Ugdb13 with 5 vehicles	0.65
10	Uval4A with 2 vehicles	Ugdb6 with 5 vehicles	0.65
11	Uval4A with 2 vehicles	Ugdb6 with 4 vehicles	0.66
12	Ugdb23 with 10 vehicles	Ugdb12 with 5 vehicles	0.67
13	Ugdb23 with 10 vehicles	Ugdb12 with 7 vehicles	0.69
14	Ugdb23 with 10 vehicles	Ugdb12 with 8 vehicles	0.7
15	Uval6B with 5 vehicles	Ugdb6 with 5 vehicles	0.7
16	Ugdb21 with 5 vehicles	Ugdb5 with 4 vehicles	0.76
17	Ugdb4 with 4 vehicles	Ugdb4 with 2 vehicles	0.89
18	Ugdb7 with 5 vehicles	Ugdb1 with 6 vehicles	0.9
19	Ugdb4 with 4 vehicles	Ugdb4 with 6 vehicles	0.9
20	Ugdb3 with 5 vehicles	Ugdb3 with 7 vehicles	0.91
21	Ugdb4 with 4 vehicles	Ugdb4 with 3 vehicles	0.91
22	Ugdb1 with 5 vehicles	Ugdb1 with 3 vehicles	0.92
23	Ugdb6 with 5 vehicles	Ugdb7 with 6 vehicles	0.92
24	Ugdb3 with 5 vehicles	Ugdb3 with 3 vehicles	0.92
25	Ugdb7 with 5 vehicles	Ugdb7 with 7 vehicles	0.92
26	Ugdb7 with 5 vehicles	Ugdb7 with 3 vehicles	0.93
27	Ugdb6 with 5 vehicles	Ugdb6 with 3 vehicles	0.93
28	Ugdb1 with 5 vehicles	Ugdb1 with 7 vehicles	0.93
29	Ugdb1 with 5 vehicles	Ugdb2 with 7 vehicles	0.93
30	Ugdb2 with 6 vehicles	Ugdb6 with 6 vehicles	0.94
31	Ugdb3 with 5 vehicles	Ugdb3 with 6 vehicles	0.94
32	Ugdb5 with 6 vehicles	Ugdb5 with 4 vehicles	0.94
33	Ugdb5 with 6 vehicles	Ugdb5 with 8 vehicles	0.94
34	Ugdb2 with 6 vehicles	Ugdb2 with 4 vehicles	0.94

Continuation of Table 3.1

Scn.	Source Instance	Target Instance	Sim.
35	Ugdb6 with 5 vehicles	Ugdb6 with 7 vehicles	0.94
36	Ugdb4 with 4 vehicles	Ugdb4 with 5 vehicles	0.94
37	Ugdb21 with 6 vehicles	Ugdb21 with 4 vehicles	0.95
38	Ugdb7 with 5 vehicles	Ugdb7 with 4 vehicles	0.95
39	Ugdb2 with 6 vehicles	Ugdb2 with 8 vehicles	0.95
40	Ugdb6 with 5 vehicles	Ugdb6 with 4 vehicles	0.95
41	Ugdb6 with 5 vehicles	Ugdb6 with 6 vehicles	0.96
42	Ugdb21 with 6 vehicles	Ugdb21 with 5 vehicles	0.97
43	Ugdb11dm2 with 5 vehicles	Ugdb11dm1 with 5 vehicles	0.98
44	Uval8A with 3 vehicles	Uval8Adm1 with 3 vehicles	0.99
45	Uval5Adm1 with 3 vehicles	Uval5Adm2 with 3 vehicles	0.99

3.5.1 Parameter Settings

The set of GP terminals are given in Table 3.2 and Table 3.3 presents the GP parameters for solving UCARP. The settings in Table 3.3 are commonly used in the UCARP literature [145, 163], which makes this study consistent and comparable with them. In our function set, the division operator is protected and returns 1 if its denominator is zero.

In our experiments, we assume that for each transfer scenario in Table 3.1, first the source instance is solved with the vanilla GPHH without using any form of knowledge transfer. Based on the GP settings in Table 3.3, this will result in 50 sets of GP individuals, each of which contains 1024 individuals of the population in each generation. These populations are considered the knowledge source that each examined transfer optimisation algorithm will utilise this knowledge source based on its configurations to train routing policies for the corresponding target instance. For fitness evaluation during the training, 5 samples are taken are rotated at each generation to reduce the change of overfitting [145]. To test the trained policies, 500 samples are used. Each algorithm is run 30 times indepen-

Table 3.2: The GP terminal set for solving UCARP.

Terminal	Description
CFH	Cost From Here
CFR1	Cost From the closest alternative Route
CR	Cost to Refill
CTD	Cost To Depot
CTT1	Cost To the closest Task
DEM	DEMand
DEM1	DEMand of the closest unserved task
FRT	Fraction of Remaining Tasks
FUT	Fraction of Unassigned Tasks
FULL	FULLness (vehicle load over capacity)
RQ	Remaining Capacity
RQ1	Remaining Capacity of closest alternative route
SC	Serving Cost
ERC	Ephemeral Random Constant number
DC	Deadheading Cost

Table 3.3: The GP parameter settings.

Parameter	Value	Function	Description
Population	1024	$+$	Addition
Crossover rate	80%	$-$	Subtraction
Mutation rate	15%	$*$	Multiplication
Reproduction rate	5%	$/$	Protected Division
Number of generations	50	$\min$	Minimum of two arguments
Number of Elitists	10	$\max$	Maximum of two arguments
Max depth	8		

dently. To compare the algorithms, the Friedman's test with a significance level of $\alpha = 0.05$ is utilised in all experiments.

3.5.2 Compared Algorithms

To assess the performance of the proposed algorithms better, we select a set of existing transfer optimisation methods from the literature and compare the performance of the proposed algorithms against them. When selecting the existing algorithms for comparing against the proposed algorithms, our options were constrained by the fact that the literature of transfer optimisation algorithms for GP is still an emerging field of research. Although transfer optimisation for evolutionary algorithms is a broad and rich area of research, the number of available methods for GP is unfortunately very limited. On the other hand, although there exist a plethora of existing transfer optimisation methods for algorithms other than GP, the majority of these algorithms are developed for specific problems or particular type of evolutionary algorithm such as GA. Adapting these existing algorithms to the context of GP for UCARP is not usually straightforward. Accordingly, the selected algorithms are BestGen [62], FullTree [62], GATL [129], SubTree [62] and TLGP [116]. Table 3.4 presents the (sub-)tree selection mechanism of these algorithms.

3.5.3 Effectiveness of Existing Algorithms

The mean and standard deviation of 30 independent runs of the compared existing algorithms from Table 3.4 are given Table 3.5. In this table, the best average performance is marked in boldface. A cursory look at the table indicate that, although some algorithms were able to perform generally better than GPHH in some scenarios, none of the them were able to improve the performance of GPHH in the majority of scenarios. To investigate deeper, we consider the Friedman test.

Table 3.4: The compared GP with knowledge transfer.

Algorithm	**Knowledge transfer mechanism**
GATL [129]	Select the best and median trees of each generation into a pool. Choose randomly from the pool to initialise the target GP population.
BestGen-k [62]	Select k of the best individuals of each generation into a pool. Choose randomly from the pool to initialise the target GP population.
FullTree [62]	Select the individuals of the final population into a pool. Choose randomly from the pool to initialise the target GP population.
SubTree [62]	Select a random subtree of each individual of final population into a pool. Choose randomly from the pool to initialise the target GP population.
TLGPC [116]	Select random subtrees of the better-than-average final individuals into a pool. During initialisation and mutation, create a root or subtree randomly or select randomly from the pool.

Table 3.5: Test performance of 30 independent runs of the compared algorithms (mean $\pm$ std)

Scn.	GPHH	GATL [129]	BestGen-1 [62]	BestGen-2 [62]	FullTree [62]	SubTree [62]	TLGPC [116]
1	91.2±0.1	91.2±0.1	**91.2±0.1**	91.2±0.1	91.2±0.1	91.2±0.1	91.2±0.1
2	551.0±10.3	**550.8±8.1**	551.1±10.4	553.7±8.8	553.8±10.3	555.5±9.1	552.7±9.0
3	**598.6±8.8**	599.6±9.5	600.5±11.6	603.6±12.6	602.9±8.5	604.1±13.9	603.5±10.3
4	639.5±11.3	**636.0±10.7**	640.6±12.1	640.6±13.7	647.4±12.6	646.7±16.4	638.5±13.1
5	58.2±0.1	58.2±0.1	58.3±0.1	**58.2±0.1**	58.3±0.1	58.2±0.1	58.3±0.1
6	424.8±8.5	424.9±8.8	423.9±8.6	424.4±8.1	423.8±8.1	**421.8±7.0**	426.8±9.1
7	432.1±7.1	**430.0±6.3**	431.2±6.3	431.3±6.5	431.0±6.7	430.8±6.8	432.9±7.2
8	432.6±5.5	**430.6±6.7**	432.7±4.8	432.2±6.1	432.3±5.1	431.6±6.5	432.9±4.4
9	576.2±3.9	**575.8±4.2**	576.8±3.7	576.9±3.8	576.6±6.5	577.0±3.8	575.9±4.1
10	340.5±4.7	338.1±4.2	**337.5±3.1**	338.5±2.1	338.2±3.6	340.0±4.7	338.7±3.1
11	347.2±6.1	347.1±6.0	**345.9±4.8**	345.9±5.3	348.0±4.9	347.4±6.0	349.3±6.4

Continuation of Table 3.5

Scn.	GPHH	GATL [129]	BestGen-1 [62]	BestGen-2 [62]	FullTree [62]	SubTree [62]	TLGPC [116]
12	**551.0±10.3**	553.7±10.5	551.8±10.1	554.5±10.3	559.1±11.7	554.1±10.2	554.4±11.4
13	598.6±8.8	598.5±7.5	597.6±8.2	**597.2±7.6**	604.0±11.5	603.8±13.3	608.7±11.2
14	639.5±11.3	640.1±12.2	639.0±11.9	**639.0±9.5**	645.7±17.4	640.1±12.6	640.0±11.5
15	340.5±4.7	339.8±3.5	339.9±5.0	339.4±3.9	**338.2±3.6**	340.4±3.9	340.4±5.1
16	444.4±4.7	445.0±7.6	**443.9±6.5**	445.5±5.2	449.2±9.3	446.0±6.7	445.6±6.4
17	324.3±6.2	323.3±5.2	**321.5±5.4**	324.0±4.3	325.1±5.6	325.0±7.8	325.0±5.2
18	360.3±3.1	359.7±3.7	359.4±3.9	359.2±4.6	**359.2±4.2**	361.1±3.7	360.9±3.0
19	358.3±2.6	**358.3±3.1**	358.8±2.7	360.1±6.9	361.0±6.4	358.6±3.8	358.5±3.7
20	359.0±1.8	**358.5±1.8**	358.6±1.9	358.7±1.8	359.1±1.9	359.4±1.8	359.2±1.6
21	340.8±4.4	340.8±2.2	**339.4±4.7**	339.7±3.8	340.0±4.3	342.0±3.2	341.0±3.2
22	351.9±3.5	**351.6±2.5**	352.5±3.5	352.2±4.4	351.9±3.4	353.2±3.7	352.1±3.8
23	356.6±1.6	356.4±1.5	**356.1±1.3**	356.6±1.6	356.6±1.6	356.7±1.7	356.7±1.7
24	310.9±0.5	311.0±0.3	310.7±0.8	310.9±0.6	**310.4±1.0**	311.0±0.6	310.7±2.3
25	389.2±0.2	**389.1±0.2**	389.2±0.2	389.2±0.2	389.2±0.1	389.1±0.2	389.1±0.2
26	363.1±2.8	**363.1±3.2**	363.4±2.6	363.6±2.7	363.3±3.1	363.7±2.5	363.3±4.0
27	342.1±6.2	342.5±7.8	**340.9±8.0**	341.5±5.8	342.4±7.1	342.4±5.9	343.9±6.8
28	382.0±5.5	**381.0±4.6**	381.3±8.0	384.4±5.6	386.8±6.4	383.3±6.9	381.2±7.5
29	382.8±3.3	383.9±2.6	**382.7±4.8**	384.1±5.2	387.7±6.4	385.1±4.9	383.7±6.4
30	351.5±2.5	**351.1±2.2**	351.7±1.2	351.4±2.1	351.4±1.4	352.1±2.0	351.5±1.3
31	326.0±4.7	325.2±4.0	325.2±5.0	**323.6±6.2**	325.6±4.7	326.8±4.3	326.2±4.8
32	444.4±4.7	443.7±5.6	**442.0±7.3**	442.6±7.6	444.3±8.3	445.7±8.6	446.2±7.6
33	448.2±0.5	449.0±2.3	**448.2±0.9**	448.4±1.0	450.7±6.9	449.0±2.2	448.8±1.5
34	**384.6±4.4**	387.1±6.0	386.9±5.0	386.7±5.1	387.0±4.4	388.2±5.4	386.2±5.2
35	369.3±1.8	**369.3±2.2**	369.8±3.8	369.8±2.3	370.2±4.0	369.3±2.3	369.5±2.7
36	**321.4±5.2**	322.7±4.2	323.8±5.1	323.0±3.8	323.5±4.8	324.4±5.2	325.2±5.7
37	166.2±2.0	166.1±1.7	**165.2±1.5**	165.5±1.6	165.7±2.1	165.8±1.8	165.8±2.2
38	**376.1±7.6**	381.2±7.0	377.8±7.5	378.8±7.9	380.8±6.7	382.3±8.8	378.7±5.6
39	415.7±9.2	415.5±7.2	**414.3±8.9**	415.0±5.2	417.8±6.5	418.8±8.5	417.2±6.3
40	347.2±6.1	347.5±5.2	**345.8±4.4**	347.6±4.9	349.7±10.1	349.2±7.0	350.3±11.5
41	351.5±2.5	**351.4±2.7**	352.0±2.3	351.8±1.7	352.0±1.8	352.6±2.5	351.9±3.9
42	165.9±1.8	165.6±1.7	165.7±1.6	165.7±1.5	**165.2±1.4**	165.4±1.4	165.7±1.7
43	462.6±6.0	**455.6±8.9**	460.2±5.4	458.9±6.7	460.5±4.8	460.2±7.1	461.3±6.9
44	**426.6±3.3**	427.3±1.6	427.0±2.6	426.8±2.6	427.8±1.6	427.2±2.0	428.3±0.4

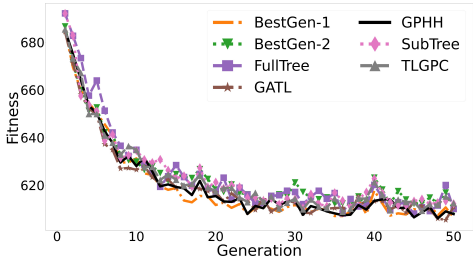
Continuation of Table 3.5

Scn.	GPHH	GATL [129]	BestGen-1 [62]	BestGen-2 [62]	FullTree [62]	SubTree [62]	TLGPC [116]
45	499.0±3.9	496.3±-1.0	497.6±4.4	497.1±5.3	498.7±4.9	499.9±5.5	**496.1±-1.0**
Rank	3.57	2.92	3.06	3.49	4.81	5.34	4.81
Friedman's p-value							**1.1e-16**

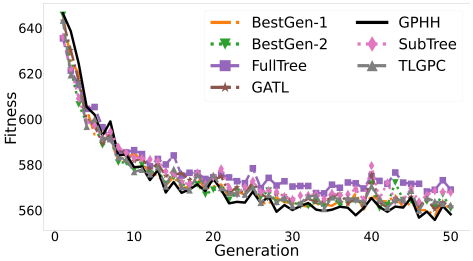
The last two rows of the table show the p-value of the test and the rank that the test gives to each algorithm. The p-value indicate that there is a significant difference between the results. To pinpoint the difference, the Conover post-hoc test [49] is conducted on the results and the obtained p-values are presented in Table 3.6 after being adjusted with the Benjamini-Hochberg method [25]. Investigating Table 3.6, we notice that none of BestGen-1, BestGen-2 and GATL are significantly different from GPHH. On the othe hand, the difference between GPHH and FullTree, SubTree and TLGPC is significant. Considering the rank of the algorithms in Table 3.5, we note that the act of knowledge transfer has decreased the performance of GPHH significantly, which indicates a negative transfer of knowledge. This effect can also be seen in the convergence curve of the algorithms.

Table 3.6: Adjusted p-value of the pairwise comparison of existing algorithms

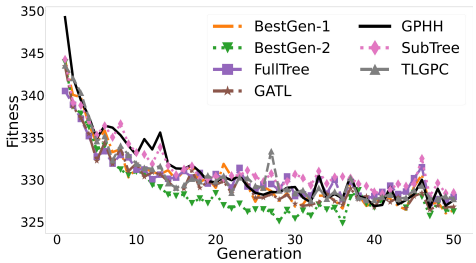
	GATL	BestGen-1	BestGen-2	FullTree	SubTree	TLGPC
GPHH	0.24	0.30	0.92	**0.01**	**0**	**0.01**
GATL	–	0.89	0.28	**0**	**0**	**0**
BestGen-1	–	–	0.33	**0**	**0**	**0**
BestGen-2	–	–	–	**0.01**	**0**	**0.01**
FullTree	–	–	–	–	0.33	0.92
SubTree	–	–	–	–	–	0.30



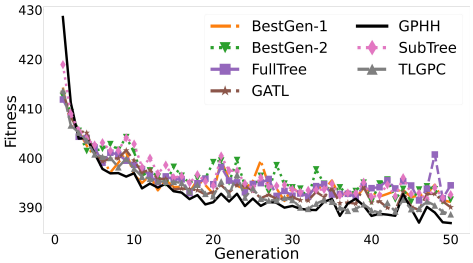
(a) *From Ugdb17 with 5 vehicles to Ugdb12 with 7 vehicles (Scn. 3)*



(b) *From Ugdb23 with 10 vehicles to Ugdb12 with 5 vehicles (Scn. 12)*



(c) *From Ugdb3 with 5 vehicles to Ugdb3 with 6 vehicles (Scn. 31)*

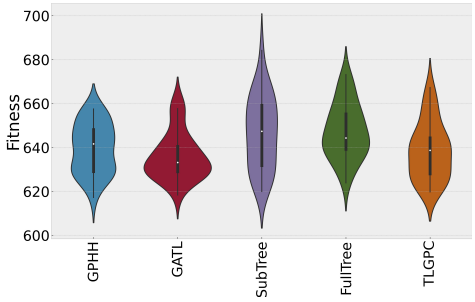


(d) *From Ugdb2 with 6 vehicles to Ugdb2 with 4 vehicles (Scn. 34)*

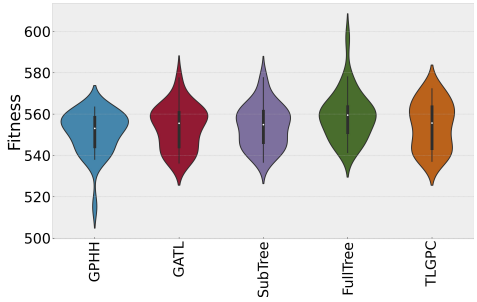
Fig. 3.2 *Convergence curve of GPHH and some existing knowledge transfer methods.*

Fig. 3.2 shows the convergence curves of the average test performance of the evolved routing policies on the test scenarios across the 30 independent runs. The figure clearly shows the negative effect of the transferred knowledge. From the figures, we can see that in some scenarios, some of the algorithms were able to give GP a better initial state (e.g. FullTree in Fig. 3.2d). However, this superiority did not last for very long and after a few generations, their performance was similar to GPHH or even worse.

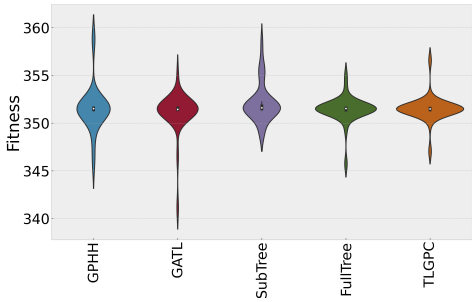
Figure 3.3 presents the distribution of the solutions obtained with each of the existing algorithms, in the form of violin plots. These plots also confirm the earlier observations. As is evident, the distributions of all algorithms are very similar to each other and GPHH and in some cases,



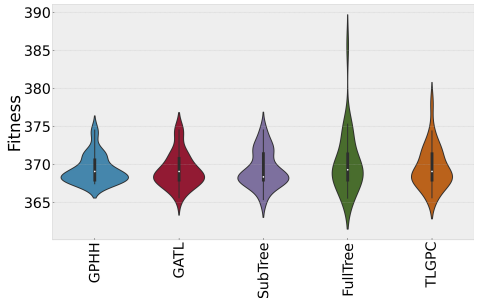
(a) *From Ugdb17 with 5 vehicles to Ugdb12 with 8 vehicles (Scn. 4)*



(b) *From Ugdb23 with 10 vehicles to Ugdb12 with 5 vehicles (Scn. 12)*



(c) *From Ugdb3 with 5 vehicles to Ugdb3 with 6 vehicles (Scn. 30)*



(d) *From Ugdb2 with 6 vehicles to Ugdb2 with 4 vehicles (Scn. 35)*

Fig. 3.3 *The distribution of solutions found with GPHH and some existing knowledge transfer methods.*

e.g. FullTree in Fig. 3.3b, the performance is even worse than the case of GP without knowledge transfer. The violin plots of other scenarios also showed similar patterns.

Overall, the results in Tables 3.5–3.6 and figures 3.2–3.3 indicate that existing transfer optimisation algorithms are not very effective for handling scenario changes of UCARP. In Sections 3.5.4–3.5.6 we will investigate if it is possible to improve the quality of the knowledge transfer through introducing more informed methods for selecting the transferable (sub-)trees that were introduced in Subsections 3.4.1–3.4.3.

3.5.4 Effectiveness of Frequent Sub-Tree Transfer

As was discussed in Section 3.4.1, the sub-trees that are frequently recreated by GP during solving the source problem may contain important genetic materials that GP reused them often. In this section, we investigate the performance the proposed three FreqSub methods in Section 3.4.1. Although all existing algorithms showed similar performances to GPHH without knowledge transfer, GATL and BestGen-1 showed slightly better performances in terms of Friedman rank and hence, they are selected from the body of existing algorithms and its performance is compared against FreqSub.

Table 3.7 presents the performance of the compared algorithms in terms of the average total cost of 30 independent runs.

Table 3.7: Test performance of 30 independent runs of the FreqSub algorithm (mean ± std)

Scn.	GPHH	GATL [129]	BestGen-1 [62]	FreqSub-All	FreqSub-Root	FreqSub-Subtree
1	91.2±0.1	91.2±0.1	**91.2±0.1**	91.2±0.1	91.2±0.1	91.2±0.1
2	551.0±10.3	**550.8±8.1**	551.1±10.4	554.5±9.6	552.0±10.9	552.1±10.4
3	**598.6±8.8**	599.6±9.5	600.5±11.6	604.6±10.5	602.5±8.1	604.8±11.3
4	639.5±11.3	**636.0±10.7**	640.6±12.1	647.3±15.1	651.2±15.6	647.9±12.9
5	58.2±0.1	58.2±0.1	58.3±0.1	58.3±0.1	**58.2±0.2**	58.2±0.1
6	424.8±8.5	424.9±8.8	423.9±8.6	**422.5±8.8**	425.3±8.2	425.6±9.4
7	432.1±7.1	**430.0±6.3**	431.2±6.3	433.6±6.6	432.9±6.9	431.4±7.6
8	432.6±5.5	**430.6±6.7**	432.7±4.8	433.1±5.4	432.4±4.5	433.2±5.7
9	576.2±3.9	**575.8±4.2**	576.8±3.7	577.5±5.0	576.4±4.0	576.4±4.1
10	340.5±4.7	338.1±4.2	**337.5±3.1**	339.8±4.3	338.3±3.4	339.1±4.6
11	347.2±6.1	347.1±6.0	**345.9±4.8**	348.3±6.1	349.2±7.6	348.6±6.1
12	**551.0±10.3**	553.7±10.5	551.8±10.1	556.3±11.2	554.3±12.0	553.2±10.0
13	598.6±8.8	598.5±7.5	**597.6±8.2**	603.2±14.8	606.7±15.4	604.6±11.4
14	639.5±11.3	640.1±12.2	**639.0±11.9**	642.3±11.1	640.1±12.6	642.3±13.4
15	340.5±4.7	339.8±3.5	339.9±5.0	**338.9±4.9**	339.8±4.8	340.4±4.0
16	444.4±4.7	445.0±7.6	**443.9±6.5**	448.2±6.8	448.3±7.8	447.5±7.4

Continuation of Table 3.7

Scn.	GPHH	GATL [129]	BestGen-1 [62]	FreqSub-All	FreqSub-Root	FreqSub-Subtree
17	324.3±6.2	323.3±5.2	**321.5±5.4**	324.1±5.0	324.6±5.7	325.9±5.6
18	360.3±3.1	359.7±3.7	359.4±3.9	360.4±4.6	359.9±3.6	**359.4±4.3**
19	358.3±2.6	**358.3±3.1**	358.8±2.7	361.2±5.6	363.2±7.6	360.1±4.3
20	359.0±1.8	**358.5±1.8**	358.6±1.9	359.2±1.6	359.3±2.0	359.3±1.7
21	340.8±4.4	340.8±2.2	**339.4±4.7**	339.8±4.6	340.0±3.8	342.0±3.2
22	351.9±3.5	**351.6±2.5**	352.5±3.5	352.7±3.8	352.9±4.2	353.4±4.4
23	356.6±1.6	356.4±1.5	**356.1±1.3**	356.5±1.6	356.5±1.6	356.7±1.7
24	310.9±0.5	311.0±0.3	310.7±0.8	310.8±0.8	**310.0±2.7**	310.7±0.9
25	389.2±0.2	**389.1±0.2**	389.2±0.2	389.2±0.2	389.2±0.2	389.1±0.2
26	363.1±2.8	363.1±3.2	363.4±2.6	**362.3±3.6**	363.2±2.7	363.9±1.7
27	342.1±6.2	342.5±7.8	**340.9±8.0**	343.0±6.4	343.3±9.0	342.6±6.3
28	382.0±5.5	**381.0±4.6**	381.3±8.0	385.0±5.7	385.1±6.4	386.1±7.0
29	382.8±3.3	383.9±2.6	**382.7±4.8**	383.1±4.4	384.8±6.2	385.1±5.4
30	351.5±2.5	351.1±2.2	351.7±1.2	351.5±1.1	**350.5±2.8**	352.0±1.6
31	326.0±4.7	325.2±4.0	**325.2±5.0**	326.6±5.1	325.6±4.1	327.4±3.5
32	444.4±4.7	443.7±5.6	**442.0±7.3**	445.0±9.3	446.9±8.2	442.8±7.4
33	448.2±0.5	449.0±2.3	**448.2±0.9**	449.8±3.4	450.3±3.4	449.7±2.8
34	**384.6±4.4**	387.1±6.0	386.9±5.0	387.4±6.2	388.6±5.0	387.1±5.6
35	369.3±1.8	**369.3±2.2**	369.8±3.8	369.5±2.2	370.7±3.6	370.2±4.1
36	**321.4±5.2**	322.7±4.2	323.8±5.1	324.0±4.6	325.3±6.1	325.3±6.4
37	166.2±2.0	166.1±1.7	**165.2±1.5**	166.1±1.9	166.2±2.1	165.7±1.5
38	**376.1±7.6**	381.2±7.0	377.8±7.5	380.8±6.7	380.1±7.2	380.6±7.8
39	415.7±9.2	415.5±7.2	**414.3±8.9**	417.2±7.5	417.0±6.9	415.9±10.7
40	347.2±6.1	347.5±5.2	**345.8±4.4**	352.1±7.3	348.0±5.8	349.7±7.6
41	351.5±2.5	**351.4±2.7**	352.0±2.3	351.7±1.4	352.3±3.2	352.4±3.7
42	165.9±1.8	165.6±1.7	165.7±1.6	**165.6±1.2**	165.9±1.3	166.0±1.6
43	462.6±6.0	**457.4±7.0**	460.2±5.4	461.5±5.2	458.3±7.3	462.7±4.9
44	**426.6±3.3**	427.3±2.0	427.0±2.6	427.4±2.1	427.7±2.5	427.1±1.4
45	499.0±3.9	498.5±4.4	**497.6±4.4**	498.2±4.6	499.2±4.8	499.5±4.8
Rank	2.92	2.43	2.34	4.23	4.46	4.41
Friedman's p-value					**4.14e-33**	

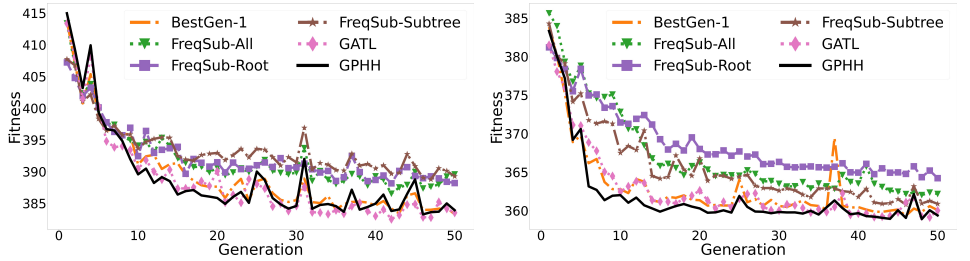
As is given in Table 3.7, the Friedman test does not rank any of the FreqSub algorithms better than GPHH, BestGen-1 or GATL. The p-value of the Friedman test indicate that there exist significant differences in the results. Table 3.8 present the p-value of the paired-wise comparison of the algorithms, adjusted by the Benjamini-Hochberg method [25].

One evident observation from Table 3.8 is that the difference between the performances of FreqSub methods and GPHH is significant. Since the test ranked FreqSub worse than GPHH in Table 3.7, it can be concluded that FreqSub performed significantly worse than GPHH. Another observation in Table 3.8 is the performance of GATL and BestGen-1 is significantly better than FreqSub. This is surprising in the sense that FreqSub selects the sub-trees from high-quality individuals. On the other hand, these methods collect individuals from every source generation and as a result, most of the individuals they collect will not have a high quality. But nevertheless, these algorithms perform better than FreqSub. This observation will be investigated further in Section 3.5.7. Another interesting point to note is that there is no significant difference between all three variants of FreqSub.

The convergence curves of the algorithms in Figure 3.4 also depict these observations clearly. The violin plots of the solutions found with algorithm are given in Fig. 3.5, which also confirm the ineffectiveness of the algorithms. The convergence plots of the algorithms in most of the

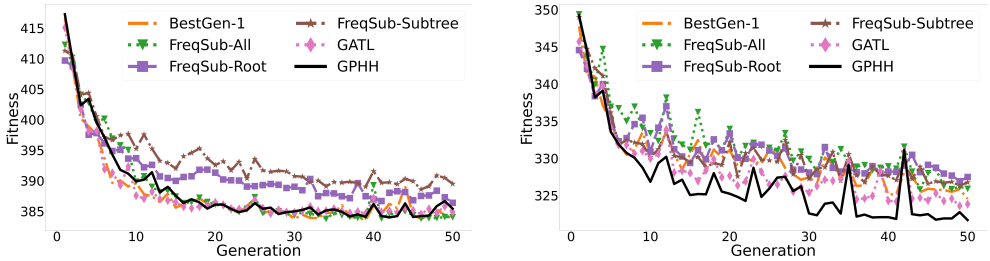
Table 3.8: Adjusted p-value of the pairwise comparison of FreqSub algorithm

	GATL	BestGen-1	FreqSub-All	FreqSub-Root	FreqSub-Subtree
GPHH	0.78	0.61	**0.01**	**0**	**0**
GATL	–	0.78	**0**	**0**	**0**
BestGen-1	–	–	**0**	**0**	**0**
FreqSub-All	–	–	–	0.78	0.78
FreqSub-Root	–	–	–	–	0.78



(a) *From Ugdb1 with 5 vehicles to Ugdb1 with 7 vehicles (Scn. 28)*

(b) *From Ugdb4 with 4 vehicles to Ugdb4 with 6 vehicles (Scn. 19)*

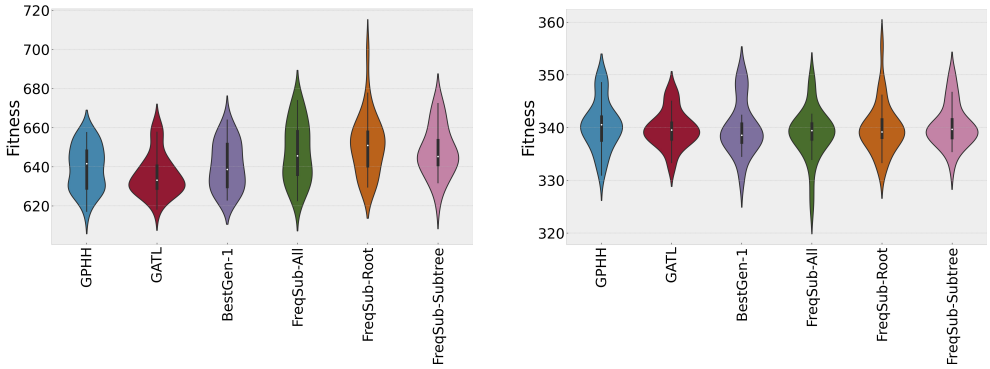


(c) *From Ugdb1 with 5 vehicles to Ugdb2 with 7 vehicles (Scn. 29)*

(d) *From Ugdb4 with 4 vehicles to Ugdb4 with 5 vehicles (Scn. 36)*

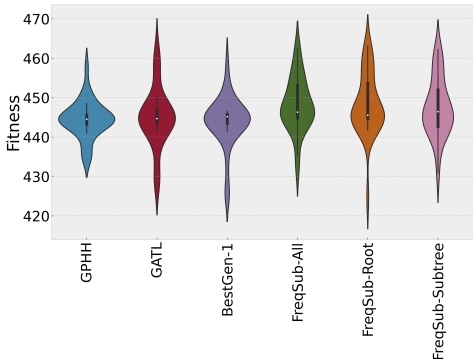
Fig. 3.4 *Convergence curve of FreqSub and some existing knowledge transfer methods.*

plots in Figure 3.4, particularly Figures 3.4c and 3.4d, reveal more interesting features. As is evident from the figures, the use of transfer optimisation has helped to create better initial population compared to the case in which there is no knowledge transfer. The extent to which the improvement has happened is different but in almost all cases it is present. The overall improvements in the quality of the initial population indicate the potential of transfer optimisation for improving the performance of GPHH for solving UCARP. However, after a few generations, the performance of these algorithms deteriorate quickly and at the end of their evolutions, the performance is clearly worse than GPHH. This is an indicator that other

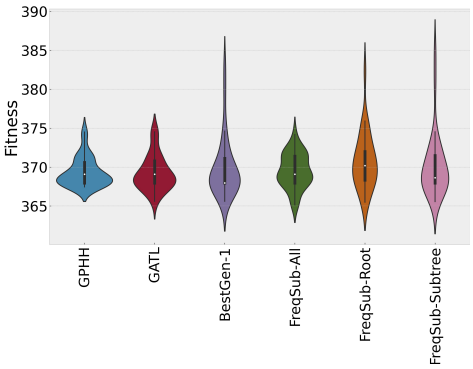


(a) *From Ugdb17 with 5 vehicles to Ugdb12 with 8 vehicles (Scn. 4)*

(b) *From Ugdb1 with 5 vehicles to Ugdb2 with 7 vehicles (Scn. 15)*



(c) *From G21 with 6 vehicles to Ugdb5 with 4 vehicles (Scn. 16)*



(d) *From Ugdb2 with 6 vehicles to Ugdb2 with 4 vehicles (Scn. 35)*

Fig. 3.5 *The distribution of solutions found with FreqSub and some existing knowledge transfer methods.*

confounding factors exist which reduce or compromise the effectiveness of knowledge transfer.

The possible reasons of the above observations can be as follows. The sub-trees of the roots are so large, and are almost unlikely to appear more than once in the final population. Thus, using frequency to choose the root sub-trees is very similar to randomly choosing them (e.g. most of them have frequency of 1). If considering all the sub-trees, then the small sub-

trees are more likely to have higher frequency, and tend to be selected. However, the frequency can be misleading, as the occurrences can be in the redundant branches. In this case, the contribution-based measure can effectively handle the misleading selections, and identify the truly important sub-trees regardless of their frequency. Therefore, the contribution-based transfer methods can work well on both the pools of all the sub-trees and the root sub-trees. Finally, another potential reason for the sub par performance of FreqSub could be that the extracted sub-trees from a knowledge source may not have enough information about the search space of the source problem. This possibility will be investigated in Section 3.5.6.

3.5.5 Effectiveness of ContribSub

As it was explained in Section 3.4.2, the existence of introns in GP trees can mislead the frequency-based approach proposed in Section 3.4.1. The contribution-based approach, ContribSubGP, in Section 3.4.2 can handle this issue and effectively discard introns. This hypothesis is investigated in this Section through experimental studies. For the sake of comparison with the body of existing literature, we also select the GATL and BestGen-1 methods and compare them with ContribSubGP.

Table 3.9: Test performance of 30 independent runs of the ContribSubGP algorithm (mean $\pm$ std)

Scn.	GPHH	GATL [129]	BestGen-1 [62]	ContribSubGP-All	ContribSubGP-Root
1	91.2±0.1	91.2±0.1	**91.2±0.1**	91.2±0.1	91.2±0.1
2	551.0±10.3	550.8±8.1	551.1±10.4	552.7±8.7	**550.8±10.5**
3	598.6±8.8	599.6±9.5	600.5±11.6	**597.8±9.6**	602.4±9.1
4	639.5±11.3	**636.0±10.7**	640.6±12.1	640.4±13.5	640.2±12.1
5	58.2±0.1	58.2±0.1	58.3±0.1	58.3±0.1	**58.2±0.1**
6	424.8±8.5	424.9±8.8	423.9±8.6	425.1±8.3	**421.2±5.5**

Continuation of Table 3.9

Scn.	GPHH	GATL [129]	BestGen-1 [62]	ContribSubGP-All	ContribSubGP-Root
7	432.1±7.1	**430.0±6.3**	431.2±6.3	431.4±7.4	431.7±5.6
8	432.6±5.5	**430.6±6.7**	432.7±4.8	432.3±6.3	434.0±4.6
9	576.2±3.9	575.8±4.2	576.8±3.7	**575.7±4.1**	576.4±4.1
10	340.5±4.7	338.1±4.2	**337.5±3.1**	337.8±3.9	339.5±4.7
11	347.2±6.1	347.1±6.0	**345.9±4.8**	349.0±5.6	346.6±6.2
12	551.0±10.3	553.7±10.5	551.8±10.1	555.1±11.0	**549.5±9.2**
13	598.6±8.8	598.5±7.5	**597.6±8.2**	610.9±14.1	603.8±12.5
14	639.5±11.3	640.1±12.2	**639.0±11.9**	650.0±14.6	640.1±12.1
15	340.5±4.7	339.8±3.5	339.9±5.0	338.8±4.6	**338.6±5.0**
16	444.4±4.7	445.0±7.6	**443.9±6.5**	449.1±9.4	445.3±7.1
17	324.3±6.2	323.3±5.2	**321.5±5.4**	323.5±5.9	322.3±5.6
18	360.3±3.1	359.7±3.7	359.4±3.9	358.7±5.3	**357.6±4.6**
19	358.3±2.6	**358.3±3.1**	358.8±2.7	363.1±7.6	358.8±3.4
20	359.0±1.8	**358.5±1.8**	358.6±1.9	359.0±1.3	358.8±1.4
21	340.8±4.4	340.8±2.2	**339.4±4.7**	340.3±3.8	343.4±4.4
22	351.9±3.5	**351.6±2.5**	352.5±3.5	353.0±4.2	352.5±3.5
23	356.6±1.6	356.4±1.5	**356.1±1.3**	356.2±1.4	356.6±1.7
24	310.9±0.5	311.0±0.3	310.7±0.8	**310.6±1.0**	310.7±0.9
25	389.2±0.2	**389.1±0.2**	389.2±0.2	389.1±0.2	389.1±0.2
26	363.1±2.8	363.1±3.2	363.4±2.6	363.5±3.2	**363.0±2.6**
27	342.1±6.2	342.5±7.8	**340.9±8.0**	341.4±5.1	341.2±5.0
28	382.0±5.5	**381.0±4.6**	381.3±8.0	386.1±6.2	384.2±6.5
29	382.8±3.3	383.9±2.6	**382.7±4.8**	384.6±5.0	383.4±6.6
30	351.5±2.5	**351.1±2.2**	351.7±1.2	351.2±1.6	351.9±1.7
31	326.0±4.7	325.2±4.0	325.2±5.0	**324.8±4.9**	326.7±4.7
32	444.4±4.7	443.7±5.6	**442.0±7.3**	445.2±6.4	446.8±7.5
33	448.2±0.5	449.0±2.3	**448.2±0.9**	449.1±1.9	448.5±0.8
34	**384.6±4.4**	387.1±6.0	386.9±5.0	387.3±5.7	386.6±4.4
35	369.3±1.8	369.3±2.2	369.8±3.8	369.7±2.8	**368.5±2.1**
36	**321.4±5.2**	322.7±4.2	323.8±5.1	324.9±5.2	324.1±5.0
37	166.2±2.0	166.1±1.7	**165.2±1.5**	165.9±2.0	165.6±1.7
38	**376.1±7.6**	381.2±7.0	377.8±7.5	381.6±6.7	377.1±8.2
39	415.7±9.2	415.5±7.2	**414.3±8.9**	416.3±6.8	415.8±6.6

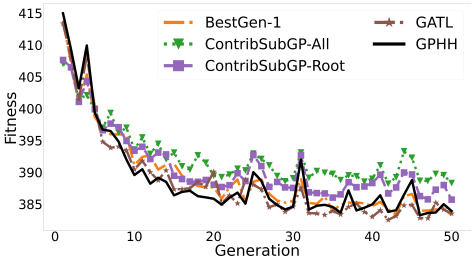
Table 3.10: Adjusted p-value of the pairwise comparison of the ContribGP algorithm

	GATL	BestGen-1	ContribSubGP-All	ContribSubGP-Root
GPHH	0.21	0.13	0.12	0.86
GATL	–	0.770727	**0.003**	0.25
BestGen-1	–	–	**0.001**	0.15
ContribSubGP-All	–	–	–	0.11

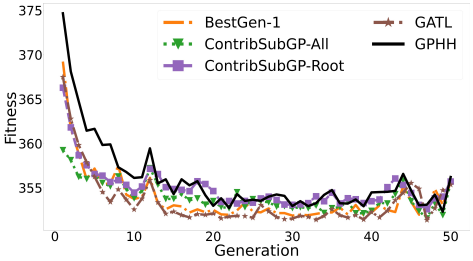
Continuation of Table 3.9

Scn.	GPHH	GATL [129]	BestGen-1 [62]	ContribSubGP-All	ContribSubGP-Root
40	347.2±6.1	347.5±5.2	**345.8±4.4**	347.3±5.6	347.4±6.4
41	351.5±2.5	**351.4±2.7**	352.0±2.3	351.5±2.4	351.9±1.7
42	165.9±1.8	**165.6±1.7**	165.7±1.6	165.9±1.7	165.8±1.7
43	462.6±6.0	**457.4±7.0**	460.2±5.4	461.2±5.4	460.7±5.6
44	**426.6±3.3**	427.3±2.0	427.0±2.6	428.1±2.2	427.0±1.5
45	499.0±3.9	498.5±4.4	**497.6±4.4**	500.4±7.2	498.6±4.4
Rank	3.1	2.63	2.52	3.72	3.02
Friedman's p-value					**0.003**

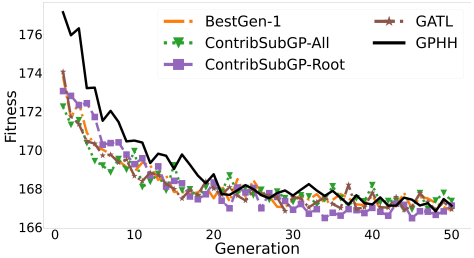
The average performance of 30 independent runs of the compared algorithms are given in Table 3.9. Table 3.10 also presents the adjusted p-values of the pairwise comparisons of the algorithms. Similar to the FreqSub algorithm, ContribGP ranked worse than GPHH but in this case, the algorithm is not significantly worse than GPHH. This indicates that the contribution measure is at least more effective than the frequency measure for evaluating the importance of sub-trees. Another observation from Table 3.9 is that ContribSubGP-Root is ranked better than ContribSubGP-All and it is even ranked slightly better than GPHH. This can be attributed to the fact root sub-trees have more genetic materials and therefore, they contain more reusable information.



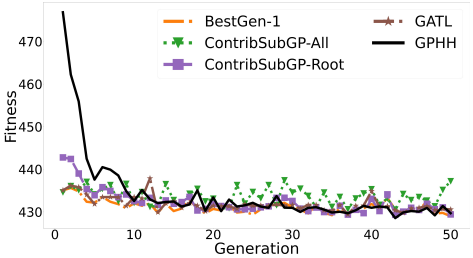
(a) *From Ugdb1 with 5 vehicles to Ugdb1 with 7 vehicles (Scn. 28)*



(b) *From Ugdb2 with 6 vehicles to Ugdb6 with 6 vehicles (Scn. 30)*



(c) *From Ugdb21 with 6 vehicles to Ugdb21 with 5 vehicles (Scn. 42)*



(d) *From val8A with 3 vehicles to val8Adm1 with 3 vehicles (Scn. 44)*

Fig. 3.6 *Convergence curve of ContribSubGP and some existing knowledge transfer methods.*

Figure 3.6 presents the convergence curve of the algorithms for a few scenarios. This figure also confirms the observation in Table 3.9 about the sub-par performance of ContribSub algorithm. An important observation from Figure 3.6 is that, similar to FreqSub, PPTGP and some of the existing algorithms, has also been able to create good initial populations for GP, as is clearly evident in Figures 3.6b–3.6d. For the case of Scenario 44, whose convergence curve is shown in Figure 3.6d, the initial state is significantly better than the state of GP without any knowledge transfer. However, as the evolution proceeds, not only the performance of the algorithms does not improve, for the ContribSub-all algorithm, it degrades significantly. This is again an indication that the transfer of sub-trees has potential for

improving the performance but then, during the course of evolution, other confounding factors may exist that degrades the performance. This issue will be investigated further in Section 3.5.7.

Comparing the performance of ContribSubGP-Root in Table 3.9 with the performance of FreqSub-Root in Table 3.7 is also informative. Both these algorithms consider the immediate sub-trees of the root nodes as the transferable knowledge. However, ContribSubGP-Root measures the importance of sub-trees with the contribution that they make to the fitness of their individuals. Consequently, ContribSubGP is more effective at finding the useful sub-trees. As a result, it can be noted that while the performance of FreqSub-Root is significantly worse than GPHH, ContribSubGP has performed slightly, although not significantly, better than GPHH. The effectiveness of the contribution measure can be examined with the help of Figure 3.7a.

The Figure 3.7a shows an individual in the best individuals of the final population in the source domain of Ugdb8 (i.e. with 10 vehicles). It contains a sub-tree shown in Figure 3.7b. The sub-tree appeared 11 times in the individuals and the FreqSub-All method considered it as useful and transferred it into the target domain. However, a deeper investigation reveals that many of the occurrences of this sub-tree are in fact in the redundant branches, and the contribution (calculated by the measure introduced in [160]) of the sub-tree to its individuals is zero. This indicates that the sub-tree is actually not useful for transfer. Figure 3.7c shows another more complex sub-tree that received different opinions from the frequency and contribution measures. This sub-tree appeared 47 times in the individuals of the source individuals, and thus was transferred by the Frequent-All method. However, its contribution to the tree shown in Fig. 3.7a is -47.57. Thus, the ContribSubGP-All method considered it as a useless sub-tree, and did not transfer it to the target domain.

Note that the above two sub-trees have complex structures, and are not easily simplified. Thus, the commonly considered algebraic simplifi-

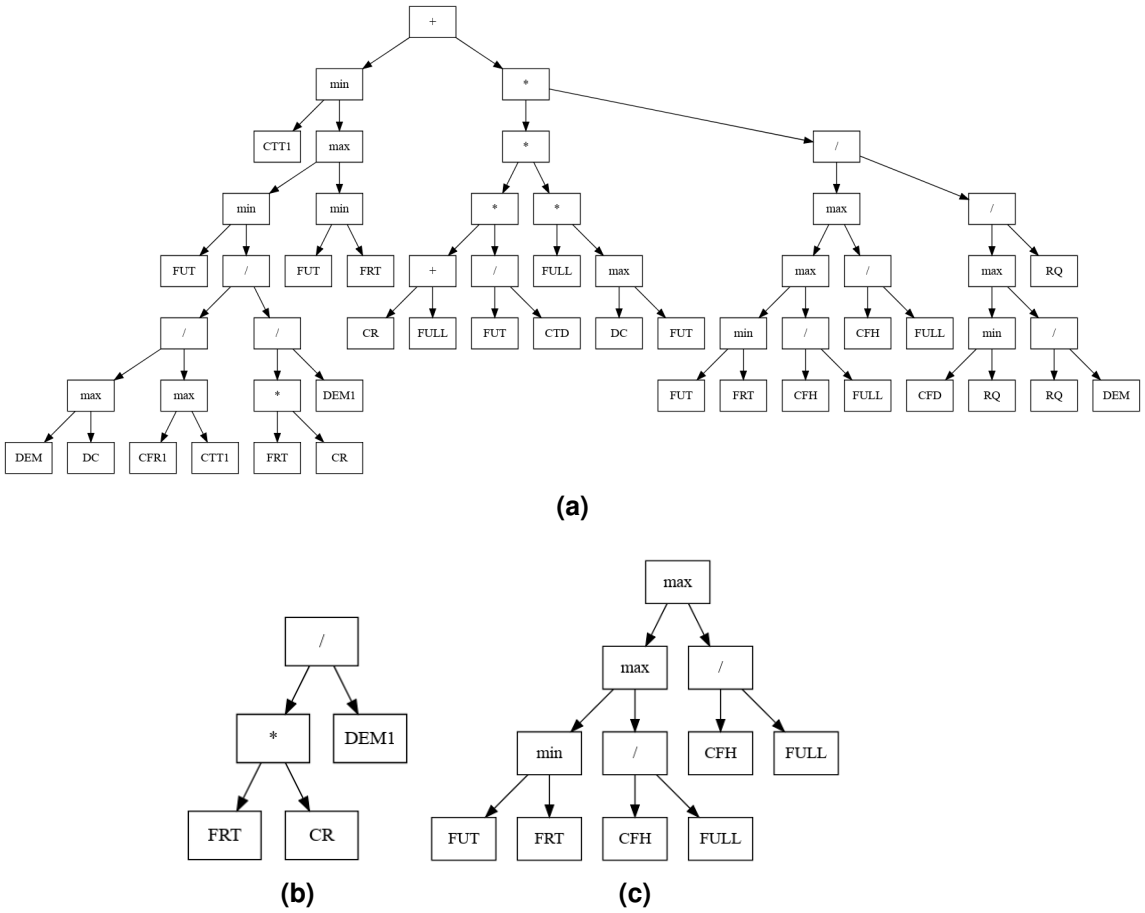


Fig. 3.7 *(a) A GP individual and two of its (b) redundant and (c) detrimental subtrees*

cation [251, 210] (e.g. $a/a = 1$) plus frequency measure still cannot effectively detect the important sub-trees for transfer. This demonstrates the effectiveness of using the contribution-based measure in identifying useful sub-trees.

3.5.6 Effectiveness of PPT Transfer

In Section 3.4.3, it was pointed out that transferring knowledge through the transfer of sub-trees may not convey comprehensive information about the search space of the source problem. Additionally, even the full trees have the limitation that they are just samples from the distribution of source solutions. Accordingly, the PPTGP algorithm was proposed in Section 3.4.3 for learning the probability distribution of high-quality source solutions. In this section, we investigate this hypothesis. As in Section 3.5.4, we also consider the GATL and BestGen-1 algorithm from the body of existing methods too.

Table 3.11 presents the average performance of 30 independent runs PPTGP and a set of existing algorithms and the adjusted p-values of the post-hoc pairwise comparisons of the algorithms are given in Table 3.12.

Table 3.11: Test performance of 30 independent runs of the PPTGP algorithm (mean $\pm$ std)

Scn.	GPHH	GATL [129]	BestGen-1 [62]	PPTGP
1	**91.2±0.1**	91.2±0.1	91.2±0.1	91.2±0.1
2	551.1±10.4	**550.8±8.1**	551.0±10.3	555.5±7.7
3	600.5±11.6	599.6±9.5	**598.6±8.8**	601.2±10.8
4	640.6±12.1	**636.0±10.7**	639.5±11.3	642.9±11.4
5	58.3±0.1	58.2±0.1	**58.2±0.1**	58.3±0.1
6	**423.9±8.6**	424.9±8.8	424.8±8.5	427.0±7.7
7	431.2±6.3	**430.0±6.3**	432.1±7.1	432.6±6.7
8	432.7±4.8	**430.6±6.7**	432.6±5.5	432.6±4.8
9	576.8±3.7	**575.8±4.2**	576.2±3.9	577.5±3.5
10	**337.5±3.1**	338.1±4.2	340.5±4.7	338.1±4.3
11	**345.9±4.8**	347.1±6.0	347.2±6.1	347.4±5.9
12	551.8±10.1	553.7±10.5	**551.0±10.3**	555.6±8.2
13	**597.6±8.2**	598.5±7.5	598.6±8.8	602.7±9.8
14	**639.0±11.9**	640.1±12.2	639.5±11.3	641.2±13.9
15	339.9±5.0	**339.8±3.5**	340.5±4.7	340.7±4.0

Continuation of Table 3.11

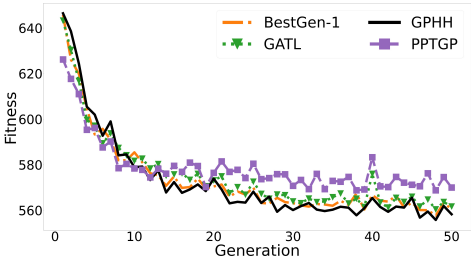
Scn.	GPHH	GATL [129]	BestGen-1 [62]	PPTGP
16	**443.9±6.5**	445.0±7.6	444.4±4.7	444.4±7.2
17	**321.5±5.4**	323.3±5.2	324.3±6.2	322.7±6.0
18	**359.4±3.9**	359.7±3.7	360.3±3.1	359.7±4.1
19	358.8±2.7	**358.3±3.1**	358.3±2.6	358.6±6.4
20	358.6±1.9	**358.5±1.8**	359.0±1.8	359.3±1.2
21	**339.4±4.7**	340.8±2.2	340.8±4.4	340.5±4.1
22	352.5±3.5	**351.6±2.5**	351.9±3.5	352.0±3.4
23	**356.1±1.3**	356.4±1.5	356.6±1.6	356.9±1.8
24	**310.7±0.8**	311.0±0.3	310.9±0.5	311.0±0.5
25	389.2±0.2	**389.1±0.2**	389.2±0.2	389.2±0.6
26	363.4±2.6	363.1±3.2	363.1±2.8	**362.5±3.5**
27	**340.9±8.0**	342.5±7.8	342.1±6.2	341.9±4.8
28	381.3±8.0	**381.0±4.6**	382.0±5.5	384.4±7.1
29	**382.7±4.8**	383.9±2.6	382.8±3.3	384.7±5.7
30	351.7±1.2	**351.1±2.2**	351.5±2.5	351.4±1.9
31	**325.2±5.0**	325.2±4.0	326.0±4.7	326.2±5.4
32	442.0±7.3	443.7±5.6	444.4±4.7	**441.8±7.1**
33	**448.2±0.9**	449.0±2.3	448.2±0.5	448.7±1.0
34	386.9±5.0	387.1±6.0	**384.6±4.4**	385.5±4.9
35	369.8±3.8	369.3±2.2	369.3±1.8	**369.2±2.1**
36	323.8±5.1	322.7±4.2	**321.4±5.2**	325.5±5.6
37	**165.2±1.5**	166.1±1.7	166.2±2.0	165.5±2.0
38	377.8±7.5	381.2±7.0	**376.1±7.6**	380.9±8.4
39	414.3±8.9	415.5±7.2	415.7±9.2	**414.3±7.8**
40	**345.8±4.4**	347.5±5.2	347.2±6.1	349.0±6.7
41	352.0±2.3	**351.4±2.7**	351.5±2.5	351.4±1.5
42	165.7±1.6	165.6±1.7	165.9±1.8	**165.4±1.6**
43	460.2±5.4	**457.4±7.0**	462.6±6.0	461.0±4.7
44	427.0±2.6	427.3±2.0	**426.6±3.3**	426.9±2.4
45	**497.6±4.4**	498.5±4.4	499.0±3.9	499.2±4.8
Rank	2.53	2.14	2.28	3.04
Friedman's p-value				**0.005**

As is evident from Tables 3.11 and 3.12, the performance of PPTGP is significantly worse than GPHH and all the existing algorithms. This effect can be seen in the convergence curves of the algorithms in Figure 3.8 and the violin plots in Figure 3.9.

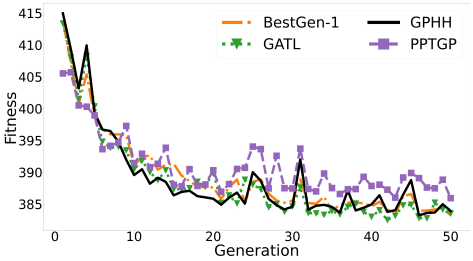
Looking at the convergence curve of the algorithms in Figure 3.8, it is noticeable that the initial state of PPTGP is better than other algorithms. Particularly, the difference is very noticeable in Figures 3.8b and 3.8c. Similar to the observation in Sections 3.5.3 and 3.5.4, Figure 3.8 also confirms the potentials of transfer optimisation for improving GP performance. However, the fact that the performance of PPTGP deteriorates significantly indicate that there are other factors that inflict the performance. Another important observation from the tables is that both GATL and BestGen-1 perform significantly better than PPTGP. What makes this observation important is the fact that PPTGP is designed to learn the probability distribution of good source GP individuals. Looking at the convergence curves of PPTGP in Figure 3.8, it can be seen that the algorithm is successful in this regard, as the initial GP state that it creates is better than GPHH without knowledge transfer and also other existing methods. GATL and BestGen-1, on the other hand, do not focus on the quality but instead, they collect individuals from every GP generation for solving the source problem. Consequently, although their initial performance was not better than PPTGP, over time they improved and became better than PPTGP. The reasons for this behaviour will be investigated in Section 3.5.7.

Table 3.12: Adjusted p-value of the pairwise comparison of PPTGP algorithm

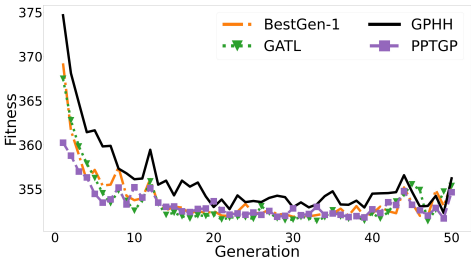
	GATL	BestGen-1	PPTGP
GPHH	0.38	0.18	0.1
GATL	–	0.57	**0.01**
BestGen-1	–	–	**0**



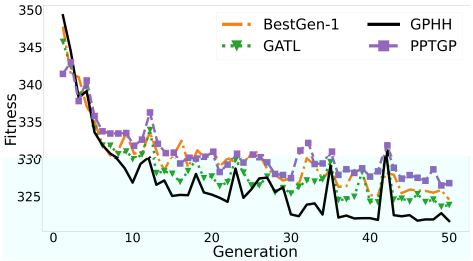
(a) *From Ugdb23 with 10 vehicles to Ugdb12 with 5 vehicles (Scn. 12)*



(b) *From Ugdb1 with 5 vehicles to Ugdb1 with 7 vehicles (Scn. 28)*



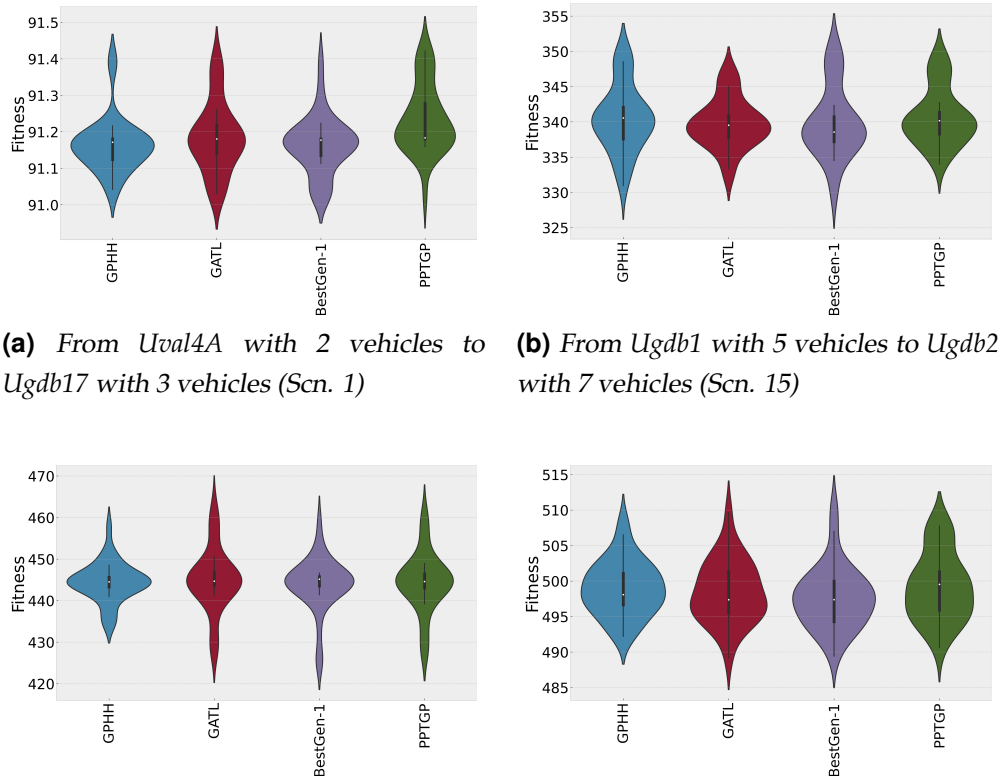
(c) *From Ugdb2 with 6 vehicles to Ugdb6 with 6 vehicles (Scn. 30)*



(d) *From Ugdb4 with 4 vehicles to Ugdb4 with 5 vehicles (Scn. 36)*

Fig. 3.8 *Convergence curve of PPTGP and some existing knowledge transfer methods.*

The main idea in PPT-based transfer learning is to learn the probability distribution of high-quality routing policies from a knowledge source and reuse it for solving the target problem. Figures 3.10–3.11 presents a PPT that is extracted from the *gdb1, 5 vehicles* source. The figure also contains the magnified view of top 2 level of nodes of the PPT. The numbers inside nodes indicate the probability of selecting corresponding GP function/terminal at that location and the labels beside each node show the indexing order defined over this PPT. The probability of missing items is zero In this work we define the underlying knowledge in the source as being the probability distribution of good GP individuals, granulated at the node level. As is seen, in high-quality source solutions, the node at



(a) *From Uval4A with 2 vehicles to Ugdb17 with 3 vehicles (Scn. 1)*

(b) *From Ugdb1 with 5 vehicles to Ugdb2 with 7 vehicles (Scn. 15)*

(c) *From G21 with 6 vehicles to Ugdb5 with 4 vehicles (Scn. 16)*

(d) *From val5Adm1 with 3 vehicles to val5Adm2 with 3 vehicles (Scn. 45)*

Fig. 3.9 *The distribution of solutions found with PPTGP and some existing knowledge transfer methods.*

$i = 3$ has a categorical distribution with parameters $p_{3,max} = 0.06, p_{3,*} = 0.54, p_{3,-} = 0.36, p_{3,min} = 0.04$.

It is interesting to note that PPTs can capture the relation between GP items with respect to their position in trees. For example, based on Figures 3.10–3.11, we can see that $\{max, *\}$ are important at node 2 and $\{max, *, -min\}$ are important at node 4 and therefore, any combination of these sets can be considered important, specifically between $\{max\}$, from node 2 and $\{*, -\}$ from node 4. This type of relation is difficult to capture with methods like

FullTree and allows for the creation of trees that may not be seen before. For example, the tree

$$\begin{aligned}
&(((((0.87 \max CFH) \max(SC - CTD)) \max(FUT * 0.13)) \\
&\min(((SC - CTD) \min(FUT * (FUT * SC))) - CTD)) \max \\
&((((0.66 \max CFH) \max(SC - CTD)) \max(((RQ - CTD) \max CR) \\
&\min(SC * SC))) * (((SC - CTD) \min(FUT * (FUT * SC))) \\
&\max CFH))) \max((CFD \max(((SC + CFH) - (SC - CTD)) \\
&\max((SC \min(FRT * DC)) \max(CTT1 * CTD)))) * (RQ - DC))
\end{aligned} \tag{3.7}$$

The tree in Equation 3.7 is one of the trees that is sampled from this PPT (written in infix notation). With a fitness value of 372.74, this was the best individual in the first population in target domain. This tree, or similar trees, was not present in the source domain and therefore, methods like *FullTree* could not transfer it. Interestingly, the best individual that *FullTree* transferred in this experiment had a fitness value of 380.94.

Looking at the PPT in Figures 3.10–3.11, it is easy to see that the probability of some GP functions/terminals is zero. Consequently, these items will not be selected for those nodes when the PPT is sampled. However, it is possible that the neglected items may have some degree of importance in a new target domain but our current approach cannot handle this and we aim to address this issue in future work.

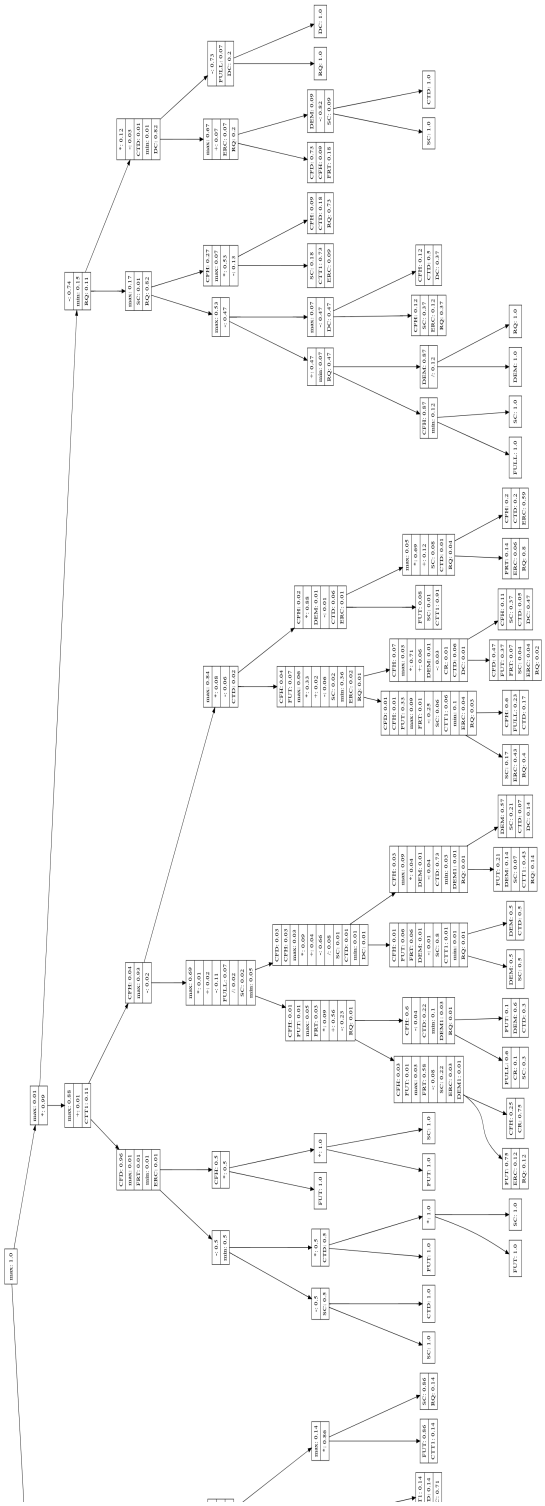


Fig. 3.10 *A learned PPT*

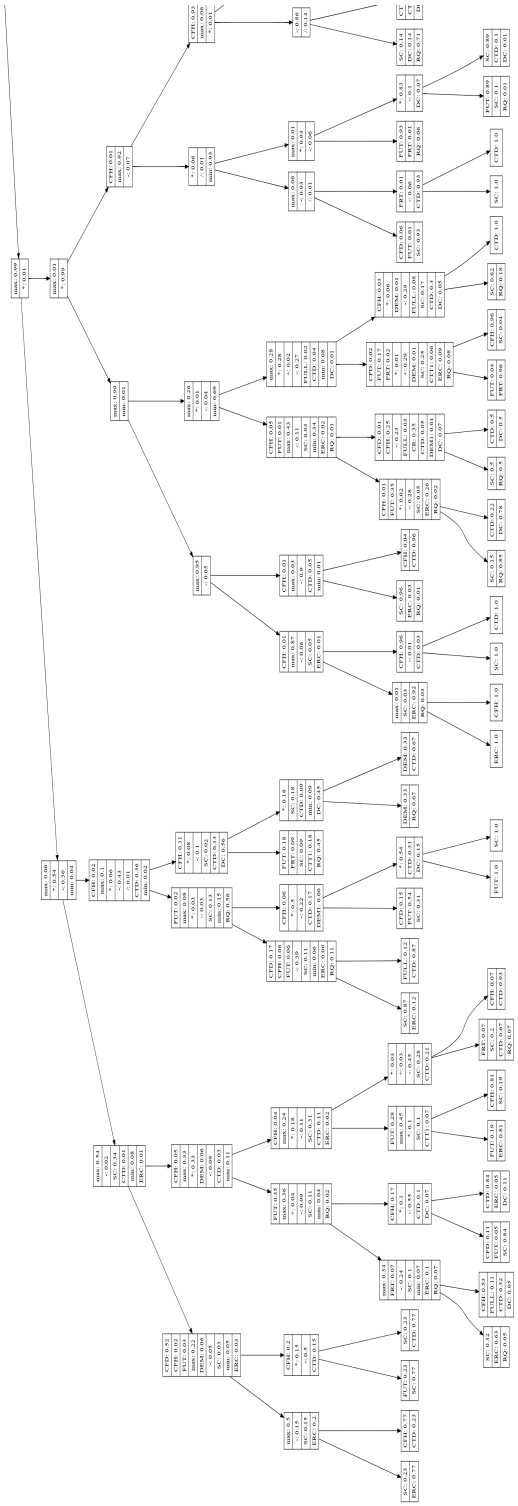


Fig. 3.11 *A learned PPT*

3.5.7 Further Analysis

The experimental results in Sections 3.5.3 – 3.5.6, revealed a few important points. In almost all experiments, it was noticed that the act of knowledge transfer has the potential for improving the performance of GP, as is indicated by the initial state of algorithms. However, in all experiments, the initial performance gain from the transferred knowledge does not last for long. This is an indicator that other confounding factors may exist that negate the good improvement that was gained with the transferred knowledge. In this section, further investigations are conducted to determine other factors that can contribute to the quality of knowledge transfer for solving UCARP with GP. For this, we first investigate the quality of the knowledge source to determine if any issues could exist in the source. Next, it will be investigated what issue(s) during the process of solving the target problem could exist that may contribute to reducing the quality of knowledge transfer.

Effect of Knowledge Source Quality on Knowledge Transfer

Based on our experimental results, in most cases, the algorithms that transferred whole GP trees performed better than the sub-tree-based methods. This is practically evident in the performance of GATL and BestGen-1 methods. These methods performed better than all the proposed algorithms. The common property of these methods is that both of them transfer full trees, rather than sub-trees, and both of them consider all the source populations and not just the final source population. This motivates the investigation of the knowledge source to find out what the reason(s) for this observation could be. In order to do so, we first investigate the distribution of GP individuals that were found for solving the source problem. The distribution is shown in Figure 3.12 in the form of a violin plot.

Figure 3.12 reveals an important property about the knowledge source. Based on the this figure, the majority of the individuals in the knowledge

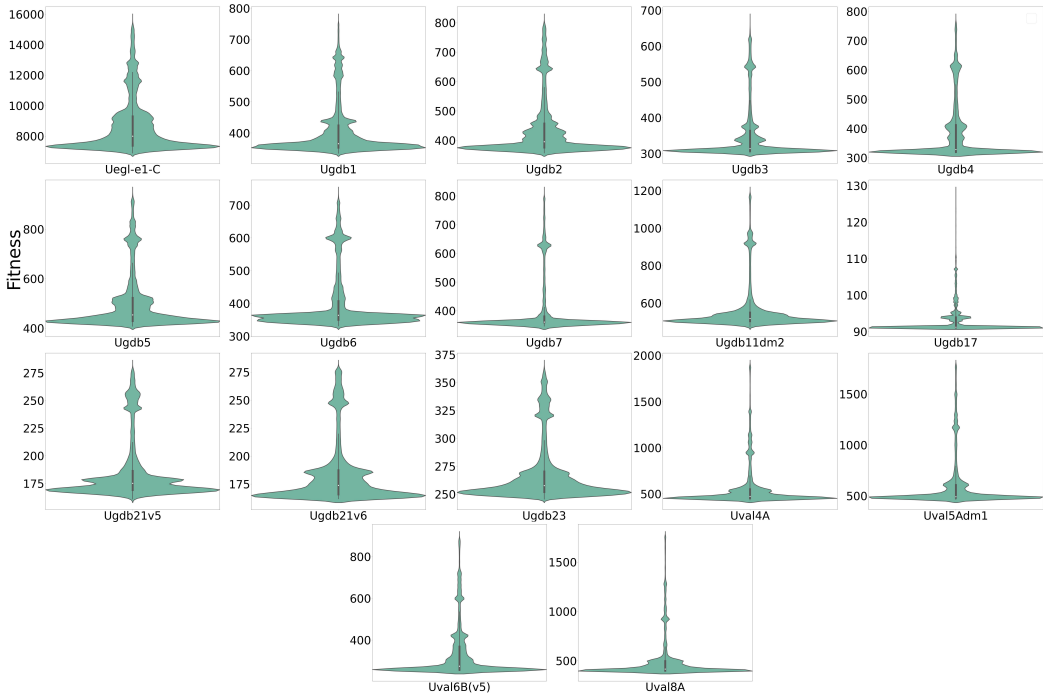


Fig. 3.12 *The distribution of the fitness values of all individuals that were found for solving each source problem.*

source have a very high fitness value. Considering the rather large number of source individuals, i.e. 50×1024 according to our experimental settings, it was expected that the fitness values were distributed more evenly. However, the plot indicate that most of the individuals have very similar, or even exact, fitness values. This observation indicate the presence of duplicates due to the fact that many individuals have the same fitness values. To confirm this, we investigated the presence of duplicates in the source. For this purpose, we focused on phenotypic similarity of routing policies. As was discussed in Section 2.7.5, phenotypic characterisation of routing policies is better capable of identifying the similarities between routing policies. Therefore, we utilise the phenotypic characterisation method proposed by Hildebrandt et al. [105] and described in Section 2.7.5 (page 55) to identify duplicates in the set of all individuals that were found for solv-

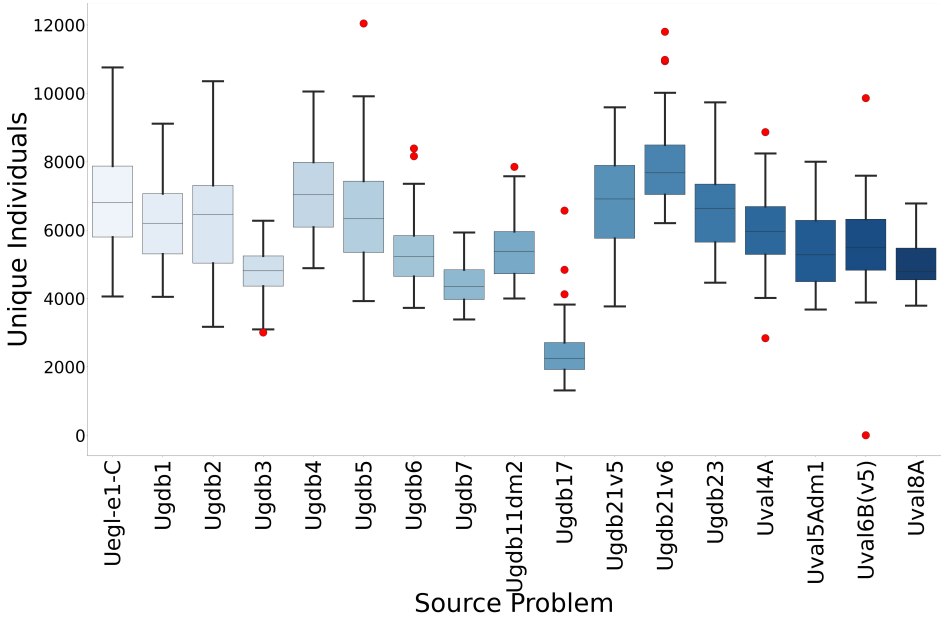


Fig. 3.13 *Number of available unique individuals in the knowledge source*

ing the source problems.

To investigate this, we recorded the number of unique individuals that were found in each source problem as defined in Table 3.1, averaged them over all runs and summarised the results into the box plots in Figure 3.13. Considering that there are $1024 \times 50 = 51200$ transferred individuals according to our experiment settings, it is obvious from the plots that in almost all scenarios, the source contained a large number of duplicates to the point that at best, 24% of the individuals were unique (*Ugdb5*). For some cases like *Ugdb3*, this rate fell even below 10%. To investigate further, we plotted the distribution of the unique and duplicate individuals in each source problem against their fitness value in Figure 3.14. As is evident, the majority of high-quality individuals were duplicates, transfer of which will amount to transfer of redundant knowledge that cannot be helpful for solving the target problem. As a result, the performance of any method, such as the proposed algorithms in this chapter, that does not

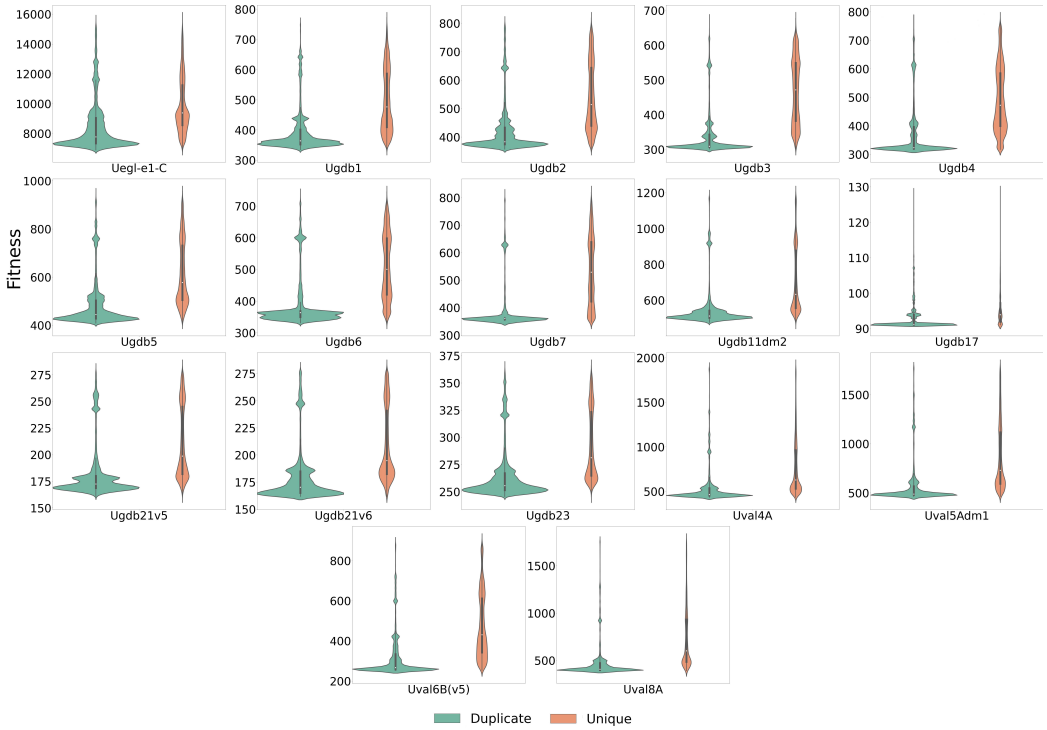


Fig. 3.14 *Distribution of unique and duplicate individuals in source problem.*

consider this redundancy, will suffer.

The observations in Figures 3.13 and 3.14 also explain why the methods such as GATL and BestGen-1 demonstrated relatively better performances. Since the majority of duplicates have high qualities, they must appear in later GP generations. Therefore, methods such as FullTree, FreqSub and PPTGP, that focus only on the last source population, will have a large number of redundant duplicates that do not convey any important information. Needless to say, when this issue is not handled properly, the issue of lack of diversity could be transferred to the target problem and impact the performance of GP negatively, as was observed by the experimental results. On the other hand, GATL and BestGen-1 select their individuals from all the source populations and as a consequent, they are less

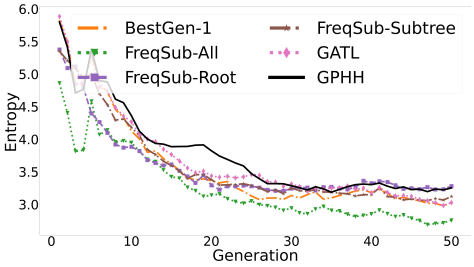
prone to the negative impact of the diversity loss in the final generation of GP for solving the source problem.

Issues When Solving Target Problems

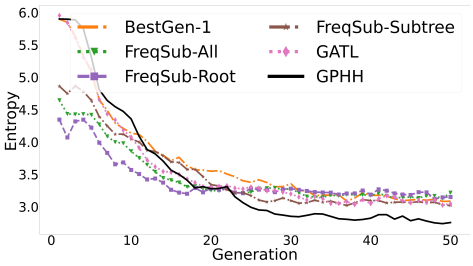
As the experimental results revealed in Sections 3.5.2 – 3.4.2, knowledge transfer has the potential for improving the performance of GP for solving UCARP. However, any initial improvement in the quality of GP population for solving a target problem is quickly lost after a few generations. This indicate that some issues may exist during the evolutionary process for solving the target problem that neutralise any initial improvements. As was discussed earlier, the knowledge source may contain a substantial amount of duplicates. This indicates that the GP process for solving the source problem suffers from losing its diversity. As a result, it is reasonable to suspect that the same issue may exist when solving the target problem. To investigate this, we investigate the population diversity during the evolutionary search for solving the target problem.

To investigate the presence of redundancies, we employ the entropy measure [32] for calculating the population diversity. To compute the entropy of the population Pop, we grouped the similar individuals into a set of clusters $\mathcal{C}$ using the DBScan clustering algorithm [69], where each individual is characterised by the phenotypic vector [105] and the cluster radius was set to zero. Then, the entropy of the population is calculated as $entropy(Pop) = -\sum_{c \in \mathcal{C}} \frac{|c|}{|Pop|} \log \frac{|c|}{|Pop|}$.

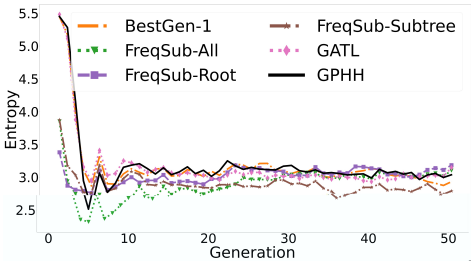
Figures 3.15–3.17 present the average entropy of GP population over the course of evolution for solving target problems for a few scenarios and for the different sets of examined algorithms. These figures confirm the hypothesis that GP population loses its diversity very quickly. This phenomenon is observed in all experiments and for all algorithms. In all experiments, we note that although the algorithm may have a relatively high entropy at first, their entropy decreases quickly and in most cases, their entropy is not very different from GPHH.



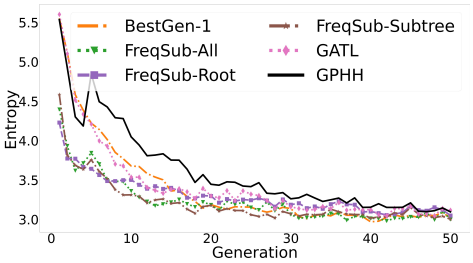
(a) *From Ugdb17 with 5 vehicles to Ugdb12 with 8 vehicles (Scn. 4)*



(b) *From Ugdb23 with 10 vehicles to Ugdb12 with 5 vehicles (Scn. 12)*



(c) *From Uval6B with 5 vehicles to Ugdb6 with 5 vehicles (Scn. 15)*

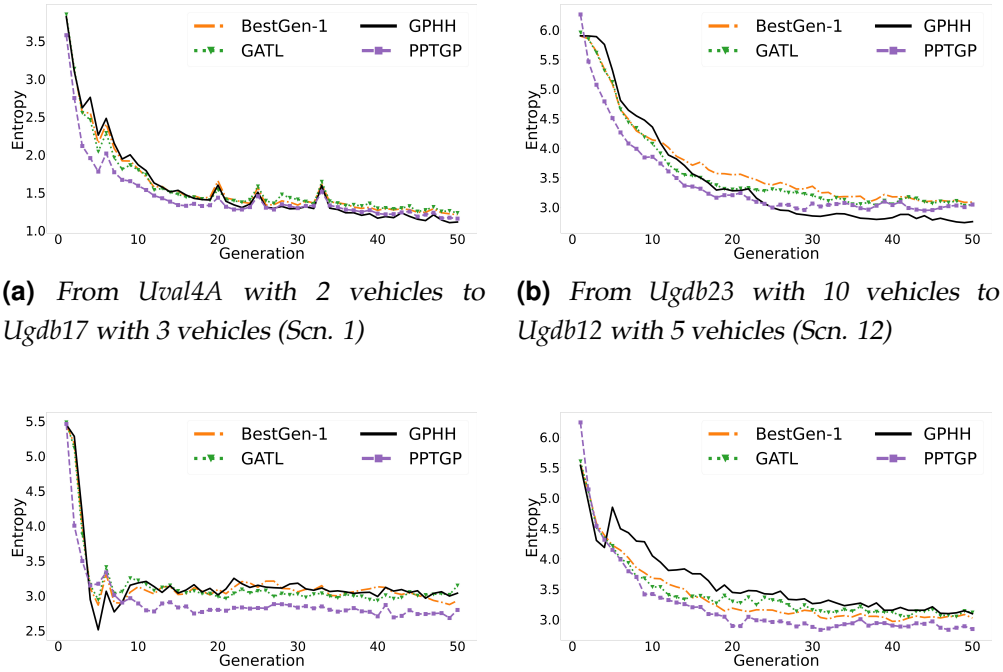


(d) *From Ugdb1 with 5 vehicles to Ugdb1 with 3 vehicles (Scn. 22)*

Fig. 3.15 *Population entropy of FreqSub and some existing knowledge transfer methods.*

Another point to note is that the algorithms that transfer sub-trees and full trees from the final source population usually tend to have lower initial entropy. This behaviour is expected considering the fact that final source population contains many redundancies. On the other hand, the BestGen-1 and GATL methods that consider all the source GP populations usually have relatively higher initial population entropies. However, PPTGP is an exception to these observations because this algorithm learns only from the final source population but nevertheless, its initial population entropy is higher than other knowledge transfer methods. This observation can be explained with the fact that this algorithm does not transfer actual trees but it transfers the probability distribution of high-quality

source trees in the form of PPTs. Then, when initialising the GP population, the algorithm creates new individuals from the transferred PPT. Therefore, even if the source contains many duplicates, the PPT will still contain many possibilities for creating individuals. However, looking at the convergence curve of this algorithm in Fig. 3.8, we can note that the initial performance of this algorithm is not much better than GPHH without any knowledge transefer. This indicates that the algorithm was not very good at capturing the knowledge. In addition to this, although the initial population has a high level of diversity, the effect also does not last for very long and after a few generations, the population is overrun with duplicates. Ultimately, these two factors contributed to the poor performance



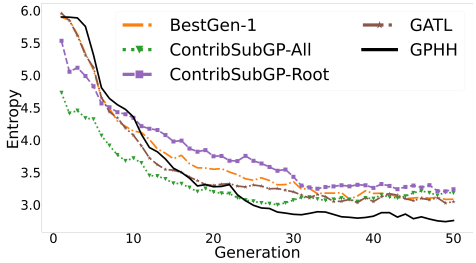
(a) *From Uval4A with 2 vehicles to Ugdb17 with 3 vehicles (Scn. 1)*

(b) *From Ugdb23 with 10 vehicles to Ugdb12 with 5 vehicles (Scn. 12)*

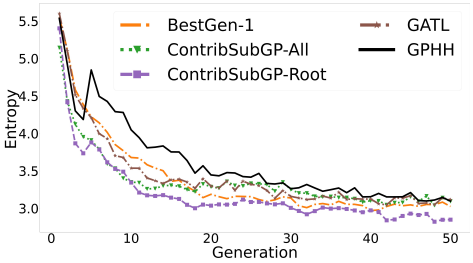
(c) *From Uval6B with 5 vehicles to Ugdb6 with 5 vehicles (Scn. 15)*

(d) *From Ugdb1 with 5 vehicles to Ugdb1 with 3 vehicles (Scn. 22)*

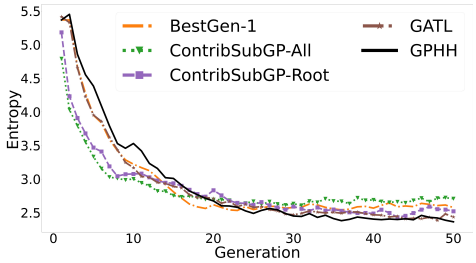
Fig. 3.16 *Population entropy of PPTGP and some existing knowledge transfer methods.*



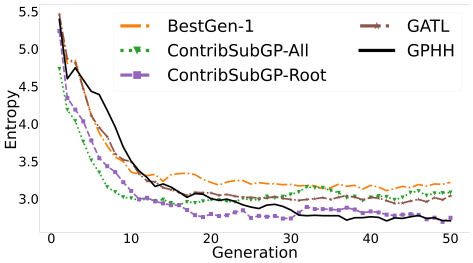
(a) *From Ugdb23 with 10 vehicles to Ugdb12 with 5 vehicles (Scn. 12)*



(b) *From Ugdb1 with 5 vehicles to Ugdb1 with 3 vehicles (Scn. 22)*



(c) *From Ugdb1 with 5 vehicles to Ugdb1 with 7 vehicles (Scn. 28)*



(d) *From Ugdb2 with 6 vehicles to Ugdb6 with 6 vehicles (Scn. 30)*

Fig. 3.17 *Population entropy of ContribSubGP and some existing knowledge transfer methods.*

of the algorithm. Similar behaviour was observed in all experiments and for all algorithms, as can be seen in Figure 3.18. This figure presents the distribution of the entropy of the final GP population for solving target problems for all scenarios. As is evident, for all algorithms, final populations do not have high levels of diversity and are not very different from GPHH without any knowledge transfer. Accordingly, it is evident that the GP process for solving the target problem is prone to losing its population diversity and this effect can reduce the effectiveness of knowledge transfer for solving UCARP with GP.

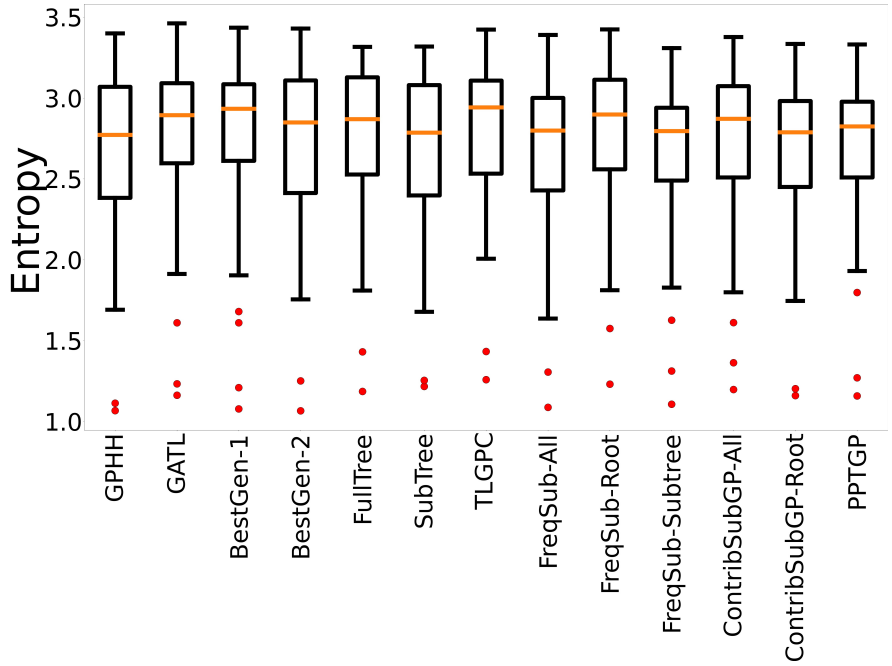


Fig. 3.18 *Entropy of the final GP population that is obtained with the examined algorithms*

3.6 Chapter Summary

In this chapter, we investigated the potential and the effectiveness of transfer optimisation for solving UCARP with GP. For this purpose, we designed a set of knowledge transfer scenarios in which transfer optimisation is performed to extract knowledge from a solved source problem and is transferred and reused for solving a related target problem. For our investigations, we considered a set of existing algorithms and additionally, we also proposed three new transfer optimisation algorithms aiming to address some of the shortcomings in the existing methods.

Our results revealed the potential of transfer optimisation for improving the performance of GP for solving UCARP. In particular, one key insight which was gained in this from the experiments in this chapter is that we discovered that reuse of genetic materials as transferable knowl-

edge can be especially helpful for initialising the GP population. However, reusing the genetic materials during the GP run, as demonstrated with the TLGPC algorithm, did not show any promise in our experiments. Furthermore, we observed that reusing the knowledge in all source generations is more helpful and the algorithms that focused only on the final source population did not perform as well as the algorithms that considered all source populations. Another important finding is that the GP process for solving the source problem can create many phenotypic duplicates, which reduces the amount of reusable knowledge and can reduce the effectiveness of knowledge transfer. Furthermore, the search process for solving the target problem can also lose its population diversity, which can also negate any initial performance boost that can be achieved with the transferred knowledge. These observations will form the baseline for the rest of this thesis and will be considered for proposing new algorithms in later chapters. Accordingly, although it is possible to investigate some of the new approaches in this chapter further, e.g. PPTs can potentially be utilized to implement surrogate-assisted sub-tree and fulltree mutation, we opt to leave such investigations for future work and instead, focus on the important insights that the experiments gave us.

More specifically, in the next chapter, we will introduce a set of algorithms to handle the presence of duplicates in the knowledge source. These algorithms will be subject to extensive experimental studies and their effectiveness will be investigated in the next chapter.

Chapter 4

Diversity-Driven Knowledge Transfer

As was discussed in Chapter 3, the presence of duplicates in the knowledge source can pose a significant negative impact on the effectiveness of knowledge transfer for UCARP. Accordingly, in this chapter we investigate the approaches that can be taken for handling this issue.

4.1 Introduction

As we demonstrated in Chapter 3, applying transfer optimisation for UCARP is not a straightforward task. We evaluated a set of existing methods for handling the problem changes of UCARP and showed that their performance were not significantly different from GPHH without any knowledge transfer. In addition, we discovered that the UCARP knowledge source can contain many duplicates, possibly due to convergence to local optima. This can reduce the amount of transferable knowledge and decrease the quality of knowledge transfer. Accordingly, in this chapter we aim to handle this problem.

In this chapter, we propose two approaches to handling the presence of duplicates in the transferred knowledge source. In the first approach,

we will propose one algorithm that strives to increase the diversity after the transfer of individuals from a solved source problem that contains duplicates. If the GP search process for solving the source problem had suffered from the loss of diversity, this algorithm could prevent the knowledge transfer from transferring the duplicates to the GP search process for solving the target problem. The proposed method is referred to as Diversity-Driven GPHH (DDGP), which relies on different mutation adaptation strategies for increasing the diversity in GP population. DDGP is described in Section 4.4.1.

In the second approach, two algorithms are proposed that are focused on removing the duplicates from the knowledge source *before* transferring and reusing the source for solving the target problem. Since the removal of the duplicates may reduce the number of high-quality individuals from the knowledge source, these algorithms rely on the surrogate models for increasing the number of high-quality solutions in the knowledge source.

The first algorithm in the second approach is called a Surrogate-Assisted Knowledge transfer for GPPHH (SAKGPHH) in which, we first create a surrogate model from the set of all individuals that were discovered for solving the source problem. This surrogate model allows us to get a computationally cheap approximation of GP individuals and can be used to evaluate individuals for solving the target problem. In this direction, we utilise the learned surrogate model to create a diverse and high-quality initial state for GP. For this purpose, we create a large pool of random individuals, remove the phenotypic duplicates from it, and evaluate the pool with the surrogate to estimate the quality of each individual in it. Finally, the best unique individuals in the pool, in terms of the surrogate fitness, are selected to initialise GP. This algorithm is explained in details in Section 4.4.2 and later in this chapter, we will demonstrate that this approach can create a good and diverse initial population for GP that can lead to the improved effectiveness of GPHH for solving UCARP.

However, one shortcoming of this algorithm is that the it does not

utilise the source individuals directly but instead, employs them to evaluate a pool of randomly created individuals and, it is the individuals in the random pool that are used for initialising GP. As a result, instead of using the actual source individuals whose actual fitness value is known, the random individuals are used for which only the approximate fitness is known. As the approximation may naturally be inaccurate, it can affect the quality of knowledge transfer negatively. On the other hand, if the set of high-quality source individuals contain a substantial number of duplicates then it is likely that not enough high-quality unique individuals would remain to be transferred after removing the duplicates from the set of source individuals.

Therefore, we develop a new Surrogate-assisted Unique Full Tree transfer algorithm, SUFullTree, that also learns a surrogate model from the transferred source individuals. For solving the target problem, SUFullTree also forms a large initial pool of *unique* individuals but unlike SAKGPHH, it selects into the pool the source individuals that are better than a threshold fitness and fills the rest of the pool with individuals that are mutated from the already selected better-than-threshold individuals. The motivation for mutating the good source individuals is that if the source and target problems are related, it is very likely that the optima of the target problem is similar to the optima of the source problem, that is, the target optima reside in the vicinity of the optima of the source problem. Therefore, mutating the high-quality source solutions increases the chance of finding the target optima. Additionally, if the search process is stuck in the local optima of the source problem, mutation will increase the chance of escaping the local optima trap. After creating the large pool, the learned surrogate model can be utilised to estimate the fitness of the individuals in the pool and select those with the best estimated fitness values to initialise GPHH for solving the target problem. The details of this algorithm are explained in 4.4.3.

It should be noted that surrogate models have been used for the pur-

pose of diversity promotion [105, 178], approximating individual fitness in multi-task optimisation scenarios [141, 248]. Also, transfer learning has been used for augmenting the quality of the surrogate models [231]. However, to the best of our knowledge, this is the first effort that utilises surrogate models for improving the quality of the transferred knowlege. As a result, in the work in this chapter, surrogate models are not used after the initialisation for in way.

4.2 Chapter Goal

In this chapter we aim to propose two novel *approaches to knowledge transfer for GPHH to handling duplicates in the set of transferred source individuals. The first approach tries to overcome any issues that may arise after the transfer of duplicates. The second approach removes the duplicates from the knowledge source and then utilise these individuals to create a diverse and high quality initial population for GPHH.* For this, we examine the performance of the proposed algorithms on a large set of experiments and compare their performance against GPHH without any knowledge transfer, as well as, a collection of existing algorithms. More specifically, we will achieve the following research objectives in this chapter:

- Propose two approaches to handling the presence of duplicates in the transferred knowledge source.
- Propose the a new approach to overcoming any issues that may occur after the transfer of duplicates.
- Propose the novel approach of using transferred knowledge for training a surrogate model and utilise the learned surrogate model for creating a diverse set of high-quality unique individuals for initialising GPHH for solving the target problem.

- Validate the effectiveness of the proposed algorithms on a large set of experimental transfer scenarios.

4.3 Chapter Organisation

This chapter is organised as follows. Detailed descriptions of the proposed algorithms are given in Section 4.4. The experiment design is shown in Section 4.5, followed by results and discussions in Section 4.5.2. Finally, Section 4.7 concludes this chapter.

4.4 Proposed Algorithms

In this section, first we introduce the Diversity-Driven GPPHH. The Surrogate-Assisted Knowledge transfer algorithm for GPPHH, SAKGPHH, is introduced in Section 4.4.2 and then in Section 4.4.3, the Surrogate-assisted Unique Full Tree transfer algorithm, SUFullTree, will be introduced.

4.4.1 Diversity-Driven Transfer of Individuals

As was reviewed in Chapter 2, there exist different approaches for extracting knowledge from a source problem to be used for solving a target problem. In this regard, the general approach is to select the individuals in the GP population (usually the final population) as the knowledge source.

Evolutionary algorithms typically endeavour to explore the search space of a problem, discover the regions that potentially contain the good solutions and, exploit the information they find in these regions to discover the global optima. In the case of GP, the crossover operator is generally considered the main tool for the exploitation phase since it creates new solutions from the genetic information of two existing individuals. The mutation operator, on the other hand, inserts random genetic materials into individuals and hence, is considered to play a major role in exploration.

However, the initialisation operator also has an important yet subtle role. When the GP population is initialised randomly, it can be considered as an act of exploration. However, when transferred individuals are a related solved problem, the operator can be considered to be participating in the exploitation phase. The main reason for this is that by transferring some information about the good regions of the search space that are potentially good for the target problem, the initialisation biases the search to exploit the transferred regions. Therefore, knowledge transfer biases the search towards exploitation.

However, this can lead to a potential issue. Due to the convergence of the GP search to good solutions of the source problem, the population may contain a low level of diversity because it may contain many individuals that are identical or very similar from either genotypic or phenotypic perspective. Additionally, if the search process was stuck in local optima of the source problem, transferring the final individuals could also possibly put GP in a local optima of the target problem.

Consequently, to increase the exploration capability of GP after the transfer of individuals from a source population, we propose to let GP explore the search space more freely. This also helps GP recover from any potential local optima that have been transferred with the transfer of source individuals. A straightforward way to increase the exploration is to increase the mutation rate (and decrease the crossover rate) after knowledge transfer. In this regard, it should be realised that, setting the mutation rate to a fixed high rate can be harmful for the search process because it interferes with its exploitation of the promising regions that it discovers through exploration.

Therefore, in this section, we propose an adaptive mutation rate for the GP for solving the target problem, after the population is initialised with the source individuals. For this purpose, we set the mutation rate to an initial high value $p_m(0)$ and update the rate at each generation t based on one of the following strategies:

- **Exponential**: The mutation rate is updated with an adaptation rate of $0 < \gamma < 1$ with $p_m(t+1) = p_m(t)\gamma^t$. This strategy allows GP to have a high degree of exploration in early generation but the rate is dropped quickly to prevent excessive exploration that could hurt the search process.

- **Power**: The mutation rate is calculated with $p_m(t+1) = P_m(0)/(1+t)^{0.5}$. This strategy is similar to the **exponential** strategy but it decreases the mutation rate more slowly.

- **Cosine**: The mutation rate is updated as $p_m(t+1) = p_m(0)\cos^2(\pi t/t_{max})$ wherein t_{max} is the maximum number of generations that GP is configured to run. Similar to the previous strategies, this strategy also starts with a high mutation rate, which is reduced to a low level at generation $t_{max}/2$. After this, the mutation rate is increased again. This allows GP to explore more in early generations, then focus more on exploitation in later generations, do more exploration to escape from any potential local optima that it may have fallen into.

Furthermore, we consider a minimum threshold mutation rate δ_m so that if the mutation strategy calculates a value below this rate, then the threshold is used instead. This threshold value ensures that at any stage, a minimum level of exploration is guaranteed. After the mutation rate is calculated, the crossover rate $p_c(t)$ is adjusted as $p_c(t) = 1 - p_m(t) - p_r$. In this equation, p_r is the reproduction rate which always has a constant value in our approach. Figure 4.1 depicts the mutation probability that GP will have at each generation based on each of the defined strategies for $p_0 = 0.95$ and $\gamma = 0.95$.

We refer to our proposed approach as Diversity-Driven GPHH, DDGPH in general and based on the utilised mutation strategy, DDGP-Exp, DDGP-Pow and DDGP-Cos refer to DDGP with the exponential, power and cosine strategies respectively.

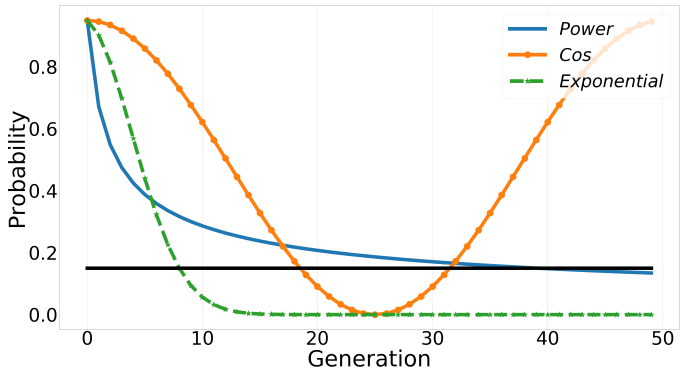


Fig. 4.1 *Mutation probabilities at each generation for different strategies*

4.4.2 Surrogate-Assisted Knowledge transfer for GPHH

During the course of training routing policies for UCARP, a significant amount of computational cost is incurred by the evaluation of the fitness of routing policies due to the fact that it requires running a complete simulation of vehicles serving the tasks in the problem environment. UCARP is not the only problem that faces this issue and it can be encountered in many other problem domains [217]. Surrogate models are computational models that can approximate the real fitness function with a substantially lower computational cost. However, as these models are intended to be approximations of the real fitness function, they are not as accurate. Nevertheless, these models can help judge how good a potential solution can be [120]. To approximate the actual fitness function, many surrogate models utilise machine learning techniques to learn the characteristics of the fitness function [217]. Amongst the available machine learning method, K-Nearest Neighbourhood, KNN, has proven to be a successful method in the context of evolutionary computation methods for solving combinatorial optimisation problems [105, 30, 248].

Algorithm 4.1 presents the proposed Surrogate-Assisted Knowledge Transfer for GP, SAKTGP. The algorithm utilises the transferred surrogate model to create a high-quality and diverse initial population for GP to

Algorithm 4.1: Pseudocode for SAKTGP

Input : k, the magnitude of the interim population,
$\mathcal{S}$, A pool of transferred individuals from a source problem.
$PopSize$, Population size

Output: The best solution ind^*

1: $k \leftarrow 0$, $ind^* \leftarrow null$
`// Load a surrogate from source domain, Algorithm 4.5.`
2: $\Im \leftarrow$ `LoadSurrogate` $(\mathcal{S})$
`// Create` $k \times PopSize$ `individuals`
3: $i \leftarrow 1$, $P \leftarrow \{\}$
4: **for** $i \leq k$ **do**
5: Create a random population $\mathcal{P}'$
6: Evaluate P' with $\Im$
7: $P \leftarrow P \cup P'$
8: $P \leftarrow$ `RemoveDuplicates` (P)
9: **if** $i = k$ *and* $|P| < k \times PopSize$ **then**
10: $i \leftarrow i - 1$
11: **end**
12: **end**
13: $P \leftarrow$ `SortFitness` (P)
14: $Pop \leftarrow$ Select top $PopSize$ of P
15: **while** *not stop* **do**
16: Evaluate each individual in Pop
17: $ind^* \leftarrow$ Best individual in Pop
18: Apply crossover, mutation and reproduction to Pop
19: **end**
20: **return** ind^*

solve the target problem. For this purpose, the algorithm first creates a surrogate model with the transferred individuals (line 2). Initialising an empty set of unique random individuals P (line 3), a population of ran-

dom individuals is created (line 5) which is evaluated with the surrogate (line 6). The population is then added to the pool of unique individuals P and then, any duplicates are removed from it (lines 7–8). This process is repeated for k times or until enough individuals exist in P. Finally, the individuals in the pool P are sorted based on their estimated fitness (line 13) and the individuals with the best estimated fitness are selected as the initial GP population (line 14). After initialising the population, a standard GP is utilised with standard breeding operators of crossover, mutation and reproduction to evolve the population until some stopping criteria is met (lines 16–19).

Phenotypic Characterisation of Routing Policies

To measure the distance between individuals, KNN needs a similarity measure. Although the similarity can be measured based on genotypic characteristics of the GP individuals, Hildebrandt et al. [105] showed that relying on phenotypic properties of individuals can capture the similarities better than the genotypic properties. In contrast to the genotypic similarity that measures how similar the genetic materials of two routing policies are, the phenotypic similarity measures how similar two policies behave in the same situation. In the context of UCARP, this means how similar the decisions of vehicles are when they are idle and face the same set of tasks to select one from.

To be more specific, we define a decision-making situation (DMS) Ω as the situation in which a vehicle is idle and faces a set of unserved tasks from which it needs to select one to serve next. In a DMS, the vehicle employs its routing policy ρ to calculate the priority of each unserved task and rank them based on their priority. Consequently, if a fixed indexing is defined for the tasks in Ω, we define $\zeta_\Omega(\rho)$, the phenotypic characterisation of ρ in situation Ω, as the index of the selected task. Given a fixed set of decision-making situations $\mathbf{\Omega}$, we define $\zeta(\rho)$, the phenotypic characterisation of ρ, as the integer vector that contains the phenotypic characterisa-

Algorithm 4.2: `Characterise` (ρ, $\mathbf{\Omega}$)

Input: A routing policy ρ; a set of decision situations $\mathbf{\Omega}$
Output: A numeric characteristic vector $\boldsymbol{\zeta}$

```
1 foreach Ω_i ∈ Ω do
2     r ← Rank (ρ, Ω_i)
      // Get the index of the task with the highest priority
3     j ← Index (r)
4     ζ_i ← j
5 end
6 return ζ
```

tion of all the decision situations $\Omega \in \zeta_\Omega(\rho)$. Having two routing policies ρ and ρ', we define the distance between them as

$$\Delta(\rho, \rho') = \sum_{\Omega \in \mathbf{\Omega}} \delta(\zeta_\Omega(\rho), \zeta_\Omega(\rho')) \tag{4.1}$$

where

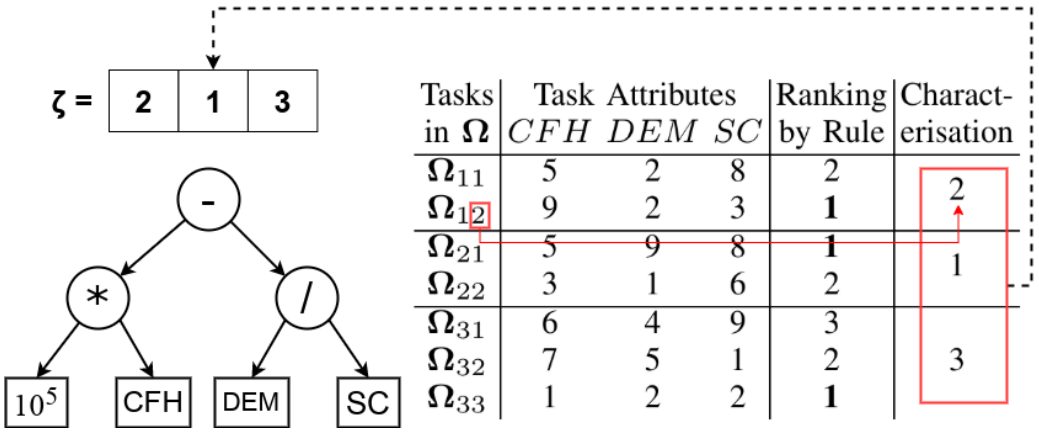
$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

Based on the findings in the work by Hildebrandt et. al. [105], the decision situations can be selected randomly from the set of situations that vehicles encounter during a GP run. Although these situations can be updated in each generation, Hildebrandt et. al. found that doing so does not provide significant performance improvement to selecting the situtations in the first generation and keeping them fixed. Therefore, we also adopted this approach.

Algorithm 4.2 describes the phenotypic characterisation process of a routing policy ρ based on a set of decision situations $\mathbf{\Omega}$. For each $\mathbf{\Omega}_i \in \mathbf{\Omega}$, it first calculates the priority of each candidate task and ranks them based on their priorities (line 2) . Then, it obtains the index of the top-rank task in the candidate task list (line 3) in the set of unserved tasks in $\mathbf{\Omega}_i$.

As an example, Figure 4.2 demonstrates the phenotypic characterisation process.The figure depicts the tree representation of the well-known path-scanning heuristic [125] as 10^5 * *CFH* - *DEM* / *SC*, where *CFH* indicates the cost from the vehicle's current location to the candidate task, *DEM* is the demand of the task, and *SC* is its serving cost (the coefficient 10^5 of *CFH* is a sufficiently large number so that smaller *CFH* is always preferred). This example contains 3 decision situations. The first decision situation has two candidate tasks (Ω_{11} and Ω_{12}), the second has two (Ω_{21} and Ω_{22}) and the third contains three (Ω_{31}, Ω_{32} and Ω_{33}). Based on the attribute values of the tasks, we can see that the indices of the tasks selected by the routing policy in the three decision situations are 2, 1, and 3, respectively. This forms the phenotypic behaviour vector $[2, 1, 3]$.

In this direction, two policies ρ and q are considered the same if they select the same tasks for all decision situations (and hence $\Delta(\rho, \rho') = 0$). It should be noted that since we defined Eq. (4.1) based on task indexes, when $\Delta(\rho, \rho') \neq 0$ the calculated distance is not accurate (its accuracy may vary depending on the number of same tasks that the policies select). However, in this work this is not problematic because we are only



Tasks in Ω	Task Attributes CFH	DEM	SC	Ranking by Rule	Characterisation
Ω_{11}	5	2	8	2	
Ω_{12}	9	2	3	**1**	2
Ω_{21}	5	9	8	**1**	
Ω_{22}	3	1	6	2	1
Ω_{31}	6	4	9	3	
Ω_{32}	7	5	1	2	3
Ω_{33}	1	2	2	**1**	

Fig. 4.2 *The tree representation of the* $10^5 * CFH - DEM/SC$ *path-scanning heuristic and its phenotypic characterisation.*

interested in cases where $\Delta(\rho, \rho') = 0$.

Removal of Phenotypically Duplicate Routing Policies

Having the definition of phenotipic characterisation and the phenotypic distance between routing policies in Eq. 4.1, it is possible to check if two routing policies are identical phenotypically or not. Algorithm 4.3 presents our approach to clearing duplicates from a set of routing policies P. Receiving P as input, the output of the algorithm is the subset of unique individuals in P. The algorithm utilises the idea of hashing the phenotypic vectors into a unique integer. The hash value, in turn, allows storing the individuals into a hash table data structure [50]. The important and helpful feature of the hash table data structure is that it allows querying its content in the constant time complexity of $O(1)$. This feature removes the need for iterating over P twice for checking if each of its individuals is duplicate or not.

When using the hash table data structure, it is important to possess a hashing technique that provides a reasonable guarantee against collision (i.e. having the same hash value for different vectors). In our work, the simple left-shifting method in Algorithm 4.4 proved to be effective as its hash values of the characterisation vectors did not colide. The Algorithm presents our proposed method for hashing a routing policy ρ. Given ρ as input, the algorithm first calculates ζ, the phenotypic vector of ρ, (line 2) and then the hash value is computed iteratively based on the components of ζ with the left-shifting formula in line 5 in which the $<<$ operator presents the left bit-shift operator. The `Hash` function has a time complexity of $O(|\zeta|)$, which depends on the number of decision situations used to characterise the phenotypic behaviour.

After defining the hashing procedure for routing policies in Algorithm 4.4, it is possible to utilise it for removing duplicates, as is presented in Algorithm 4.3. More specifically, this algorithm starts by initialising an empty hash table $\mathbf{\Psi}$ (line 2) and then, it sorts the individuals in P based on

Algorithm 4.3: `RemoveDuplicates`(P)

Input: A set of individuals P
Output: A set of unique individuals $P' \subset P$

```
1  begin
2      Ψ ← ∅
3      P ← SortFitness (P)
4      foreach ρ ∈ P do
5          h_ρ ← Hash (ρ) // See Algorithm 4.4
6          if h_ρ ∉ Ψ then
7              Ψ[h_ρ] ← ρ
8          end
9      end
10     return Stored individuals in Ψ
11 end
```

their fitness in ascending order so the best individual with lowest fitness is first (line 3). Then, for each individual in $\rho \in P$ the algorithm calculates h_ρ, the hash value of ρ based on its phenotypic characterisation using the hashing method defined in Algorithm 4.4 (line 5). Then, $\mathbf{\Psi}$ is checked to see if it contains h_ρ or not (line 6). If the hash table does not contain h_ρ, ρ is recorded in $\mathbf{\Psi}$ (line 7) so that any individual with the same phenotypic characterisation and hash value will be detected as duplicate. Finally, after iterating over all items in P, the set of individuals in $\mathbf{\Psi}$ are returned the set of unique routing policies in P.

KNN-based Surrogate Models for UCARP

A KNN-based surrogate model estimates the fitness value of a given individual ind based on a pool of individuals whose fitness is already known. It first measures the distance of ind from each of individuals in the KNN pool. Then k nearest individuals in the KNN pool are considered and their average fitness is returned as the approximate fitness of ind. Algorithm 4.5 presents the pseudo code of a basic KNN-based surrogate model.

Algorithm 4.4: `Hash`(ρ)

Input: A routing policy ρ
Output: A unique hash value for ρ

```
1 begin
2     ζ ← ζ(ρ)
3     h ← 1
4     for i = 1 → |ζ| do
5         h ← h << 5 − h + ζ_i
6     end
7 end
8 return h
```

Algorithm 4.5: Pseudocode for a KNN-based surrogate

Input : A pool Ψ of individuals with known fitness values
Neighborhood size k
Individual i to estimate the fitness of
Output: Estimated fitness for i

```
1: Ψ ← removeDuplicates (Ψ)
2: foreach ψ ∈ Ψ do
3:     Calculate the distance between i and ψ
4: end
5: Ψ′ ← k individuals in Ψ nearest to i
6: φ(i) ← average fitness of items in Ψ′
7: return φ(i)
```

The set of individuals that GP created during the search process for solving the source problem can be utilised as the KNN pool for building a KNN-based surrogate model to estimate the fitness of individuals with significantly less computational cost for the target problem. This surrogate model is constructed from the source individuals and hence, is better suited for the source problem. However, the basic and fundamental axiom in transfer optimisation is that the source and target problems are related, therefore, good (bad) solutions of the source problem are expected

to have good (bad) performance for the target problem. Since the surrogate model is constructed for the source problem, it is expected to detect good (bad) source individuals which then can be expected to be relatively good (bad) for the target problem. For the act of knowledge transfer for initialising GP, this level of assurance in the performance of the surrogate model, which are innately expected to be inaccurate, is acceptable because creating individuals that are moderately better than random can help GP start its search from a potentially good state.

The learned/transferred surrogate model empowers GP to evaluate a large number of individuals with a low computational overhead to create a good initial state. This indicates an advantage over the methods that simply transfer source genetic materials without taking any measures to reassure the quality of transferred individuals [129, 62, 100]. In Chapter 3, we discussed how the presence of duplicates and redundancies in the set of transferred individuals can have a drastic negative impact on the effectiveness of knowledge transfer for UCARP. There, apart from the potential quality of the individuals, the phenotypic diversity of the initial population is our next second criterion for the design of our algorithm. For this, enforce the initial GP population for solving the target problem to have no duplicates.

4.4.3 Surrogate-assisted Unique Full Tree transfer algorithm

The SAKGPHH algorithm that we described in Section 4.4.2 initialises a diverse initial population of random individuals with the help of a surrogate model that is learned from the knowledge that is tranferred from a solved source problem. However, this algorithm discards the actual source individuals whose fitness value is known in favour of randomly created individuals whose fitness is only approximated with the surrogate model. This means that valueble information could be lost due to the inaccurate nature of surrogate models. In this section, we propose an approach to

overcome this problem.

Similar to SAKGPHH, our proposed algorithm Surrogate-Assisted Genetic Programming with Diverse Transfer for the Uncertain Capacitated Arc Routing Problem, SUFullTree, also creates an initial population of diverse individuals of a high-quality. For this, the algorithm also utilises a surrogate model to estimate the fitness of the initial individuals. However, unlike SAKGPHH, the individuals are not created randomly but are mutated from the high-quality individuals that are transferred from the source problem. By putting an emphasis on creating diverse and unique individuals, creating individuals with mutation allows SUFullTree to create a set of high-quality individuals that are in the vicinity of the good source individuals, which are likely to be better than random solutions for the target problem. This increases the likelihood of finding more high-quality solutions.

SUFullTree is presented in Algorithm 4.6. Given a set of transferred source individuals $\mathcal{S}$ and a threshold fitness value $\tau \in [0, 1]$, the algorithm uses the individuals in $\mathcal{S}$ whose normalised fitness is better than τ to initialises a duplicate-free initial state for GPHH to solve a related target problem. SUFullTree first uses the transferred individuals to load a KNN-based surrogate model (line 1), utilises the duplicate removal method in Algorithm 4.3 (line 2)then, sorts the remaining unique individuals based on their fitness (line 2). To have a unified fitness range, SUFullTree normalises the fitness values of the individuals into the range $[0, 1]$ so that the best individual has a fitness value of 0 (line 3). Having the fitness values of the transferred individuals normalised, SUFullTree then selects the individuals whose fitness are below the threshold τ as the best of the source individuals that will be considered as the reusable transferred individuals and other individuals will be discarded (line 3).

After selecting the best of the source individuals, they are used for creating a large diverse set of high-quality individuals from which the best will be used to initialise GPHH for solving the target problem. SUFull-

Algorithm 4.6: SUFullTree, Surrogate-Assisted Unique Full Tree Transfer Learning for initialising GP of the target domain

Input : A pool $\mathcal{S}$ of transferred individuals from a source problem
Fitness threshold τ

Output: Initial population for GP of the target domain

`// Load a surrogate from source domain, Alg. 4.5.`

$\Im \leftarrow$ `LoadSurrogate` $(\mathcal{S})$
$P \leftarrow$ `RemoveDuplicates` $(\mathcal{S})$, $P \leftarrow$ `SortFitness` $(\mathcal{S})$
$P' \leftarrow$ `NormaliseFitness` $(\mathcal{S})$, $P' \leftarrow$ `SelectBest` (P', τ)
$\Psi \leftarrow \emptyset$
for $i = 1, ..., |P'|$ **do**
    $h \leftarrow$ `Hash` $(P'[i])$ `// See Alg. 4.4`
    $\Psi[h] \leftarrow P'[i]$
end
while $|\Psi| < 10 \times PopSize$ **do**
    foreach $\rho \in P'$ **do**
        $\rho' \leftarrow$ `Mutate` (ρ) `// Standard GP mutation operator`
        $h \leftarrow$ `Hash` (ρ') `// See Alg. 4.4`
        if $h \notin \Psi$ **then**
            `// Surrogate evaluation. See Alg. 4.5`
            `Fit` $(\rho') \leftarrow \Im(\mathcal{S}, \rho')$
            $\Psi[h] \leftarrow \rho'$
        end
    end
end
$Pop \leftarrow$ Select top $PopSize$ of Ψ
while *not stop* **do**
    Evaluate each individual in Pop
    $ind^* \leftarrow$ Best individual in Pop
    Apply crossover, mutation and reproduction to Pop
end
return ind^*

Tree utilises a hash table data structure to store the large pool of individuals that it creates to benefit from its $O(1)$ query time complexity. To do so, SUFullTree first initialises an empty table (line 4), employs the hashing mechanism in Algorithm 4.4 to compute the hash value of each good transferred individual (line 6) and store then in the table.

When the knowledge source contains many duplicates, it is likely that after removing the duplicates, the number of high-quality individuals that remain are less than what is needed to initialise GPHH. To overcome this, SUFullTree mutates the selected transferred individuals and evaluates the mutation offspring with the learned surrogate model to estimate their fitness. Instead of creating just enough individuals to fill the initial population, SUFullTree creates a large pool of individual to increase the chance of finding good solutions. In this work, the size of the pool of mutated individuals is set to be 10 times the population size. Accordingly, for each good transferred individual ρ', SUFullTree utilises the standard mutation operator to create a new individual ρ' (line 11), uses the hash algorithm in 4.4 to calculate its hash (line 12), and checks if the hash value is inside the hash table (line 13). If the individual is not in the hash table, then the transferred surrogate model is utilised to estimate the fitness of ρ' (line 14) before adding it to the hash table (line 15). Finally, after the pool is created, the best individuals in it are selected to initialise GP. After initialisation, the standard GP search mechanism is utilised to evolve the individuals and obtain the best routing for the target problem (lines 21–24).

4.5 Experimental Studies

The effectiveness of the proposed algorithms are evaluated in this chapter with a set of simulated experiments on the scenarios in Table 3.1 (page 73). For the sake of consistency, we also use the same GP settings in our previous experiments, as is given in Table 3.2.

4.5.1 Experiment Settings

Similar to experiments in Chapter 3, first we solve the source problem with the vanilla GP that evolves 1024 individuals for 50 generations. Then, the individuals, i.e., GP trees, the are considered as the transferable knowledge. Of the three proposed algorithms in this chapter, DDGP only focuses on the final source population while the rest of the algorithms consider all the source populations as the knowledge source.

The proposed SAKTGP and SUFullTree algorithms rely on phenotypic characterisation of routing policies based on a set of decision situations $\mathbf{\Omega}$. To have a set of decision situations, the path-scanning policy in Figure 4.2 was utilised to serve a set of tasks for each scenario and 20 of the encountered situations were selected as $\mathbf{\Omega}$. Using a small number of situations can reduce the accuracy of the characterisation. However, a large $\mathbf{\Phi}$ also increases the computational cost. In our experiments, the size of 20 demonstrated a good balance between accuracy and computational cost.

Both SATKGP and SUFullTree create an initial interim pool of unique GP individuals that are evaluated with the transferred surrogate. In our experiments, we tried different sizes for the interim pool and noticed that while smaller population sizes tend to have lower performances, we did not observe any increase in performance beyond the size 10×1024.

In Section 3.5, a set of existing transfer optimisation algorithms were evaluated for solving UCARP, from which BestGen-1 and GATL demonstrated the performances. Accordingly, in this section, these algorithms are selected from the existing literature of transfer optimisation methods and the performance of the proposed algorithms are compared against them.

The significance of the results in this section is examined with the Friedman test with a confidence level of $\alpha = 0.05$. Furthermore, whenever the Friedman test indicates the existence of significant difference in the results, the Conover post-hoc analysis [49] is performed on the results to pinpoint which algorithms have shown different performances and the p-values of the pairwise comparisons are adjusted with the Benjamini-Hochberg

method [25].

4.5.2 Performance of DDGP

Table 4.1 presents the test performance of 30 independent DDGP with different mutation strategies, as well as a set of existing methods. The table also includes the rank of each algorithm and also the p-value of the Friedman test.

Table 4.1: Test performance of 30 independent runs of the DDGP algorithms (mean $\pm$ std)

Scn.	GPHH	GATL [129]	BestGen-1 [62]	DDGP-Cos	DDGP-Exp	DDGP-Pow
1	91.2±0.1	91.2±0.1	**91.2±0.1**	91.2±0.1	91.2±0.1	91.2±0.1
2	551.0±10.3	**550.8±8.1**	551.1±10.4	555.4±10.8	552.1±8.3	555.5±9.0
3	**598.6±8.8**	599.6±9.5	600.5±11.6	603.9±9.7	602.3±12.5	601.4±8.4
4	639.5±11.3	**636.0±10.7**	640.6±12.1	643.3±10.4	640.4±10.4	644.6±13.1
5	58.2±0.1	58.2±0.1	58.3±0.1	58.2±0.1	**58.2±0.1**	58.3±0.1
6	424.8±8.5	424.9±8.8	423.9±8.6	423.8±7.5	425.3±6.1	**422.2±7.4**
7	432.1±7.1	**430.0±6.3**	431.2±6.3	430.6±8.6	430.6±7.6	431.2±5.4
8	432.6±5.5	**430.6±6.7**	432.7±4.8	431.0±5.3	431.4±5.5	432.3±5.0
9	576.2±3.9	**575.8±4.2**	576.8±3.7	576.2±4.7	576.0±3.9	576.2±3.6
10	340.5±4.7	338.1±4.2	**337.5±3.1**	337.9±3.5	338.0±2.6	338.1±2.5
11	347.2±6.1	347.1±6.0	**345.9±4.8**	346.9±5.6	346.7±5.7	348.3±6.2
12	**551.0±10.3**	553.7±10.5	551.8±10.1	555.8±9.0	553.2±11.4	555.1±9.2
13	598.6±8.8	598.5±7.5	**597.6±8.2**	603.9±15.1	604.6±11.2	604.1±13.0
14	639.5±11.3	640.1±12.2	**639.0±11.9**	642.8±17.0	640.5±15.4	643.9±20.9
15	340.5±4.7	339.8±3.5	339.9±5.0	**337.2±4.4**	338.5±3.6	338.8±3.3
16	444.4±4.7	445.0±7.6	**443.9±6.5**	447.7±7.0	445.4±6.4	445.6±7.7
17	324.3±6.2	323.3±5.2	**321.5±5.4**	323.5±5.7	323.8±5.7	324.8±5.3
18	360.3±3.1	359.7±3.7	359.4±3.9	359.2±3.8	**358.4±4.4**	359.0±4.0
19	358.3±2.6	**358.3±3.1**	358.8±2.7	361.5±6.7	362.0±7.1	361.8±6.4
20	359.0±1.8	**358.5±1.8**	358.6±1.9	359.0±1.6	358.9±1.7	359.5±1.8
21	340.8±4.4	340.8±2.2	**339.4±4.7**	339.9±3.5	340.8±3.3	340.6±2.8
22	351.9±3.5	**351.6±2.5**	352.5±3.5	352.6±3.7	351.7±3.0	353.2±4.0

Continuation of Table 4.1

Scn.	GPHH	GATL [129]	BestGen-1 [62]	DDGP-Cos	DDGP-Exp	DDGP-Pow
23	356.6±1.6	356.4±1.5	356.1±1.3	356.9±1.8	**356.0±1.1**	356.4±1.5
24	310.9±0.5	311.0±0.3	310.7±0.8	310.0±1.6	**310.0±1.5**	310.2±1.2
25	389.2±0.2	**389.1±0.2**	389.2±0.2	389.1±0.2	389.2±0.2	389.2±0.2
26	363.1±2.8	363.1±3.2	363.4±2.6	363.3±3.0	**361.2±4.6**	362.9±3.8
27	342.1±6.2	342.5±7.8	**340.9±8.0**	341.8±7.0	343.3±5.9	341.0±5.4
28	382.0±5.5	**381.0±4.6**	381.3±8.0	385.1±7.0	384.0±9.3	384.7±7.8
29	382.8±3.3	383.9±2.6	**382.7±4.8**	383.2±4.3	383.0±5.2	384.5±4.9
30	351.5±2.5	351.1±2.2	351.7±1.2	**351.0±1.5**	351.2±2.0	351.2±1.2
31	326.0±4.7	325.2±4.0	325.2±5.0	325.6±4.3	324.5±5.4	**324.3±4.8**
32	444.4±4.7	443.7±5.6	**442.0±7.3**	444.1±7.6	443.5±7.6	444.7±8.2
33	448.2±0.5	449.0±2.3	**448.2±0.9**	449.6±3.5	449.4±2.6	449.8±2.9
34	**384.6±4.4**	387.1±6.0	386.9±5.0	387.8±5.9	389.2±5.5	387.4±5.2
35	369.3±1.8	**369.3±2.2**	369.8±3.8	370.3±4.5	371.3±4.1	370.1±3.1
36	**321.4±5.2**	322.7±4.2	323.8±5.1	323.2±5.0	322.9±5.5	324.3±5.6
37	166.2±2.0	166.1±1.7	**165.2±1.5**	165.5±1.7	165.9±1.6	165.6±1.8
38	**376.1±7.6**	381.2±7.0	377.8±7.5	378.4±8.0	378.8±7.5	379.8±8.0
39	415.7±9.2	415.5±7.2	**414.3±8.9**	416.0±7.9	414.8±8.0	416.3±6.7
40	347.2±6.1	347.5±5.2	**345.8±4.4**	347.7±6.3	348.8±8.2	346.8±6.2
41	351.5±2.5	**351.4±2.7**	352.0±2.3	351.8±2.7	352.6±4.9	352.6±5.3
42	165.9±1.8	165.6±1.7	165.7±1.6	165.5±1.7	**165.3±1.9**	165.5±1.8
43	462.6±6.0	457.4±7.0	460.2±5.4	460.1±6.5	459.6±5.7	**457.1±7.0**
44	**426.6±3.3**	427.3±2.0	427.0±2.6	427.7±2.1	427.7±2.2	426.9±3.1
45	499.0±3.9	498.5±4.4	**497.6±4.4**	499.8±3.6	500.0±5.4	500.2±4.9
Rank	3.49	2.91	2.84	3.89	3.54	4.32
Friedman's p-value						**0.001**

As is evident from the table, none of the three variants of DDGP could rank better than GPHH or existing methods. Since the p-value of the test indicate the presence of significant difference between the results, post-hoc analysis of the results are conducted to determine if any of DDGP variants performed significantly worse than GPHH or the existing methods. The

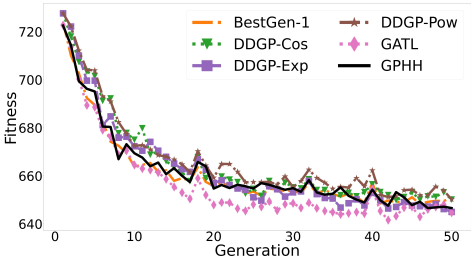
Table 4.2: Adjusted p-value of the pairwise comparison of DDGP algorithms

	GATL	BestGen-1	DDGP-Cos	DDGP-Exp	DDGP-Pow
GPHH	0.20	0.15	0.34	0.85	0.077
GATL	–	0.85	**0.04**	0.15	**0.002**
BestGen-1	–	–	**0.03**	0.12	**0.001**
DDGP-Cos	–	–	–	0.42	0.34
DDGP-Exp	–	–	–	–	0.10

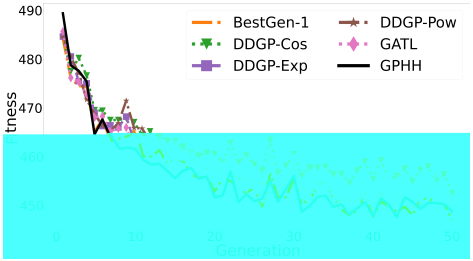
post-hoc analysis is presented in Table 4.2. As is evident from the table, there is no significant difference between the performance of GPHH and any of the DDGP variants. Therefore, although DDGP ranked worse than GPHH in Table 4.1, it is not significantly worse.

Another point to notice from Table 4.2 is that there is no significant difference between any variants of DDGP. Amongst the three variants, DDGP-Exp has shown a rather better performance. On the other hand, considering Tables 4.1 and 4.2, we can note that the performance of GATL and BestGen-1 are significantly better than the performance of DDGP-Cos and DDGP-Pow. However, DDGP-Exp does not perform significantly worse that the existing methods. According to Figure 4.1, the mutation adaptation strategy of DDGP-Exp has the steepest decrease over the course of generation so that for the majority of the evolution, the mutation rate equals the configured threshold value, which is the same as the mutation rate of GPHH without knowledge. On the other hand, DDGP-Cos has the highest overall mutation rates over the course of evolution and our results indicate that this feature has a disruptive effect on the performance of the algorithm.

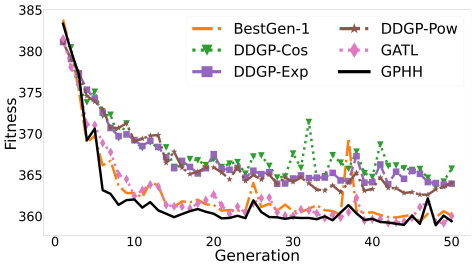
The convergence curve of the algorithms in Figure 4.3 also confirm this observations. Again, it is evident in the Figure that DDGP starts with rather better initial states, e.g. Figure 4.3c, but in later generations, this advantage is lost. The convergence curves also confirm that there is not



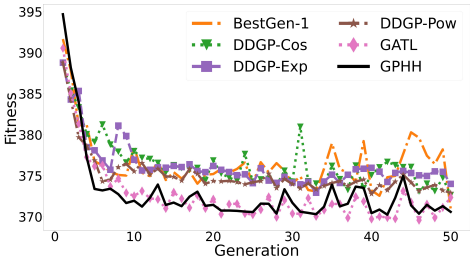
(a) *From Ugdb17 with 5 vehicles to Ugdb12 with 8 vehicles (Scn. 4)*



(b) *From Ugdb21 with 5 vehicles to Ugdb5 with 4 vehicles (Scn. 16)*



(c) *From Ugdb4 with 4 vehicles to Ugdb4 with 6 vehicles (Scn. 19)*

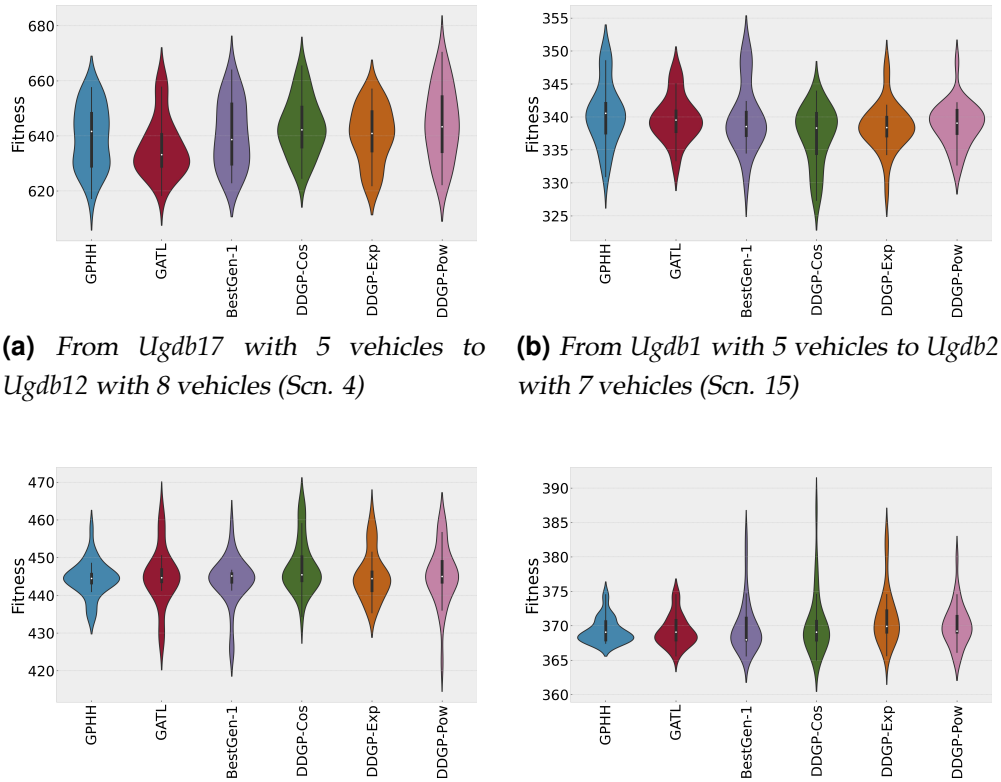


(d) *From Ugdb2 with 6 vehicles to Ugdb2 with 4 vehicles (Scn. 35)*

Fig. 4.3 *The convergence curve of DDGP and some existing knowledge transfer methods.*

any important difference between the three variants of DDGP. However, looking at the plots, it is discernible that in some cases, e.g. Figures 4.3a, 4.3b and 4.3c, DDGP-Cos performs noticeably worse than other methods. This is in line with our previous observation about this algorithm.

Additionally, focusing on the convergence curve of DDGP-Cos, it is evident that performance of the algorithm stops improving midway through the course of evolution. This effect is particularly obvious in Figures 4.3c and 4.3d. The reason for this behaviour can be attributed to the fact that, according to the mutation adaptation strategy of DDGP-Cos, the GP mutation rate starts increasing after generation 25 (the mutation rate is below the threshold value for a few generations after this point). This observation also confirms the negative impact of high mutation rates.



(a) *From Ugdb17 with 5 vehicles to Ugdb12 with 8 vehicles (Scn. 4)*

(b) *From Ugdb1 with 5 vehicles to Ugdb2 with 7 vehicles (Scn. 15)*

(c) *From G21 with 6 vehicles to Ugdb5 with 4 vehicles (Scn. 16)*

(d) *From Ugdb2 with 6 vehicles to Ugdb2 with 4 vehicles (Scn. 35)*

Fig. 4.4 *The distribution of solutions found with DDGP and some existing knowledge transfer methods.*

The distribution of the solutions found with each of the compared algorithms is given in Figure 4.4. This figure also confirms that in general, the solutions obtained with DDPG are not any better than other compared algorithms. Interestingly, the distribution of DDGP-Cos is not very different from other algorithms. DDGP-Cos has the highest overall mutation rate and the violin plots indicate that the increased mutation rate did not have any positive or negative impact.

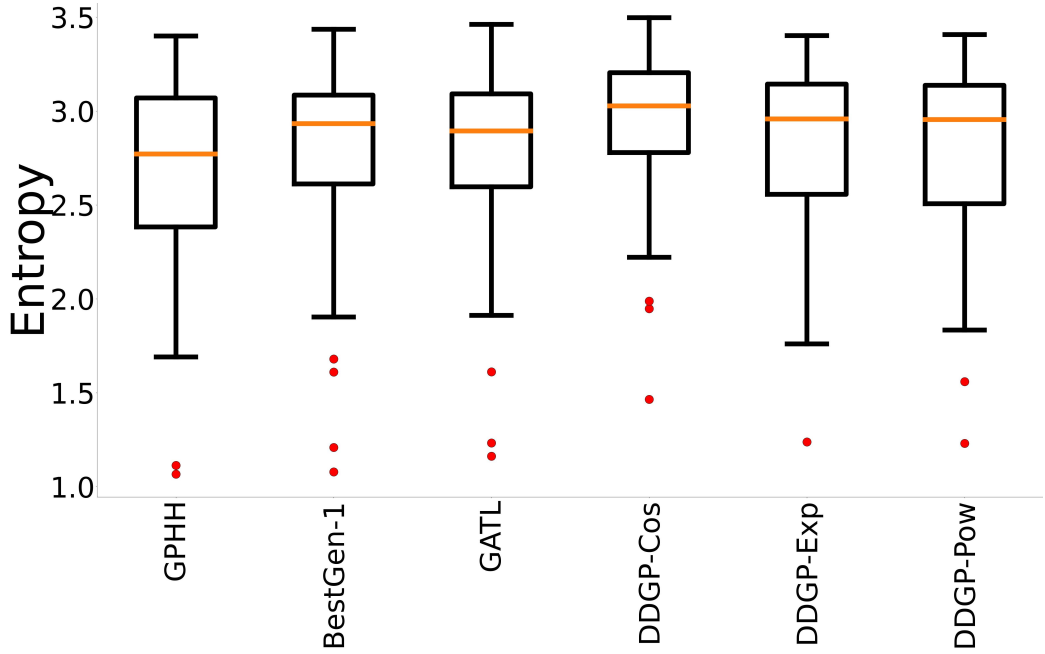
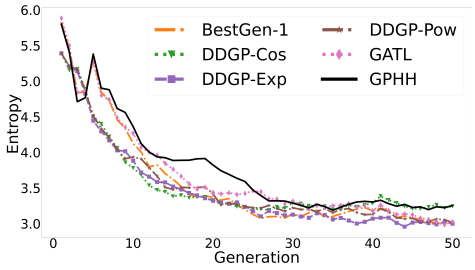


Fig. 4.5 *Average population entropy of DDGP and the compared algorithms.*

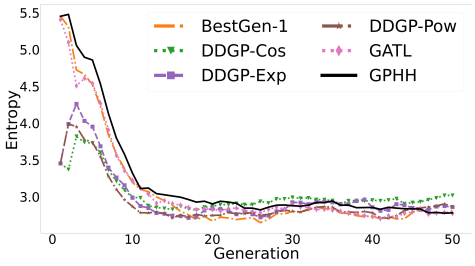
Further Analysis

As the experimental results of DDGP indicated, in contrast to our expectation, increasing the mutation rate after knowledge transfer did not improve the quality of transfer optimisation. Our goal in increasing the mutation rate was to improve the population diversity and overcome the loss of diversity that is inflicted by the transferred individuals. To investigate this, the average population entropy of the algorithms are calculated and depicted in Figure 4.5. As is clearly evident in the plot, the distribution of population entropy of the compared algorithms is very similar. In particular, none of the variants of DDGP could improve the average population entropy significantly. It can be noted that DDGP-GP, which had the highest overall mutation rate, has a slightly better distribution and median entropy but in general, the difference is not very high.

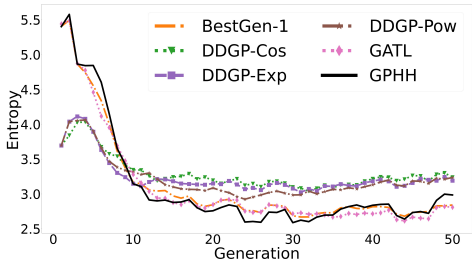
Looking at the change of population entropy over time in Figure 4.6



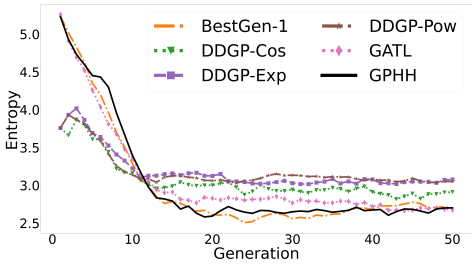
(a) *From Ugdb17 with 5 vehicles to Ugdb12 with 8 vehicles (Scn. 4)*



(b) *From Ugdb7 with 5 vehicles to Ugdb1 with 6 vehicles (Scn. 18)*



(c) *From Ugdb4 with 4 vehicles to Ugdb4 with 6 vehicles (Scn. 19)*



(d) *From Ugdb2 with 6 vehicles to Ugdb2 with 4 vehicles (Scn. 35)*

Fig. 4.6 *The population entropy of DDGP and some existing knowledge transfer methods.*

also reveal more information. As can be seen, all variants of DDGP have a low initial entropy compared to BestGen-1, GATL and GPHH. Over the course of evolution, which is in line with our previous finding in Section 3.5.7 that the transfer of individuals from the final population of a solved source problem can lead to the loss of population diversity. BestGen-1 and GATL, on the other hand, do not have this problem as they select their individuals from all populations.

Over the course of evolution, the population entropy drops for all algorithms. However, having a close look at the plots shows that in some cases, DDGP has a slightly better situation as the population entropy tends to remain slightly better than other methods. This is particularly noticeable in Figure 4.6c for all variants of DDGP. However, amongst the three versions

of DDGP, DDGP-Cos seems to have slightly higher population entropy in later generations. This is due to the fact the DDGP-Cos increases the mutation rate to very high rates in later generations and it is expected to see this behaviour.

As plots in Figure 4.6 confirm, the transfer of final source population can lead to the loss of diversity. To counteract this effect, we proposed increasing the mutation rate to a high value. However, as we observe in Figure 4.6, in our experiments, the increase in mutation rate does not necessarily lead to any substantial increase in population diversity. One reason for this phenomenon can be the fact that the GP process for solving UCARP tends to create many duplicates. This is particularly clear from the entropy curves of GPHH in Figure 4.6. This behaviour is not unexpected as the crossover and mutation operators operate on the genotypic level and it is possible that genotypically different individuals may have the same phenotypic behaviours. Accordingly, it is possible that even the increased mutation rates of DDGP cannot increase the phenotypic diversity. Another reason for the failure to increase population diversity through increasing mutation could be that the loss of diversity in the population is so high that even the high mutation rates cannot overcome it.

In conclusion, our results confirm again that the lack of diversity in the knowledge source can have a very negative impact on the quality of knowledge transfer. Additionally, we note that the results are also indicate that increasing the diversity after the transfer of individuals is not helpful and in some cases, it can also be harmful. In the next section, it will be investigated how handling the presence of duplicates in the knowledge source before transferring individuals can affect the quality of knowledge transfer.

4.6 Performance of Surrogate-based Knowledge Transfer Methods

In this chapter, two transfer optimisation algorithms were introduced for handling the presence of duplicates in the knowledge source. Both of these methods focus on creating an initial population of unique and high-quality individuals for solving the target problems and incorporate surrogates and share important similarities and as a result, these methods are examined together in this section. Since DDGP did not perform any better than existing methods, it is excluded in our comparisons. From the set of existing methods, GATL and BestGen-1 are selected as their performances were better than other existing methods.

Table 4.3 presents the performance of SAKTGP and SUFullTree for solving the designed UCARP scenarios. The last two rows of the table present the rank of each algorithm and the p-value of the Friedman test. As the table indicate, SUFullTree has the best performance and the p-value indicate the existence of significant difference between the results.

Table 4.3: Test performance of 30 independent runs of the SAKTGP and SUFullTree algorithms (mean $\pm$ std)

Scn.	GPHH	GATL [129]	BestGen-1 [62]	SAKTGP	SUFullTree
1	91.2±0.1	91.2±0.1	91.2±0.1	91.2±0.1	**91.2±0.1**
2	551.0±10.3	**550.8±8.1**	551.1±10.4	552.0±8.5	551.5±9.4
3	**598.6±8.8**	599.6±9.5	600.5±11.6	599.1±8.1	601.9±11.0
4	639.5±11.3	**636.0±10.7**	640.6±12.1	643.4±14.1	644.1±15.8
5	58.2±0.1	58.2±0.1	58.3±0.1	**58.2±0.1**	58.2±0.1
6	424.8±8.5	424.9±8.8	423.9±8.6	**419.9±7.8**	423.2±8.6
7	432.1±7.1	**430.0±6.3**	431.2±6.3	430.0±5.2	431.8±8.3
8	432.6±5.5	430.6±6.7	432.7±4.8	431.2±5.6	**430.0±6.1**
9	576.2±3.9	**575.8±4.2**	576.8±3.7	576.8±3.1	576.4±3.3
10	340.5±4.7	338.1±4.2	337.5±3.1	**337.0±4.6**	337.2±2.1

Continuation of Table 4.1

Scn.	GPHH	GATL [129]	BestGen-1 [62]	SAKTGP	SUFullTree
11	347.2±6.1	347.1±6.0	345.9±4.8	**344.4±3.9**	344.4±5.1
12	551.0±10.3	553.7±10.5	551.8±10.1	**550.6±7.7**	552.7±8.9
13	598.6±8.8	598.5±7.5	**597.6±8.2**	598.4±9.2	600.2±10.3
14	639.5±11.3	640.1±12.2	639.0±11.9	**637.3±11.4**	640.8±14.8
15	340.5±4.7	339.8±3.5	339.9±5.0	338.9±3.2	**337.0±3.3**
16	444.4±4.7	445.0±7.6	443.9±6.5	**442.4±5.6**	444.1±4.9
17	324.3±6.2	323.3±5.2	321.5±5.4	321.0±6.3	**319.8±4.6**
18	360.3±3.1	359.7±3.7	359.4±3.9	360.1±3.4	**356.2±4.2**
19	358.3±2.6	358.3±3.1	358.8±2.7	**356.2±2.8**	357.6±5.3
20	359.0±1.8	358.5±1.8	358.6±1.9	**358.0±1.6**	358.3±1.1
21	340.8±4.4	340.8±2.2	339.4±4.7	341.0±2.9	**337.6±4.6**
22	351.9±3.5	351.6±2.5	352.5±3.5	351.0±2.6	**350.2±3.3**
23	356.6±1.6	356.4±1.5	356.1±1.3	**356.0±1.2**	356.3±1.4
24	310.9±0.5	311.0±0.3	310.7±0.8	310.9±0.6	**308.8±2.8**
25	389.2±0.2	389.1±0.2	389.2±0.2	**389.0±0.5**	389.1±0.2
26	363.1±2.8	363.1±3.2	363.4±2.6	**360.9±4.8**	362.2±3.1
27	342.1±6.2	342.5±7.8	340.9±8.0	341.6±6.2	**338.6±4.7**
28	382.0±5.5	**381.0±4.6**	381.3±8.0	381.1±3.7	384.7±6.0
29	382.8±3.3	383.9±2.6	382.7±4.8	382.9±3.8	**382.1±3.2**
30	351.5±2.5	351.1±2.2	351.7±1.2	351.5±1.2	**351.0±1.6**
31	326.0±4.7	325.2±4.0	325.2±5.0	326.3±3.2	**322.8±4.9**
32	444.4±4.7	443.7±5.6	442.0±7.3	445.0±3.7	**441.2±5.8**
33	448.2±0.5	449.0±2.3	448.2±0.9	**448.0±0.5**	448.9±1.8
34	384.6±4.4	387.1±6.0	386.9±5.0	384.4±4.8	**384.3±5.1**
35	369.3±1.8	369.3±2.2	369.8±3.8	**368.4±1.6**	368.4±2.6
36	321.4±5.2	322.7±4.2	323.8±5.1	322.0±4.3	**321.1±2.5**
37	166.2±2.0	166.1±1.7	165.2±1.5	165.7±1.6	**165.1±2.0**
38	376.1±7.6	381.2±7.0	377.8±7.5	373.2±8.2	**372.0±9.2**
39	415.7±9.2	415.5±7.2	414.3±8.9	416.6±5.7	**412.3±7.4**
40	347.2±6.1	347.5±5.2	345.8±4.4	346.3±5.8	**344.5±5.0**
41	351.5±2.5	351.4±2.7	352.0±2.3	**351.0±1.7**	351.2±1.9
42	165.9±1.8	165.6±1.7	165.7±1.6	165.9±1.8	**165.2±1.2**
43	462.6±6.0	457.4±7.0	460.2±5.4	458.6±6.2	**456.7±6.5**

Continuation of Table 4.1

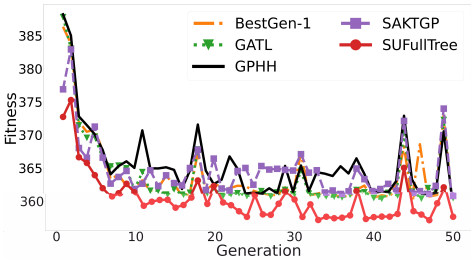
Scn.	GPHH	GATL [129]	BestGen-1 [62]	SAKTGP	SUFullTree
44	426.6±3.3	427.3±2.0	427.0±2.6	426.6±2.9	**425.8±3.8**
45	499.0±3.9	498.5±4.4	497.6±4.4	498.4±5.4	**497.5±3.3**
Rank	3.74	3.43	3.26	2.52	2.04
Friedman's p-value					**0.0012**

The pairwise comparisons of the compared algorithms is presented in Table 4.4. The comparisons indicate that both SAKTGP and SUFullTree are significantly different from GPHH, GATL and BestGen-1. Considering that both SAKTGP and SUFullTree rank better than these existing algorithms, the p-values indicate that these algorithms are significantly better than GPHH, BestGen-1 and GATL. This indicates that the proposed knowledge transfer mechanisms were successful. These results become more prominent when we remember from Section 3.5 that none of the existing algorithms could manage to be significantly better than GPHH without any knowledge transfer.

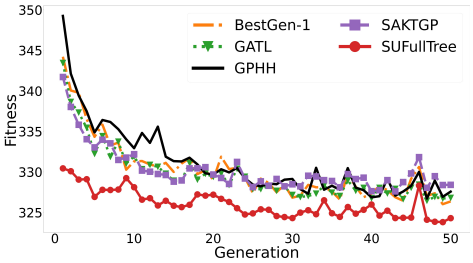
The convergence curve of the algorithms for some transfer scenarios are given in Figure 4.7. One clear observation in the figure is the superior performance of SUFullTree. As can be seen, the algorithm always starts in better initial states, compared to all other algorithms. When comparing

Table 4.4: Adjusted p-value of the pairwise comparison of SAKTGP and SUFullTree algorithms

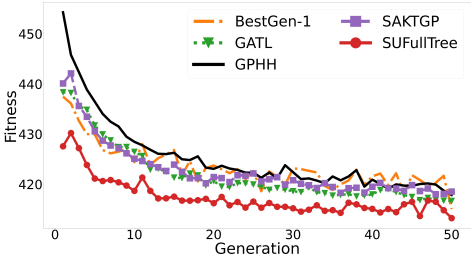
	BestGen-1	GATL	SAKTGP	SUFullTree
GPHH	0.17	0.36	**0**	**0**
BestGen-1	–	0.55	**0.04**	**0**
GATL	–	–	**0.01**	**0**
SAKTGP	–	–	–	0.18



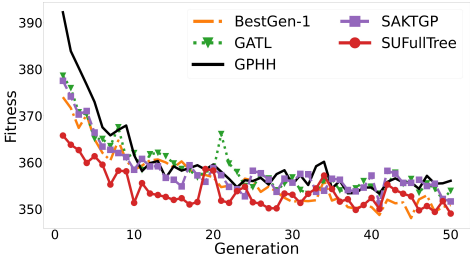
(a) *From Ugdb7 with 5 vehicles to Ugdb1 with 6 vehicles (Scn. 18)*



(b) *From Ugdb3 with 5 vehicles to Ugdb3 with 6 vehicles (Scn. 31)*



(c) *From Ugdb2 with 6 vehicles to Ugdb2 with 8 vehicles (Scn. 39)*



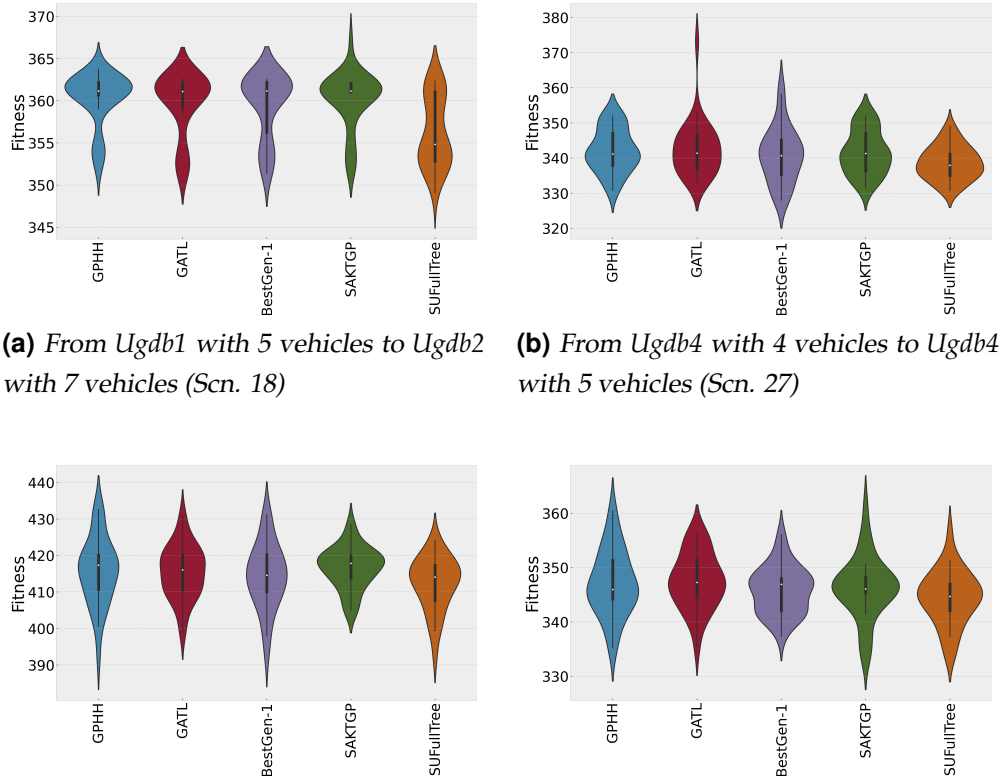
(d) *From Ugdb6 with 5 vehicles to Ugdb6 with 4 vehicles (Scn. 40)*

Fig. 4.7 *Convergence curve of SAKTGP and SUFullTree and some existing knowledge transfer methods.*

with SAKTGP, we note that the quality of initial states is much better than the quality of SAKTGP. Both of these algorithms create an initial large pool of individuals. However, SAKTGP creates these individuals randomly. On the other hand, SUFullTree creates the initial pool from the high-quality source individuals and these figures show how this difference affects the initial performance.

Another point to notice in Figure 4.7 is that the initial state of SAKTGP. Although similar to GPHH, this algorithm also creates the individuals randomly, the initial performance of SAKTGP is better than that of GPHH. This is the indication of effectiveness the surrogate algorithm on estimating the fitness of individuals.

Finally, it can be noted in Fig. 4.7 that the performance of SUFullTree



(a) *From Ugdb1 with 5 vehicles to Ugdb2 with 7 vehicles (Scn. 18)*

(b) *From Ugdb4 with 4 vehicles to Ugdb4 with 5 vehicles (Scn. 27)*

(c) *From Ugdb1 with 5 vehicles to Ugdb1 with 7 vehicles (Scn. 39)*

(d) *From Ugdb23 with 10 vehicles to Ugdb12 with 5 vehicles (Scn. 40)*

Fig. 4.8 *The distribution of solutions found with SAKTGP and SUFullTree and some existing knowledge transfer methods.*

at around generation 10 is almost comparable to the final performance of GPHH, that is achieved at generation 50. After generation 10, the performance of SUFullTree surpasses the performance of GPHH for the rest of the evolution. This is a clear indicator that the transfer of knowledge can improve the efficiency of the search process as well as its effectiveness; that is, transfer optimisation can reduce the cost (including the time) of finding the results that can perform similarly to the *final* solutions that are found with GPHH without knowledge transfer.

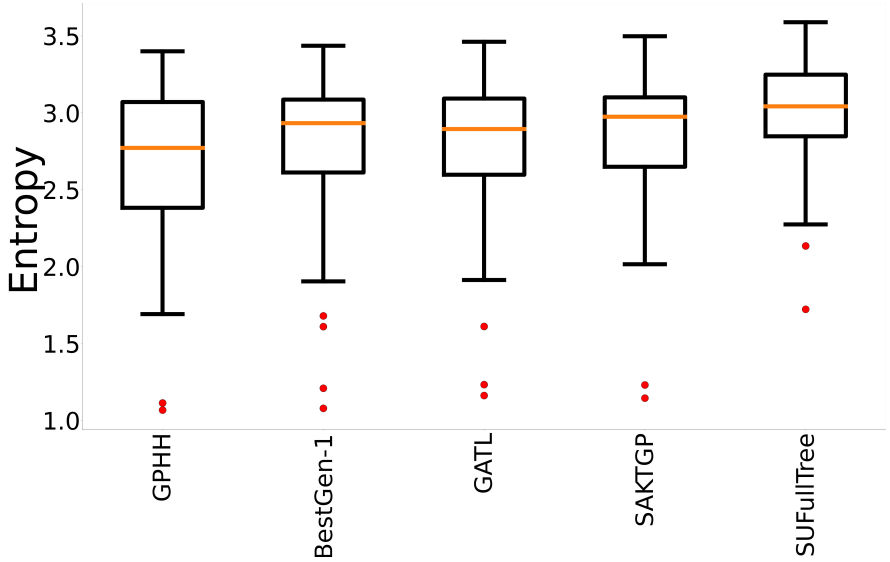


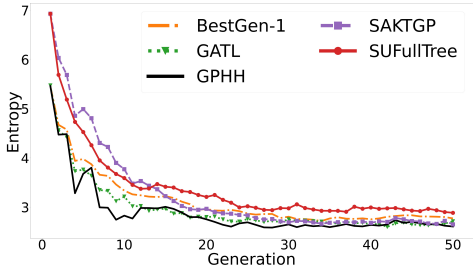
Fig. 4.9 *Average population entropy of SAKTGP, SUFullTree and the compared algorithms.*

The distribution of the solutions found with the compared algorithms are presented in Figure 4.8. The plots in this figure also confirm the results in Tables 4.3–4.4 and Figure 4.7 as we note SAKTGP and SUFullTree generally tend to create better solutions. In this figure, we note that the difference between the results obtained with SAKTGP and SUFullTree. Comparing the violin plots of these algorithms indicate the impact of creating the initial pool from the high-quality source individuals, rather than randomly.

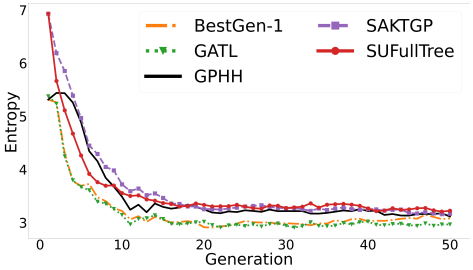
Figure 4.9 presents the distribution of population entropy of SAKTGP, SUFullTree and the compared algorithms. As is evident in the plot, SUFullTree has a better population entropy compared to the existing methods. In this figure, it can be noted that the entropy distribution of SAKTGP is rather similar to other methods, albeit with slightly better median value. On the other hand, SUFullTree tends to be better than SAKTGP. Considering that both SAKTGP and SUFullTree focus on creating diverse initial populations, Figure 4.9 indicates that creating the initial pool from

the set of unique high-quality source individuals is more likely to lead to a more diverse population. One reason for this observation could be that after removing phenotypic duplicates from the source individuals, each unique high-quality individual represent the areas of the search space that are more likely to contain local optima of the source problem. Because of the existence of similarity between the source and target problems, these local optima are also likely to be the local optima of the target problem or, to be near them. As a result, initialising GP with these local optima divides the population around multiple local optima and reduces the chance of getting the whole population stuck around a single local optimum. It is likely that after a few generations, the diversity will be lost significantly. However, it is likely that the duplicates are dispersed around more local optima. SAKTGP, on the other hand, focuses just on the uniqueness of the individuals without regarding their quality. As a result, many of the unique individuals probably will not have high quality and will be likely discarded in early generations of GP for solving the target problem. The differences in the average fitness of the first generation of SAKTGP and SUFullTree in Figure 4.7 also confirm this observation.

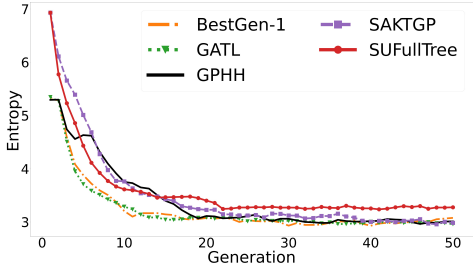
Figure 4.10 presents the population entropy of the algorithms, averaged over 30 runs. One common feature of all the plots in this figure is the entropy of the initial population of the SAKTGP and SUFullTree algorithms. The plots indicate that the effort on creating diverse initial populations has been successful. This figure also demonstrate that, even if the initial population diversity of SAKTGP and SUFullTree is high, the entropy drops quickly after a few generations. In some scenarios, as is seen in Figures 4.10a, 4.10b and 4.10d, the proposed algorithms manage to maintain higher entropies than the existing methods. Nevertheless, the plots indicate that the GP breeding operators tend to reduce the population diversity of routing policies for solving UCARP. This issue will be investigated further in Chapter 6.



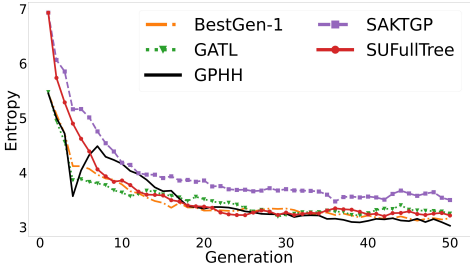
(a) *From Ugdb6 with 5 vehicles to Ugdb6 with 3 vehicles (Scn. 27)*



(b) *From Ugdb3 with 5 vehicles to Ugdb3 with 6 vehicles (Scn. 31)*



(c) *From Ugdb2 with 6 vehicles to Ugdb2 with 8 vehicles (Scn. 39)*



(d) *From Ugdb6 with 5 vehicles to Ugdb6 with 4 vehicles (Scn. 40)*

Fig. 4.10 *The average population entropy of SAKTGP and SUFullTree and some existing knowledge transfer methods.*

4.7 Chapter Summary

In Chapter 3, it was observed that the existence of duplicates and the lack of diversity in the transferred knowledge can decrease the effectiveness of transfer optimisation for solving UCARP. In this chapter, we proposed two approaches for handling this issue and improve the quality of transfer: 1) increasing population diversity *after* transfer of source knowledge; 2) removing duplicates from the knowledge source *before* transferring and then, transfer the unique individuals.

In the first approach, we proposed using high mutation rates after the transfer, with several mutation adaptation strategies to change the mutation rate dynamically over time. This approach was tested through a large

set of experimental studies. The studies indicated that although there were cases in which high mutation rates could increase the diversity slightly, it could not mitigate the negative impact of the transferred duplicates. Consequently, the proposed approach did not improve the performance of training routing policies for target problems. It can be argued that the mutation rate can be adjusted during the GP run in accordance with the population diversity. However, the results of DDGP-Exp indicate that mutation operator is not the appropriate tool for increasing the population diversity of UCARP because DDGP-Cos sets the mutation rate to very high values in early and late generations. However, our experiments indicate that the population diversity was not very high in these generations.

In the second approach set, two algorithms were proposed that removed phenotypic duplicates from the transferred knowledge. Since the knowledge source is dominated by phenotypic duplicates, removing them reduces the set of transferable individuals. To compensate for this, a large pool of individuals are created, either randomly or through mutating the unique high-quality source individuals. This pool is then evaluated with a surrogate model that is learned from the source individuals. Finally, the best individuals in the pool, in terms of the surrogate fitness, are used for initialising GP for solving the target problem. Our experimental studies demonstrated the superior performance of this approach. Our analysis showed that both algorithms performed significantly better than a selected set of existing methods as well as GPHH without knowledge. In addition, creating the diverse pool from the best of the source individuals tend to give better results that creating them randomly. Furthermore, we observed that the diversity of the initial population created by the proposed methods is better than the compared existing methods. However, this initial diversity is quickly lost after a few generations. This issue will be investigated and handled in Chapter 6.

In the next chapter, we will propose a new algorithm for considering the phenotypic behaviour of source routing policies and reusing them dur-

ing the course of the evolutionary process for solving the target problem. In this regard, extensive experiments will be conducted to verify the effectiveness of the proposed algorithm.

Chapter 5

Knowledge-guided Search

Our discussion of using the sub-tree transfer approach to performing transfer optimisation for UCARP in Chapters 3 and 4 were focused on using the transferred individuals for initialising GP for solving the target problem. In this chapter, we propose a novel approach to the use of the transferred knowledge after the initialisation and during the GP process.

5.1 Introduction

In Chapter 3, the general approach to transferring individuals from the final population of the source problem to initialise GP for solving the target problem was discussed and was shown that the final population may have converged to local optima and contained redundancies and duplicates. This becomes more serious when many GP individuals have different genotypes (i.e. tree structures) but the same phenotypic behaviour (routing behaviour). The duplicated genetic materials in the transferred individuals can mislead the search for solving the target problem into poor local optima. In this case, the transferred knowledge may misguide the search of the target problem and make it get stuck to local regions around the best region found for the source problem.

In Chapter 4, a set of novel approaches were introduced to handle

this problem. In that chapter, we showed how removing the duplicates from the set of transferred individuals and creating a diverse initial population for solving the target problem can improve the performance of GPHH. However, these approaches, as well as most of the existing methods, mainly focus on high-quality source solutions, ignoring the low-quality ones. Although there are cases in which low-quality solutions are also considered [150, 129, 62], this is usually done in a limited fashion with the purpose of increasing the diversity in the pool and handling the situations when the source and target problems are not as related as expected [150]. Based on the assumption that the source and target problems are related and their fitness landscapes share similarities, the high (low) quality solutions of the source problem tend to have high (low) fitness in the target problem. This knowledge about the search space of the source problem is important and can be beneficial in the target problem. It is even more important to consider a wider range of individuals from the source problem when the pool of good individuals is limited due to the lack of diversity.

Furthermore, the proposed approaches, as well as the majority of existing approaches for GP, do not utilise the transferred knowledge after the initialisation. The TLGPC algorithm [116], as described in Chapter 2, is one of the few exceptions to this observation but as was shown in Chapter 3, this method does not show a satisfactory performance for UCARP. The sub-par performance of TLGPC for UCARP could be attributed to the issue of the lack of diversity in the knowledge source. On the other hand, another reason for the observed performance of TLGPC could be the fact that it extracts the genetic materials, i.e. sub-trees, of good source individuals and was designed to be used in non-hyper-heuristic GP in which the genetic materials in the population also constitute the final solutions of the problem. However, for the case of UCARP, the genotypic contents of the population are not the final solutions of the problem but are the priority functions that can generate the final solutions, which are the vehicle routes. Accordingly, it is difficult to establish a relationship between

the genotypic space of the source problem from which TLGPC extracts the sub-trees, to the phenotypic space of the target problem that contains the target solution. The extracted sub-trees may be good enough for creating a better-than-random initial population for solving the target problem but as the evolution proceeds, these materials may lose their relevance for the target problem. This exemplifies the general issue which any method that tries to reuse the source genetic materials during the GP run may face for solving UCARP.

To address these shortcomings, in this chapter we propose a novel method for reusing the transferred knowledge after the initialisation and during the evolutionary process for solving the target problem. In our approach, after initialising the search with the high-quality source solutions, the search process is guided by the transferred individuals to discover phenotypically new individuals that have not been seen in the source. This encourages GP to explore new regions of the search space and prevents it from spending too many computational resources on evaluating individuals that have been seen in the knowledge source, which are unlikely to have high-qualities for the target problem.

5.2 Chapter Goal

To address the above issues, we propose a novel Genetic Programming with Knowledge transfer and Guided Search (GPGS). Our algorithm has two knowledge-guided components: 1) a new guided initialisation, and 2) newly proposed guided crossover and mutation operators. The guided initialisation salvages the good and unique genetic materials transferred from the source problem. Although it is possible to reuse previously proposed initialiation methods (i.e. SAKGPHH or SUFullTree), these algorithms utilise a surrogate model, and to avoid increasing the complexity of the proposed algorithm in this chapter, a new and simpler algorithm is proposed. The guided crossover and mutation operators employ the

transferred knowledge that include both high-quality and low-quality individuals, and maintain a tabu list to guide the offspring generation so that the search for the target problem will not revisit the regions already explored when solving the source problem. Since the tabu list can be very large, we devise an elaborate hashing mechanism for scanning the tabu list efficiently. More specifically, we will achieve the following research objectives in this chapter:

- Develop a new GP initialisation method which identifies the phenotypic duplicates from the source individuals and removes them before transferring to the target initial population.
- Develop novel guided crossover and mutation operators that take advantage of all the source individuals to prevent the search from revisiting the regions that have been explored for the source problem.
- Adapt a fast hashing algorithm for routing policies to speed up the comparisons to the large number of source individuals during the revisit check, which can greatly improve the efficiency of the guided crossover and mutation operators.
- Investigate the effectiveness of the proposed algorithm through extensive experimental studies and analyse its various components in detail.

5.3 Chapter Organisation

This chapter is organised as follows. Detailed descriptions of the proposed algorithm are given in Section 5.4. The experiment design is shown in Section 5.5, followed by results and discussions in Section 5.5.2. Finally, Section 5.6 concludes this chapter.

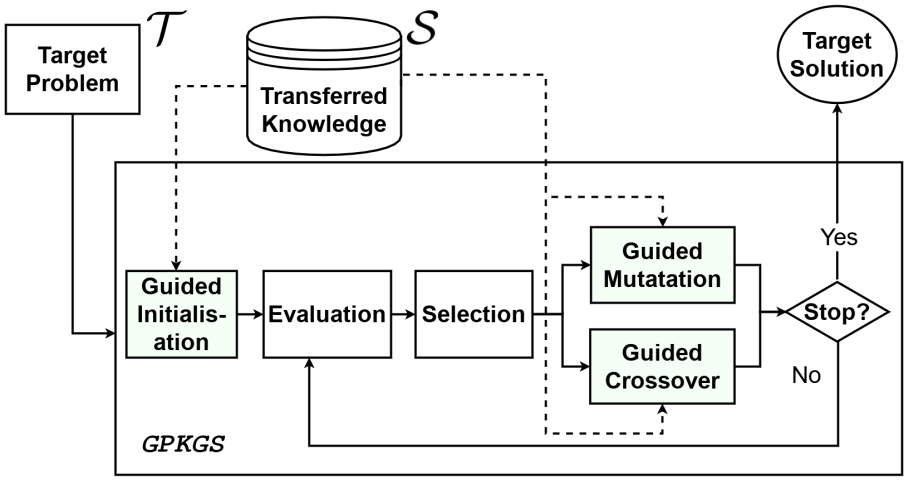


Fig. 5.1 *The GPKGS framework.*

5.4 Proposed Algorithm

5.4.1 Overall Framework

The overall framework of our proposed *GPKGS* is depicted in Figure 5.1. *GPKGS* has two inputs: 1) a target problem to solve, and 2) the knowledge transferred from the source problem(s). Here, we consider all the individuals evaluated by GP for solving the source problem, $\mathcal{S}$, as the transferred knowledge.

Compared with existing GP transfer optimisation methods, the proposed *GPKGS* contains the following novel contributions. First, *GPKGS* has a novel initialisation that removes the phenotypic duplicates from the source individuals before transferring them. It should be noted that this algorithm is different from the proposed initialisation method of SUFullTree, Algortihm 4.6, in that the new algorithm in this chapter does not create a large initial pool and hence, does not evaluate the pool with any surrogates. The newly proposed method also substantially increases the diversity of the transferred knowledge. Second, *GPKGS* contains novel guided crossover and mutation operators. In contrast to most existing methods

that use the transferred knowledge to generate (better) individuals, the guided crossover and mutation operators use the transferred knowledge to avoid revisiting the source individuals and thus encourage exploring new regions. The guided crossover and mutation operators are assisted by a powerful hashing algorithm that greatly improves their computational efficiency.

Algorithm 5.1 presents the pseudo code of *GPKGS*. It receives the transferred knowledge $\mathcal{S}$ from a previously solved source problem and a target problem $\mathcal{T}$ as inputs, and returns the best solution for $\mathcal{T}$. The algorithm starts by creating a hash table $\mathbf{\Psi}$ (line 2), which contains a summary of $\mathcal{S}$ for a fast query of its content. This hash table is inherently different from the hash table data structure that was discussed in Chapter 4. The details about $\mathbf{\Psi}$ will be given in Section 5.4.2. After creating the hash table, the guided initialisation (as shown in Algorithm 5.2) is used to create the initial GP population. After this, the population is evolved for $MaxGen$ number of generations. In each generation, the fitness of each individual in the population is evaluated (line 6) and the best individual is updated (line 7). Then, a new population is bred by the proposed `GuidedCrossover` (Algorithm 5.5), `GuidedMutation` (Algorithm 5.6), reproduction and elitism [130] (lines 9-26).

5.4.2 Guided Initialisation

The newly developed guided initialisation is described in Algorithm 5.2. First, the algorithm removes the duplicated trees from $\mathcal{S}$ (line 2) using the duplicate removal method in Algorithm 4.3 of Section 4.4. This takes into account the possible presence of redundancy in the transferred pool. Then, the unique individuals are sorted based on their fitness for the source problem (line 3), and $\kappa\%$ of GP population for the target problem is initialised with the best of them (line 4). The remaining of the population, considering the situation in which the number of unique individuals is

Algorithm 5.1: Pseudocode for *GPKGS*

Input : κ, the percentage of the initial population to transfer from the source problem; $\mathcal{S}$, all GPHH individuals discovered for solving source problem; $\mathcal{T}$, target problem to solve

Output: Ind^*, the best solution for target problem $\mathcal{T}$

```
1:  begin
2:      Ψ ← GenerateLSHTable (S) (Algorithm 5.3)
3:      Pop ← GuidedInitialisation (Ψ, κ) (Algorithm 5.2)
4:      g ← 0, Ind* ← null
5:      while g < MaxGen do
6:          Evaluate (Pop)
7:          Ind* ← SelectBest (Pop)
            // Create a new population
8:          Pop' ← Elitism ()
9:          while |Pop'| < |Pop| do
10:             op ← SelectOperator ()
11:             switch op do
12:                 case 'Crossover' do
13:                     ρ1, ρ2 ← TournamentSelect (Pop, 2)
                        // See Alg. 5.5
14:                     ρ'1, ρ'2 ←GuidedCrossover (Ψ, ρ1, ρ2)
15:                     Pop' ← Pop' ∪ {ρ'1, ρ'2}
16:                 end
17:                 case 'Mutation' do
18:                     ρ ← TournamentSelect (Pop, 1)
                        // See Algorithm 5.6
19:                     ρ' ← GuidedMutation (Ψ, ρ)
20:                     Pop' ← Pop' ∪ {ρ'}
21:                 end
22:                 case 'Reproduction' do
23:                     ρ' ← Reproduction (Pop)
24:                     Pop' ← Pop' ∪ {ρ'}
25:                 end
26:             end
27:             g ← g + 1, Pop ← Pop'
28:         end
29:     end
30:     return Ind*
31: end
```

Algorithm 5.2: `GuidedInitialisation`($\mathcal{S}$, $\mathbf{\Psi}$, κ)

Input : $\mathcal{S}$, all GPHH individuals discovered for solving source problem; $\mathbf{\Psi}$, the hashed summary of $\mathcal{S}$; κ, the percentage of the initial population to transfer from the source problem

Output: Pop, the initial GPHH population

1: $Pop \leftarrow \emptyset$
2: $\mathcal{S}' \leftarrow$ `RemoveDuplicate` ($\mathcal{S}$) (Algorithm 4.3)
3: $\mathcal{S}' \leftarrow$ `SortFitness` ($\mathcal{S}'$) `// from best to worst`
4: $Pop \leftarrow$ `SelectBest` ($\mathcal{S}'$, κ) `// Select the best`
5: **while** $|Pop| < PopSize$ **do**
6: $\quad ind \leftarrow$ `RandIndividual` ()
7: $\quad$ **if** `IsSeen` ($\mathbf{\Psi}$, ind) = ***false*** **then**
8: $\quad\quad Pop \leftarrow Pop \cup \{ind\}$
9: $\quad$ **end**
10: **end**
11: **return** Pop

smaller than $\kappa\%$, is filled with randomly-created individuals (lines 5-10) that are not explored in the source problem. Specifically, for adding each individual, we first create a new individual using the Ramped Half and Half approach [130]. Then, we scan the hash table $\mathbf{\Psi}$, and add the new individual into the population if it is not seen in $\mathbf{\Psi}$.

Hashing the Transferred Knowledge $\mathcal{S}$

To search for an individual q in $\mathcal{S}$, a straightforward approach is to simply compare each $s \in \mathcal{S}$ with q based on the dissimilarity function Δ (Eq. 4.1). This approach has a time complexity of $O(n)$ where $n = |\mathcal{S}|$, which can be very slow when $\mathcal{S}$ is very large. To address this issue, we employ an approximate search technique used for solving the nearest neighbour search problems that can search much more efficiently in large datasets [1, 209]. Although there exist exact search methods for low dimensions (e.g. dimensionality of 2 or 3) [57, 112], for high dimensions, these methods do not perform better than the linear search [232]. Hence, approximate

Algorithm 5.3: Eq. (5.2), `GenerateLSHashTable`($\mathcal{S}$)

Input: $\mathcal{S}$, history of all GPHH individuals discovered for solving source problem

Output: $\mathbf{\Psi}$, a hash table of the policies in $\mathcal{S}$ that maps a hash value to the set of routing policies with the same value

```
1  begin
       // The table of hashed individuals in S
       // to return.
2      Ψ ← {}
3      for i = 1 → τ do
4          Ψ ← ∅
5          for s ∈ S do
6              h ← LSHash (s) (Equation 5.2)
7              Ψ[h] ← Ψ[h] ∪ {s}
8          end
9          Ψ ← Ψ ∪ {Ψ}
10     end
11 end
12 return Ψ
```

methods with better time complexities have been proposed that are almost as effective as the exact ones [15, 132, 99, 128].

In this chapter, we employ the Locality-Sensitive Hashing (LSH) technique [112] that has a sub-linear dependence on data size. Indyk et al. [112] originally proposed the method for the d-dimensional binary Hamming space $H = \{0, 1\}^d$ and Datar et al. [57] extended LSH for the Euclidean space $\mathbb{R}^d$.

The key idea of LSH is to hash the items of a set S using several hashing functions, called *LSH functions* or *LSH family*, in a way that if two data items are more similar to each other, the probability of collision (having the same hash value) is higher. For $\mathbb{R}^d$, the LSH function can be of the form

$$h_{\boldsymbol{a},b}(\boldsymbol{o}) = \lfloor \frac{\boldsymbol{a} \cdot \boldsymbol{o} + b}{w} \rfloor \tag{5.1}$$

This LSH function projects the vector $\boldsymbol{o} \in \mathbb{R}^d$ into buckets of width w by first projecting it along a random line identified by $\boldsymbol{a} \in \mathbb{R}^d$ with the inner product operation and then, shifting the projection by a random shift value $b \in \mathbb{R}$. At last, the bucket that the projection $\boldsymbol{a} \cdot \boldsymbol{o}$, and hence $\boldsymbol{a}$, is located by applying the floor function. To reduce the probability that distant/dissimilar data items fall into the same bucket, a compound hash function is created by constructing λ hash functions via choosing a set of random lines $\boldsymbol{A}$ and shift values $\boldsymbol{B}$ and concatenating their results

$$h(\boldsymbol{o}) = \bigoplus_{\boldsymbol{a},b \in \boldsymbol{A},\boldsymbol{B}} h_{\boldsymbol{a},b}(\boldsymbol{o}) \tag{5.2}$$

where $\bigoplus$ represents the concatenation operator. To further decrease the chance of incorrect collisions, τ compound hash functions can be considered. Having a hash function, any dataset can be effectively hashed into τ hash tables [112, 57]. As a result, given a query point q, its hash value is first calculated and the hash tables are then queried and all the items with the same hash value are compared to q to find the match [112, 57, 84, 109].

The phenotypic characterisation in Algorithm 4.2 and the norm in Eq. 4.1 map routing policies to the $\mathbb{R}^d$ space. This allows us to utilise the LSH technique for efficient hashing and searching of the transferred archive $\mathcal{S}$ in a straightforward manner. Algorithms 5.3 and 5.4 present the process. First it is needed to hash the routing policies in $\mathcal{S}$ and create the set of hash tables $\boldsymbol{\Psi}$ with Algorithm 5.3. Then, Algorithm 5.4 is used to search for a policy q in $\boldsymbol{\Psi}$. Specifically, the hash value of q is calculated first (lines 2-3) and looked up in each hash table (line 5). If the hash value is in the table, the corresponding routing policies are acquired from the table. The retrieved set has a small size and by comparing the policies inside it against q (lines 6-10), it can be realized if q is in $\mathcal{S}$ or not.

Algorithm 5.3 hashes each $s \in \mathcal{S}$ for τ times and stores them in τ hash tables. As a result, the algorithm has $O(\tau n)$ time and space complexities. Algorithm 5.4 does not store any data and hence, has a space complexity of $O(1)$. Datar et al. [57] proved that the time complexity of Algorithm 5.4

Algorithm 5.4: `IsSeen`($\mathbf{\Psi}$, ρ)

Input: $\mathbf{\Psi}$, history of all GPHH individuals discovered for solving source problem, hashed with Algorithm 5.3; An individual ρ to search for in $\mathbf{\Psi}$
Output: **true** if ρ is in $\mathbf{\Psi}$ and, **false** otherwise

```
1  begin
2      ζ ← Characterise (ρ) (Algorithm 4.2)
3      h ← LSHash (ζ)
4      for Ψ ∈ Ψ do
           // Get policies with the same hash value
5          S' ← Ψ[h]
6          for s ∈ S' do
7              if Δ(s, ρ) = 0 then
8                  return true
9              end
10         end
11     end
12     return false
13 end
```

cannot be worst than $O(\tau n^{1/2} \log n)$ which is significantly better than the $O(n)$ time complexity of the linear search approach.

5.4.3 Guided Genetic Operators

To take better advantage of the transferred knowledge $\mathcal{S}$, the guided genetic operators consider the hash table $\mathbf{\Psi}$, and retain only the generated offspring that are not seen in $\mathbf{\Psi}$. This way, the search for the solutions of the target problem can focus more on the regions that are not explored when solving the source problem. In this regard, the role of the crossover operator becomes more prominent. However, since in our approach the population is initialised with the transferred solutions, it is possible that the neighbourhood of these solutions have been seen and evaluated before in the source problem. This increases the risk of creating individuals that have been seen before. To address this, we modify the crossover operator

Algorithm 5.5: `GuidedCrossover`(Ψ, ρ_1, ρ_2)

Input: Ψ, a hashed history of all GPHH individuals discovered for solving source problem; Two individuals ρ_1 and ρ_2 to perform crossover on
Output: Two offsprings ρ'_1 and ρ'_2

```
1  begin
       // Number of trials
2      try ← 0
3      Γ ← {}
4      while try ≤ η and |Γ| < 2 do
5          try ← try + 1
6          ρ'1, ρ'2 ← Crossover (ρ1, ρ2)
7          if IsSeen (Ψ, ρ'1) = false then
8              Γ ← Γ ∪ {ρ'1}
9          end
10         if IsSeen (Ψ, ρ'2) = false then
11             Γ ← Γ ∪ {ρ'2}
12         end
13     end
       // In case |Γ| < 2
14     Γ ← {ρ'1, ρ'2}
15     return Γ[1], Γ[2]
16 end
```

as is presented in Algorithm 5.5.

Algorithm 5.5 presents `GuidedCrossover`, the crossover operator of *GP-KGS*. The algorithm takes two parents ρ_1 and ρ_2, and the hash table Ψ. After performing the standard point-wise crossover [130] and creating two offspring, the algorithm uses the `IsSeen` function (Subsection 5.4.2) to check if any of the offspring has been seen in ψ. An offspring is stored in the Γ set if it is new. The crossover operation is repeated to create new offspring until either two new offspring are found or a the maximum number of trials, η, is reached (lines 4-10). The parameter η controls the amount of effort that *GPKGS* exerts for finding new and unseen individuals. It should be noted that for $\eta = 1$, `GuidedCrossover` reduces to the

Algorithm 5.6: `GuidedMutation`($\mathbf{\Psi}$, ρ)

Input: $\mathcal{S}$, history of all GPHH individuals discovered for solving source problem; an individual ρ to mutate
Output: A mutated individual ρ'

```
1  begin
       // Number of trials
2      try ← 1
3      ρ' ← Mutate (ρ)
4      found ← false
5      while try ≤ η do
6          ρ' ← Mutate (ρ)
7          try ← try + 1
8          if IsSeen (Ψ, ρ') = false then
9              found ← true
10             break
11         end
12     end
13     if found = false then
14         r ← rand (0, 1)
15         if r ≤ 0.5 then
16             ρ' ← randomly created individual
17         end
18     end
19     return ρ'
20 end
```

standard point-wise crossover. Finally, the Γ set is returned as the output of `GuidedCrossover`. If Γ contains less than two individuals, the most-recently created individuals are returned as needed (lines 14-15).

Algorithm 5.6 shows the `GuidedMutation` operator. Given the parent individual ρ, it is first mutated with the standard mutation [130] (line 3 and 6) into ρ'. Then, ρ' is checked against $\mathbf{\Psi}$, to find out if it has been seen previously or not (line 8). If ρ' is seen then it is discarded and a new mutation is performed on ρ with the hope of finding a previously unseen individual. This process is repeated for pre-set number of η times (lines

5-12).

After multiple tries, the mutation operator may not yield a new individual. In this case, one option is to accept the mutation result, even if it is seen before. On the other hand, since the intention of the mutation is to introduce diversity and randomness into population, we can create a totally random individual as the mutation result, with the hope that the random individual has a better chance of discovering a less explored region of the search space. The decision of returning a previously-seen individual or randomly-created one is made stochastically with a probability of 0.5 (lines 13-18).

5.4.4 Summary

The main goal in the development of *GPKGS* is utilise the phenotypic information that the GP process gained for solving a source problem more efficiently than the methods that were discussed and proposed in chapters 2–4. In this regard, *GPKGS* also reuses the unique and high-quality individuals of the transferred knowledge to initialise GP for solving the target problem. However, after the initialisation operation, the proposed algorithm remembers the phenotypic information of all the transferred individuals, including the low-quality ones. During the search, it tries to avoid recreating individuals that have the same phenotypic characteristics of a transferred individual. The reason for this attitude is that such individuals have already been evaluated for the source problem and if they have a high quality for the source problem, they are reused for initialising the target search. On the other hand, if they have a low quality for the source problem, they are also likely to have a low quality for the target problem too. In either case, allowing to create them during the search for the solution of the target problem will waste the computational resources. To prevent the search from recreating the source individuals, in this chapter we modified the crossover and mutation operators to check their results

against the set of transferred source individuals and if the result is in the set, repeat their operation.

It should be noted that act of prohibiting the search from creating some individuals shares some similarities with the tabu search algorithm [86, 87, 88]. However, tabu search is a meta-heuristic algorithm that performs local search around a single potential solution, i.e. the population size is 1 (although some of its recent variants do not have this restriction [89, 17, 136]). More importantly, in the tabu search algorithm, the tabu list is not fixed and is updated in every iteration of the algorithm. The update removes the old solution in the list in favour of the newest one. However, in *GPKGS* the list is fixed and is not updated with the results of search process for solving the target problem. Although it is possible to update the list during the *GPKGS* search, it is not done in this work so that focus is placed only on understanding the impact of the transferred knowledge.

5.5 Experimental Studies

In this section we test the effectiveness of the proposed *GPKGS* on a wide range of UCARP transfer optimisation scenarios based on the Ugdb, Uval and Uegl datasets described in [162]. The source and target problems of each scenario are presented in Table 3.1 (page 73).

5.5.1 Parameter Settings

For each scenario, we first solve the source problem using GP with population size of 1024 and 50 generations, and then transfer the $1024 \times 50 = 51200$ individuals to the target problem. Table 3.2 (page 75) presents the GP parameter settings in our experiments. As before, the division operator / is protected and will return 1 if its denominator is 0 and the $\min$ ($\max$) function returns the smaller (larger) value of its two input arguments. For training the routing policies, 5 sampled instances are considered for each

generation. The routing policies were tested with 500 samples. These settings are commonly used by previous studies, e.g. [145, 163].

GPKGS requires a set of decision situations $\mathbf{\Omega}$ to characterise routing policies (Algorithm 4.2). The accuracy of the dissimilarity measure in Eq. 4.1 depends on the number of decision-making situations utilised for computing it. The accuracy will increase by using more situations but the computational cost will also increase. To construct $\mathbf{\Omega}$ for each scenario and before starting the experiment, a vehicle was equipped with the path-scanning policy to serve a few tasks and the encountered situations were recorded. For our experiments, we considered twenty situations for characterisation of the routing policies which provided a good balance between accuracy and computational cost.

We set the LSH bucket size to $w = 10$ according to the analyses in [57]. The number of hash functions λ and hash tables τ control the accuracy of the compound hashing mechanism. Small values of λ and τ can increase the chance of false collisions and hence, increase the time required for performing the exact comparisons. However, after a certain point, increasing these values do not have a noticeable impact on the accuracy. In our preliminary testing, we realised that $\lambda = 10$ and $\tau = 100$ yielded good precisions so we used these values for all our experiments.

In addition to the conventional GP parameters, *GPKGS* has two extra parameters, the initialisation rate κ and the number of trials, η, for finding new individuals in `GuidedCrossover` and `GuidedMutation`. In our experiments, we evaluated the performance of the algorithm for a wide variety of values for these parameters and we obtained the best results with the $\kappa = 100$ and $\eta = 10$ values, which are used throughout the rest of this chapter.

5.5.2 Results and Discussions

To verify the effectiveness of *GPKGS*, we compare it with GP with knowledge transfer methods, i.e. *BestGen-1* [62], *GATL* [129] and DDGP [14], as well as the GP without knowledge transfer. In Chapter 4, two new algorithms, SAKTGP and SUFullTree, were introduced that demonstrated superior performance compared to the existing knowledge transfer algorithms, as well as the standard GPHH. However, these new algorithms are equipped with surrogate models and allows them approximate the fitness of a large number of individuals. However, since APTGP does not benefit from this additional opportunity for fitness evaluation, we decided exclude them from the experimental studies in this section.

Table 5.1 presents the fitness of solutions obtained with the compared methods over 30 independent runs. In this table, the minimum mean fitness value is marked in boldface. To investigate if there is a significant difference, the results in Table 5.1 are compared with the Friedman's test with a significance level of $\alpha = 0.05$. The ranks of the algorithms and the p-value are given at the bottom of the table. The test's p-value reveals that the difference between the compared algorithms is significant. Additionally, the rank of *GPKGS* is the best. To perform a pairwise comparison, the Conover post-hoc analysis [49] is applied on the results.

Table 5.1: Test performance of 30 independent runs of the compared algorithms (mean $\pm$ std)

Scn.	GPHH	GATL [129]	BestGen-1 [62]	DDGP [14]	GPKGS
1	91.2±0.1	91.2±0.1	91.2±0.1	91.2±0.1	**91.1±0.1**
2	551.0±10.3	550.8±8.1	551.1±10.4	552.1±8.3	**550.4±7.3**
3	**598.6±8.8**	599.6±9.5	600.5±11.6	602.3±12.5	599.0±9.2
4	639.5±11.3	**636.0±10.7**	640.6±12.1	640.4±10.4	642.7±12.7
5	58.2±0.1	58.2±0.1	58.3±0.1	58.2±0.1	**58.2±0.1**
6	424.8±8.5	424.9±8.8	423.9±8.6	425.3±6.1	**423.6±9.7**

Continuation of Table 5.1

Scn.	GPHH	GATL [129]	BestGen-1 [62]	DDGP [14]	GPKGS
7	432.1±7.1	**430.0±6.3**	431.2±6.3	430.6±7.6	431.9±7.3
8	432.6±5.5	430.6±6.7	432.7±4.8	431.4±5.5	**424.6±6.1**
9	576.2±3.9	575.8±4.2	576.8±3.7	576.0±3.9	**575.0±3.7**
10	340.5±4.7	338.1±4.2	337.5±3.1	338.0±2.6	**337.0±3.3**
11	347.2±6.1	347.1±6.0	345.9±4.8	346.7±5.7	**345.2±6.4**
12	**551.0±10.3**	553.7±10.5	551.8±10.1	553.2±11.4	554.4±9.7
13	598.6±8.8	598.5±7.5	**597.6±8.2**	604.6±11.2	601.1±12.6
14	639.5±11.3	640.1±12.2	639.0±11.9	640.5±15.4	**637.1±9.1**
15	340.5±4.7	339.8±3.5	339.9±5.0	338.5±3.6	**338.5±4.0**
16	444.4±4.7	445.0±7.6	**443.9±6.5**	445.4±6.4	446.2±5.6
17	324.3±6.2	323.3±5.2	321.5±5.4	323.8±5.7	**321.2±5.9**
18	360.3±3.1	359.7±3.7	359.4±3.9	358.4±4.4	**355.7±4.2**
19	358.3±2.6	**358.3±3.1**	358.8±2.7	362.0±7.1	359.7±6.4
20	359.0±1.8	358.5±1.8	358.6±1.9	358.9±1.7	**358.2±1.6**
21	340.8±4.4	340.8±2.2	339.4±4.7	340.8±3.3	**337.8±4.2**
22	351.9±3.5	**351.6±2.5**	352.5±3.5	351.7±3.0	351.7±4.7
23	356.6±1.6	356.4±1.5	356.1±1.3	**356.0±1.1**	356.1±1.2
24	310.9±0.5	311.0±0.3	310.7±0.8	310.0±1.5	**309.1±3.2**
25	389.2±0.2	**389.1±0.2**	389.2±0.2	389.2±0.2	389.2±0.2
26	363.1±2.8	363.1±3.2	363.4±2.6	**361.2±4.6**	362.0±3.9
27	342.1±6.2	342.5±7.8	340.9±8.0	343.3±5.9	**339.5±7.2**
28	382.0±5.5	**381.0±4.6**	381.3±8.0	384.0±9.3	383.6±6.1
29	382.8±3.3	383.9±2.6	**382.7±4.8**	383.0±5.2	382.9±4.4
30	351.5±2.5	351.1±2.2	351.7±1.2	351.2±2.0	**350.8±2.2**
31	326.0±4.7	325.2±4.0	325.2±5.0	324.5±5.4	**323.2±4.8**
32	444.4±4.7	443.7±5.6	**442.0±7.3**	443.5±7.6	443.4±5.7
33	448.2±0.5	449.0±2.3	**448.2±0.9**	449.4±2.6	449.9±2.8
34	**384.6±4.4**	387.1±6.0	386.9±5.0	389.2±5.5	386.6±5.9
35	369.3±1.8	**369.3±2.2**	369.8±3.8	371.3±4.1	369.3±3.0
36	**321.4±5.2**	322.7±4.2	323.8±5.1	322.9±5.5	322.2±3.2
37	166.2±2.0	166.1±1.7	165.2±1.5	165.9±1.6	**164.8±1.5**
38	376.1±7.6	381.2±7.0	377.8±7.5	378.8±7.5	**375.3±8.0**
39	415.7±9.2	415.5±7.2	414.3±8.9	414.8±8.0	**413.6±5.1**

Continuation of Table 5.1

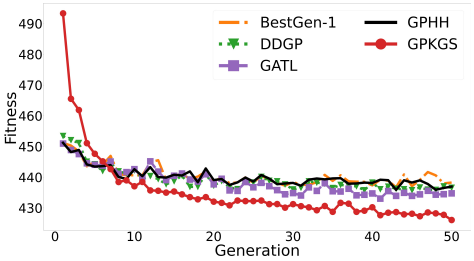
Scn.	GPHH	GATL [129]	BestGen-1 [62]	DDGP [14]	GPKGS
40	347.2±6.1	347.5±5.2	345.8±4.4	348.8±8.2	**344.3±4.6**
41	351.5±2.5	**351.4±2.7**	352.0±2.3	352.6±4.9	351.4±3.1
42	165.9±1.8	165.6±1.7	165.7±1.6	**165.3±1.9**	165.3±1.3
43	462.6±6.0	**457.4±7.0**	460.2±5.4	459.6±5.7	457.7±6.9
44	426.6±3.3	427.3±2.0	427.0±2.6	427.7±2.2	**425.7±3.2**
45	499.0±3.9	498.5±4.4	497.6±4.4	500.0±5.4	**497.1±3.8**
Rank	3.42	3.02	2.99	3.58	**1.99**
Friedman's p-value					**0**

Table 5.2 presents the p-values for the post-hoc tests with a significance level of $\alpha = 0.05$ after being adjusted with the Benjamini-Hochberg method [25]. In the table, the significant values are highlighted in boldface. As is evident from the table, the test rejects the null hypotheses for all the comparisons between *GPKGS* and other algorithms and signifies its superiority.

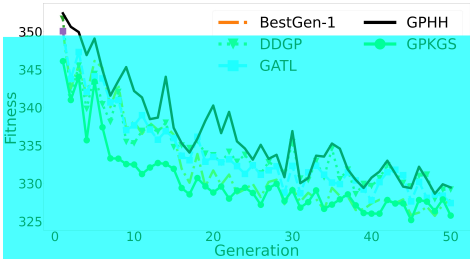
Figure 5.2 presents the convergence curve of algorithms for a few scenarios. According to these curves, we see that *GPKGS* starts with a better initial state compared to other existing methods, as well as GPHH with no knowledge transfer. Both *DDGP* and *GPKGS* initialise GP with the best individuals from the source problem. However, *GPKGS* removes phenotyp-

Table 5.2: Post-hoc comparison of the compared existing algorithms with adjusted p-values

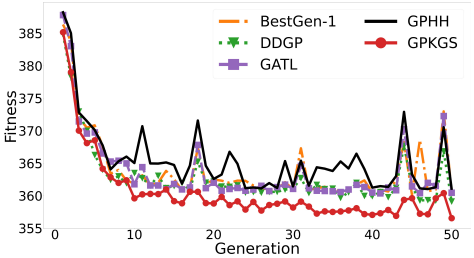
	GATL	BestGen-1	DDGP	GPKGS
GPHH	0.27	0.23	0.67	**0**
GATL	–	0.87	0.13	**0**
BestGen-1	–	–	0.11	**0**
DDGP	–	–	–	**0**



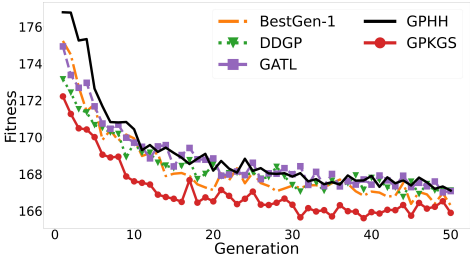
(a) *Ugdb17 with 5 vehicles to Ugdb17 with 5 vehicles (Scn. 8)*



(b) *Ugdb17 with 5 vehicles to Ugdb17 with 5 vehicles (Scn. 8)*



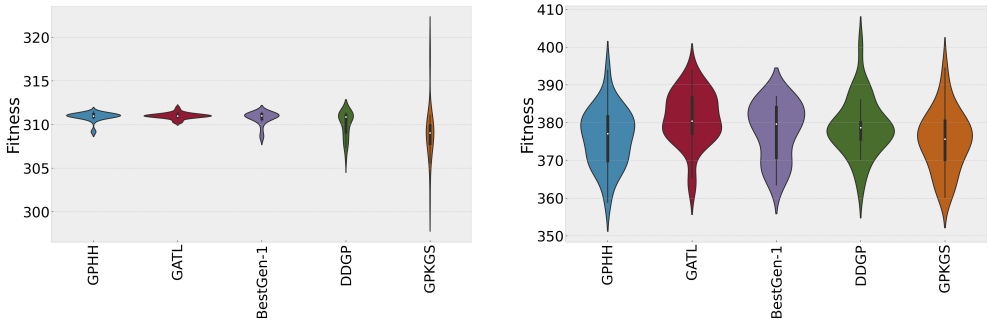
(c) *Ugdb17 with 5 vehicles to Ugdb17 with 5 vehicles (Scn. 8)*



(d) *Ugdb17 with 5 vehicles to Ugdb17 with 5 vehicles (Scn. 8)*

Fig. 5.2 *Convergence curve of GPKGS and some existing transfer methods.*

ically similar trees and Figure 5.2 indicates that the removal of duplicates leads to a better initial solution. This is due to the fact that the duplicates occupy the population without contributing any extra phenotypic information. Hence, when they are removed, other individuals, that may be less fit for the source problem but can perform better for the target problem, will have a chance to be included. Additionally, we see that *GPKGS* manages to retain its initial good performance throughout the evolutionary process. This is in contrast to other methods that may perform better than GPHH initially but then, they gradually lose their superiority and become very similar to GPHH. Although the convergence curve of other scenarios are not given due to space limit, we observed similar patterns in almost all other scenarios too.



(a) *Ugdb3, from 5 to 3 vehicles (Scn. 11)* **(b)** *Ugdb7, from 5 to 4 vehicles (Scn. 27)*

Fig. 5.3 *Performance violin plots of GPKGS and the compared transfer algorithms.*

Figure 5.3 presents the distributions of the performances obtained from 30 independent runs for a few of the experimented scenarios. The plots in this figure also confirm the difference between the *GPKGS* and the compared algorithms. Similar patterns were observed in almost all other scenarios.

5.5.3 Training Time

In the guided initialisation and guided genetic operators, *GPKGS* needs to search the archive of transferred individuals, $\mathcal{S}$, during crossover and mutation. In our experiments, GP is run for 50 generations for the source problem to evolve populations of 1024 individuals. As a result, $|\mathcal{S}| = 50 \times 1024 = 51200$ individuals are transferred for the target problem. Needless to say, linear search of this large set of individuals every time a new individual is created can be daunting. In *GPKGS*, we alleviated this challenge with the LSH technique to hash $\mathcal{S}$ into a hash table $\mathbf{\Psi}$ and search the $\mathbf{\Psi}$ instead. To confirm the challenge and to verify the need for using the hashing method, we disabled the LSH-based search mechanism of Algorithm 5.4 in *GPKGS* and replaced it with a simple linear search of $\mathcal{S}$. This increased the runtime of *GPKGS* significantly and made the it practically

impossible to use. To investigate further, we also considered a version of *GPKGS* that removes any duplicates from $\mathcal{S}$ and performs the linear search on it, instead of the LSH-based search. We refer to this algorithm as *GPKGS Duplicate-free Linear Search* or *DL-GPKGS*. This modification improved the execution time significantly but the execution time was still worse than *GPKGS*. This effect is clearly visible in Figure 5.4 which presents the mean training time of each investigated algorithm, averaged over all scenarios.

5.5.4 Further Analysis

Negative Transfer During Initialisation

It is clear from the convergence curve of the algorithm in Figure 5.2 that the guided initialisation helps jump-start the search. However, it is interesting to see an example of negative transfer during initialisation in Figure 5.2a, and note how the guided genetic operators could then serve to improve the search behaviour. In this direction, we can speculate two possible reasons as follows.

1. The source and target problems are not related or have limited similarities. In this case, it is natural that knowledge transfer is harmful

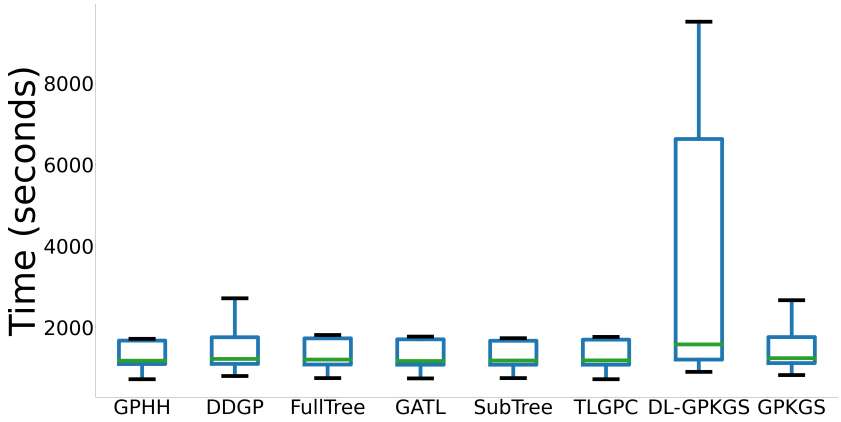


Fig. 5.4 *Training time of the compared algorithms*

for the target problem. With respect to our algorithm, this effect can manifest itself as a very bad initial state, i.e. worse than the random initialisation. Additionally, since our method uses the transferred knowledge during the search, we can expect the guided genetic operators hurt the effectiveness of the GP search because they prohibit visiting the areas of the search space that were not good for the source problem, which could potentially contain good solutions for the target problem. Looking at Figure 5.2a, this is not likely to be the reason.

2. There are some degree of similarity between the source and target problem but the change in the problem characteristics placed the local/global optima of the source problem in a very bad position of the search space of the target problem. In this case, it can be expected that the initialisation with the transferred knowledge will make the initial state of GP for the target problem very bad but over time, the guided search operators will improve the quality of the population. Figure 5.2a seems to represent such a scenario. We measured the similarity of the problems as $r = 0.61$. As can be seen in the Figure, the initial state of GPKGS is very bad compared to other algorithms, including GPHH. However, GPKGS is not the only algorithm whose initial state is worse than GPHH, for example, DDGP has the same situation. We can attribute the difference between the state of GPKGS and DDGP to the removal of duplicates from the transferred knowledge. The removal of duplicates removes the good duplicate individuals and allows non- duplicate individuals with worse fitness to be transferred. Naturally, these unique individuals have worse fitness for the target problem. As is evident, although the initial state of GPKGS is worse than all other methods, during the course of evolution, GPKGS improves its state and quickly, it becomes better than other methods despite the fact that its initial state was much worse. Considering the fact that GPKGS is the only algorithm that makes

an extensive use of the transferred knowledge after the initialisation stage, its better performance can only be attributed to the effectiveness of the proposed guided operators.

Effectiveness of Guided Search

As described in Section 5.4, *GPKGS* is comprised of three new components: initialisation, crossover and mutation. In this section, we consider the effect of each components on the performance of *GPKGS*. Specifically, we disable the `GuidedCrossover` by considering the ordinary crossover operator and name it *GPKGS-OX*. Similarly, *GPKGS-OM* is the *GPKGS* algorithm with its `GuidedMutation` replaced with the ordinary mutation operator. Additionally, if both the guided crossover and mutation operators are replaced with ordinary operators in *GPKGS*, the knowledge transfer is conducted only by the initialisation (Algorithm 5.2). This method, referred to as *GPKGS-OXM*, can be considered a modified version of the *FullTree* [62] method that does not transfer duplicate individuals and also does not transfer any knowledge about the search space of the source problem. Furthermore, we replaced the guided initialisation of *GPKGS* with the *FullTree* method but kept `GuidedCrossover` and `GuidedMutation`. This algorithm is referred to as *GPKGS-FT*. Finally, *GPKGS-0* stands for the *GPKGS* with $\kappa = 0$, i.e. no initial individual is transferred.

Table 5.3: Average fitness of 30 independent runs of *GPKGS* and its variants

Scn.	GPKGS-0	GPKGS-FT	GPKGS-OM	GPKGS-OX	GPKGS-OXM	GPKGS
1	60.0±0.1	59.9±0.2	59.9±0.2	59.9±0.2	59.8±0.2	**59.8±0.2**
2	91.1±0.1	91.1±0.1	91.2±0.1	91.2±0.1	91.2±0.1	**91.1±0.1**
3	58.2±0.1	58.2±0.1	58.2±0.1	58.2±0.1	58.2±0.2	**58.1±0.1**
4	**424.5±5.2**	427.7±6.0	425.9±6.4	427.5±4.9	428.8±6.9	426.9±5.0
5	576.0±4.0	574.9±6.2	575.1±3.8	576.3±4.6	575.4±5.2	**574.1±4.6**

Continuation of Table 5.3

Scn.	GPKGS-0	GPKGS-FT	GPKGS-OM	GPKGS-OX	GPKGS-OXM	GPKGS
6	125.5±1.0	125.3±0.7	**125.2±0.6**	125.7±0.9	125.4±0.9	125.4±0.7
7	249.4±1.7	250.0±1.7	249.5±1.8	249.1±1.6	249.4±1.9	**248.9±1.2**
8	443.5±4.7	446.8±7.8	446.8±7.8	444.4±8.1	447.4±9.2	**442.0±7.3**
9	322.5±6.5	321.6±5.5	319.9±5.5	319.3±5.3	320.8±5.9	**318.0±4.2**
10	**355.9±2.7**	358.0±5.6	356.4±4.5	356.9±3.4	358.4±5.4	358.0±5.0
11	340.8±6.5	338.2±5.1	337.4±6.2	337.2±6.1	338.7±3.8	**336.1±5.5**
12	350.7±3.0	351.7±4.0	351.0±2.5	**350.2±3.0**	350.4±1.6	350.3±3.9
13	356.2±1.2	356.6±1.7	356.5±1.6	**356.1±1.3**	356.4±1.6	356.3±1.4
14	310.6±2.5	309.9±1.8	308.6±1.8	308.9±2.4	308.9±1.8	**308.2±2.1**
15	340.1±5.9	340.5±4.9	**337.7±4.5**	338.6±7.1	338.8±6.5	338.7±4.9
16	356.7±3.8	357.0±4.8	**355.0±5.2**	355.6±4.0	357.0±4.4	355.2±5.3
17	358.5±1.9	358.4±1.8	357.5±2.1	357.7±1.6	358.1±1.3	**357.3±1.5**
18	361.5±4.5	362.2±3.3	361.8±1.8	361.0±3.7	361.5±2.1	**360.9±3.0**
19	384.4±8.0	386.2±6.7	384.4±6.3	384.6±6.0	383.4±6.1	383.8±6.5
20	441.8±5.0	442.4±7.4	441.7±5.4	441.3±6.4	441.2±7.0	**440.8±5.9**
21	382.4±4.9	383.8±4.1	381.9±3.9	**381.2±3.4**	381.8±3.9	381.6±2.9
22	449.0±1.9	449.7±2.6	449.0±1.6	448.4±1.3	448.4±0.7	448.6±1.4
23	165.2±1.5	165.4±1.8	165.1±1.8	**164.7±2.0**	165.0±1.5	164.8±1.4
24	321.3±4.1	323.9±5.1	322.2±2.9	321.7±3.5	321.0±2.9	**320.9±2.5**
25	368.8±2.2	370.9±4.1	368.5±2.4	368.9±2.3	368.2±2.2	**368.0±2.1**
26	389.1±0.1	389.2±0.2	389.1±0.2	389.1±0.1	389.1±0.2	**389.1±0.2**
27	370.0±7.2	377.7±7.7	373.1±7.1	373.1±7.8	376.2±7.6	**369.9±7.3**
28	383.4±6.1	387.7±5.7	387.1±5.0	383.6±4.5	384.6±5.0	**383.1±4.8**
29	351.7±1.0	350.7±2.2	351.2±1.1	**350.7±2.7**	351.1±1.9	351.1±1.9
30	345.6±6.4	**345.2±6.2**	346.6±4.6	345.4±6.0	347.0±5.9	345.6±6.4
31	324.6±4.5	325.1±3.5	324.2±4.5	**322.1±4.8**	322.7±4.8	322.6±5.2
32	165.2±1.5	165.2±1.2	165.1±1.6	165.3±1.0	164.9±1.4	**164.7±1.7**
33	411.0±10.4	**410.7±8.2**	412.6±7.3	411.7±6.5	411.9±6.6	410.7±9.7
34	354.9±4.5	356.3±5.4	354.1±3.9	354.4±3.9	353.8±4.0	**353.6±3.8**
35	381.8±2.7	383.1±2.7	381.9±2.4	**381.6±2.3**	382.0±2.0	381.9±2.8
36	351.6±1.2	352.0±3.2	**351.1±1.2**	351.3±2.1	351.1±2.2	351.1±1.2
37	194.9±2.9	195.3±2.4	195.3±2.4	195.2±2.3	194.9±2.2	**194.0±1.9**
38	381.9±4.0	381.1±3.4	381.0±3.2	380.4±3.2	380.1±4.3	**380.0±4.2**

Continuation of Table 5.3

Scn.	GPKGS-0	GPKGS-FT	GPKGS-OM	GPKGS-OX	GPKGS-OXM	GPKGS
39	249.2±1.4	249.4±1.3	248.5±1.2	248.5±1.1	**248.3±1.4**	248.4±1.4
40	247.6±2.2	247.0±2.1	246.6±1.5	246.7±1.1	246.8±1.4	**246.3±1.4**
41	457.9±8.0	459.5±5.8	457.4±7.2	458.2±7.2	458.6±6.3	**454.8±7.6**
42	426.0±4.6	426.7±4.0	425.8±4.3	**425.2±3.9**	426.5±3.1	425.8±3.6
43	500.5±4.6	498.1±4.8	498.4±4.6	496.8±3.3	**496.6±3.8**	497.2±4.5
Rank	4.15	5.33	3.71	3.13	3.8	**1.87**
Friedman's p-value						**1.1102e-16**

Table 5.3 presents the average performance of the algorithms in which the best fitness values are highlighted in boldface. To perform statistical analysis, we included the performance of GPHH from Table 5.1 (not shown again in Table 5.3 to save space) and conducted the Friedman test. GPHH was ranked 6.01 in this context. The ranks of other methods, as well as the p-value, are given in Table 5.3. As can be seen from Table 5.3, *GPKGS* has the best rank amongst the algorithms. Since the p-value indicates the existence of significant difference, the Nemenyi post-hoc comparison is performed and the obtained p-values are given in Table 5.4.

By comparing *GPKGS* and *GPKGS*-0, we observe that when the trans-

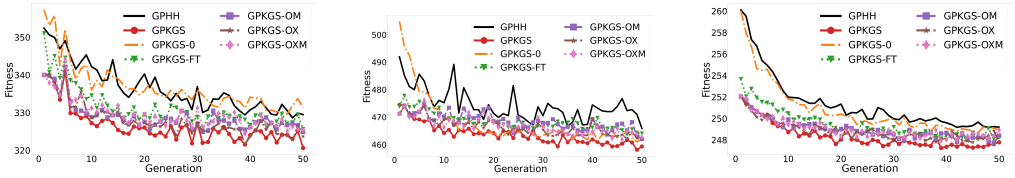
Table 5.4: p-values for the post-hoc comparison of the *GPKGS* and its variants

	GPHH	GPKGS-0	GPKGS-FT	GPKGS-OM	GPKGS-OX	GPKGS-OXM	GPKGS
GPHH	–	**13e-5**	1	**2e-05**	**0**	**4e-05**	**0**
GPKGS-0	–	–	0.26	1	0.59	1	**2e-05**
GPKGS-FT	–	–	–	**0.01**	**5e-05**	**0.02**	**0**
GPKGS-OM	–	–	–	–	1	1	**16e-5**
GPKGS-OX	–	–	–	–	–	1	0.15
GPKGS-OXM	–	–	–	–	–	–	**72e-5**

fer of genetic materials is disabled, the effectiveness of the resulting algorithm, *GPKGS-0*, degrades significantly. This observation indicates the importance of genetic materials. However, relying solely on knowledge-guided initialisation, crossover and mutation, and without any transfer of genetic materials, *GPKGS-0* still manages to outperform GPHH, which highlights the effectiveness of guided initialisation and search even if no genetic materials are transferred.

Furthermore, *GPKGS-FT* performs statistically similar to GPHH but significantly worse than *GPKGS*. *GPKGS-FT* is a version of *GPKGS* with guided crossover and mutation but without duplicate removal during initialisation. The fact that it performs worse than *GPKGS* but similar to GPHH indicates how detrimental the presence of duplicates can be to the effectiveness of knowledge transfer. This is further confirmed when we note that *GPKGS*-0, which does not transfer any genetic materials, outperforms GPHH. The presence of duplicates contributes to the lack of diversity in initial population and gives GP a negative bias towards the smaller regions of the search space that the duplicates represent. The negative effect of duplicates in the initial population can be understood even better when we note that *GPKGS-OXM*, which transfers only the unique individuals without any guided search, has a significantly better performance than *GPKGS-FT*.

By comparing *GPKGS-OXM* and GPHH we can further verify the effectiveness of duplicate removal. However, *GPKGS*-OXM performs significantly worse than *GPKGS* which indicates that, despite the effectiveness of the duplicate removal, the search performance can be improved significantly when it is guided with knowledge. In this regard and by comparing the performance of *GPKGS*-OM, *GPKGS*-OX and *GPKGS*-OXM, we note that the guided mutation is more effective than the guided crossover because *GPKGS-OX*, which utilises guided mutation but an ordinary crossover, performs significantly better than *GPKGS-OXM* and similarly to *GPKGS*. *GPKGS-OM*, on the other hand, cannot reach the performance of *GPKGS*,



(a) *Ugdb4, from 4 to 2 vehicles (Scn. 14)*

(b) *From Ugdb11dm2 with 5 vehicles to Ugdb11dm1 5 vehicles (Scn. 29)*

(c) *Ugdb23, from 10 to 12 vehicles (Scn. 37)*

Fig. 5.5 *Convergence curve of GPKGS and its variants.*

although it outperforms *GPKGS*-OXM. This result can be explained when the role of the mutation operator is considered as the main way that GP can introduce diversity into its population to escape from local optima. This role becomes more prominent when we know that GP may lose its population diversity and can converge to local optima when it solves UCARP [11, 10]. The convergence curve of the examined variants of *GPKGS* is given in Figure 5.5.

Computational Advantage

The `GuidedCrossover` and `GuidedMutation` operators attempt η times at finding unseen routing policies and discard the seen ones until either enough unseen individuals are created or the threshold is reached and seen results are accepted. The goal here is to avoid wasting computational resources on evaluating the individuals that are likely to have bad quality and use the saved resources better on evaluating new routing policies that are not seen previously. Table 5.5 presents the average number of times that `GuidedCrossover` and `GuidedMutation` found seen and unseen individuals over all our experiments. As can be seen, during a *GPKGS* run, `GuidedCrossover` encounters 44131.89 unseen and 15064.29 seen individuals on average. As a result, fitness evaluation is avoided for 15064.29 previously seen policies. Additionally, there were $11.84 + 29.09$ cases that `GuidedCrossover` could not find unseen offspring and previously seen in-

Table 5.5: Performance Statistics of `GuidedCrossover` and `GuidedMutation`

`GuidedCrossover`					
	0 Unseen Offspring	1 Unseen Offspring	2 Unseen Offspring	Total Seen	Total Unseen
Mean	18.00	36.06	22037.94	15937.69	44111.96
Std	24.90	28.18	64.19	6060.67	107.39
`GuidedMutation`					
	Total Random			Total Seen	Total Unseen
Mean	5.58			1510.63	3331.22
Std	14.85			975.11	1348.89

dividuals were accepted instead.

Similarly, `GuidedMutation` discards most of the 1649.22 previously seen individuals that it runs into and manages to find 4138.96 new policies on average. Additionally, there were 2.67 average number of times that `GuidedMutation` could not find any new individual and had to create a totally random policy. Hence, for the majority of cases, `GuidedMutation` managed to find unseen individuals and there were no need for creating random policies.

5.6 Chapter Summary

In this chapter, we aimed to propose a novel approach to transfer optimisation for GP to evolve effective routing policies for UCARP. We achieved this goal by devising an algorithm that performs knowledge transfer through two components: 1) transfer of unique genetic materials of a diverse set of good solutions; 2) transfer of an approximate overview of the areas of the search space that has low potential to be explored for the target prob-

lem, since its useful information has already been transferred in the initialisation. We developed an elaborate hashing method that enabled GP to efficiently compare solutions and avoid creating new individuals that are likely to be in a previously-explored area. We evaluated our method with an extensive set of experiments and observed that the new method improved the performance of GP significantly and also, outperform existing knowledge transfer methods. Our experimental studies showed that the proposed algorithm improves the effectiveness of GP significantly on almost all the test datasets and outperforms some of the existing transfer optimisation methods for GP. Furthermore, through a wide range of control experiments we verified the effectiveness of each proposed novel component of the algorithm.

This work has the following two key findings. First, the detection of duplicates in the source knowledge plays a crucial role in the effectiveness of knowledge transfer and significant improvement can be achieved in the quality of knowledge transfer if the duplicates are removed. Second, the transferred knowledge could be used in different ways rather than only initialising individuals. Given that the promising source individuals have been transferred to generate the initial target individuals, the other source individuals, regardless of their quality, can be used to encourage the search to explore new regions and solve the target problem better.

In the next chapter, we utilise the transferred knowledge to take a step further and, in addition to improving the final performance of GP, prevent the GP population from getting populated with phenotypic duplicates. Altough it is possible to reuse the guided reproduction algorithms that were proposed in this chapter, they are not utilised in the next chapter and the focus is put on investigating and detecting the contribution of each compoent of the new algorithm instead. In this regard, the effectiveness of the proposed algorithm will be investigated through extensive experimental studies.

Chapter 6

Knowledge Transfer with Auxiliary Population

In Chapter 3 we identified how lack of diversity in the knowledge source can contribute negatively to the quality of knowledge transfer for UCARP. In Chapters 4–5 we proposed a collection of approaches for handling this issue. In this Chapter, we will discuss how the transferred knowledge can be utilised so that the GP population will not be afflicted with the presence of duplicates and will not lose its diversity for solving the target problem.

6.1 Introduction

As was discussed through chapters 3–5, the GP process can lose its population diversity. For example, the final GP population can contain many phenotypic duplicates with the same (good) fitness and with or without common building blocks (i.e., sub-trees). The existing knowledge transfer methods that simply transfer knowledge from the top individuals tend to transfer many duplicated building blocks, and make the GP search in the target instance easily stuck into poor local optima. In addition, the existing GP-based transfer learning and optimisation methods are mostly limited to reusing the transferred building blocks (e.g., sub-trees or tree structure)

to initialise the target population. In this regard, on one hand, the use of the transferred building blocks can help the GP process start from a better-than-random initial population. On the other hand, the initial population may be too confined to the local regions of the individuals transferred from the source instance, especially when they have many duplicated building blocks due to the loss of diversity. As a result, it will be very difficult for the GP search for solving the target problem to jump out of the initial local region to find better regions for the target instance.

Apart from reducing the amount of high-quality transferable knowledge, a consequence of this phenomenon is that if the issue is not addressed correctly, an uninformed knowledge transfer may also transfer the problem of lack of diversity which may lead to convergence to local optima. In addition to these issues, to the best of our knowledge, there does not exist any transfer optimisation approach that can leverage the transferred knowledge to overcome the issue that afflicted the source problem, e.g. loss of diversity, for solving the target problem.

To address this issue, in this chapter we propose a novel GP with Auxiliary-Population for knowledge Transfer (*APTGP*). *APTGP* evaluates all the individuals for solving the source problem to form a pool of transferred knowledge. The algorithm fights off the issue of diversity loss by removing phenotypic duplicates from the pool and utilises the cleared pool to initialise the GP population for solving the target problem. Furthermore, after the initialisation, the transferred knowledge is utilised to help GP maintain its population diversity during the search for solving the target problem. In this regard, the transferred pool is preserved as a second auxiliary population. This auxiliary population evolves alongside the main population but instead of using the actual fitness function for evaluating its individuals, it is equipped with a surrogate model that is learned from and update by the main population. The main purpose of the auxiliary population is to help GP maintain its population diversity. To achieve this, we propose an elaborate immigrant exchange mechanism between

the main and the auxiliary populations.

6.2 Chapter Goal

In this chapter, we propose a novel *Knowledge Transfer Genetic Programming with Auxiliary Population for Solving UCARP*. In our approach, the transferred knowledge is retained after being used for initialising the GP population for solving the target problem. After initialisation, the auxiliary population is evolved with a surrogate model that is initially learned from the source individuals but is updated later on with the individuals that were found for solving the target problem. The main purpose of the auxiliary population is to help the main population maintain its phenotypic diversity. The benefit of this approach, which has drawn some inspiration from the island-based algorithms, is that the evolutionary process can be conducted separately and hence, each population has more potential for discovering different solutions, the exchange of which can improve the population diversity of both populations. Accordingly, in this chapter, the following research goals are pursued:

- To handle the potential presence of duplicates in the transferred knowledge, we develop a new initialisation mechanism that removes duplicates from the knowledge pool and transfers unique individuals.

- To enable GP to reuse the transferred knowledge after the initialisation phase, we propose a method for adapting the transferred knowledge for the target problem and reusing it more efficiently.

- To prevent GP from losing its population diversity, we devise an elaborate immigrant exchange mechanism between the main and the auxiliary populations that reduces duplicates in the population.

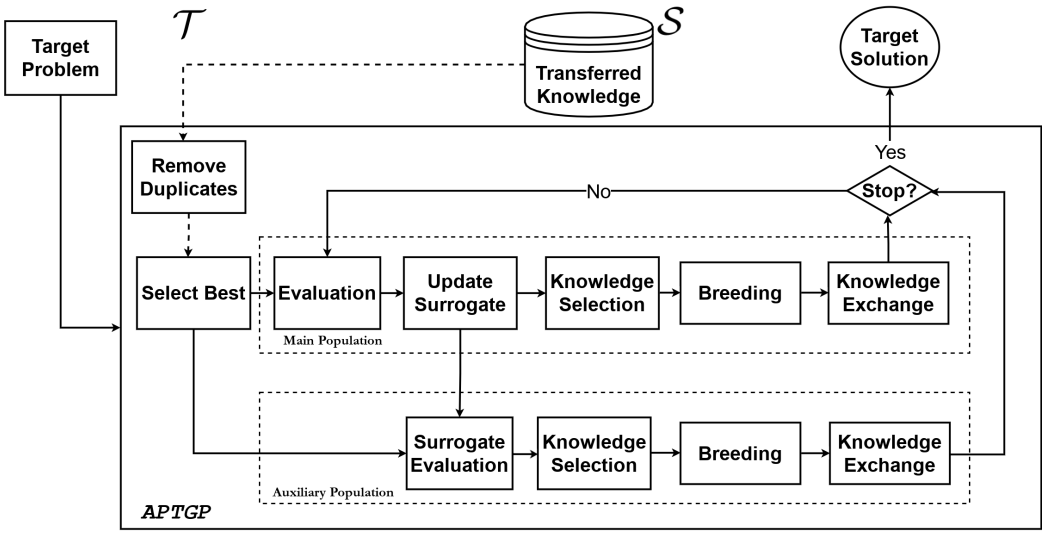


Fig. 6.1 *The APTGP framework.*

6.3 Chapter Organisation

This chapter is organised as follows. Detailed descriptions of the proposed algorithm are given in Section 6.4. The experiment design is shown in Section 6.5, followed by results and discussions in Sections 6.5.2 – 6.5.6. Finally, Section 6.6 concludes this chapter.

6.4 Proposed Algorithm

6.4.1 Overall Framework

Figure 6.1 presents the overall framework of *APTGP*. The inputs of the algorithm include the target UCARP instance to be solved, as well as the knowledge gained from solving the source instance. In this study, we consider that the knowledge simply includes all the examined routing policies by GP when solving the source instance.

To solve the target instance, *APTGP* first initialises the main popula-

tion and auxiliary population using the transferred knowledge, in order to have a better-than-random initial population. Then, the main and auxiliary populations are evolved in parallel. In addition, since the auxiliary population aims to assist the evolution of the main population, all the individuals in the auxiliary population are evaluated by surrogate. Here, we use the KNN surrogate that was described in Section 4.4.2.

To solve the target problem, *APTGP* initialises a GP population with the transferred knowledge in order to have a better-than-random initial state. However, in contrast to the existing methods, *APTGP* maintains and updates the transferred knowledge as a separate population and continues to exploit it during the course of evolution. However, since the fitness evaluation is computationally expensive, *APTGP* trains a surrogate model and uses it for estimating the fitness of the auxiliary population. After the evaluation of the main population, the surrogate is first updated. This allows *APTGP* to evaluate the auxiliary population with the most up-to-date surrogate. During the course of evolution, the populations exchange useful knowledge between themselves, which are selected before breeding and transferred after it. In our approach, the exchange phase is designed to share the common and useful knowledge between the populations and, to help GP overcome the problem of losing its population diversity. For breeding, the standard point-wise crossover, mutation and reproduction operators [130] are used.

The pseudocode of *APTGP* is presented in Algorithm 6.1. The behaviour of the algorithm is controlled with two parameters η and ϑ that correspond to the number of immigrants and the number of novelty trials respectively. The algorithm begins with removing phenotypic duplicates from the set of individuals (line 1) using the `RemoveDuplicates` method in Algorithm 4.3. Then, *APTGP* initialises two populations with the transferred knowledge (lines 1–2). Then, *APTGP* evolves these populations separately for $MaxGen$ number of generations. During the course of evolution, the main population is first evaluated with the actual fitness func-

Algorithm 6.1: The proposed *APTGP*

Input: $\mathcal{K}_S$: routing policies examined by GP for the source instance
Input: I_g: the target UCARP instance to solve
Input: Parameters η and ϑ
Output: $\texttt{rp}^*$: the best routing policy for the target instance

```
// Initialisation
```
1 $\mathcal{K}_{uni} = \texttt{RemoveDuplicates}(\mathcal{K}_S)$; $pop_1 = \mathcal{K}_{uni}[1 : popsize]$; // main population
2 $pop_2 = \mathcal{K}_{uni}[1 : popsize]$; // auxiliary population
3 $\texttt{rp}^* = null$, $gen = 0$;
```
// Search loop
```
4 **while** $gen < MaxGen$ **do**
5 Evaluate the routing policies in pop_1;
6 Update $\texttt{rp}^*$ with pop_1 ;
7 Update the surrogate model Υ with pop_1;
8 Evaluate the routing policies in pop_2 using the surrogate Υ;
```
// Select immigrants
```
9 $\Psi_1 = \texttt{Immigrants}(pop_1, \eta)$;
10 $\Psi_2 = \texttt{Immigrants}(pop_2, \eta)$;
```
/* Breeding with standard GP crossover, mutation and
   reproduction */
```
11 $pop_1 = \texttt{Breed}(pop_1)$;
12 $pop_2 = \texttt{Breed}(pop_2)$;
```
// Exchange Knowledge
```
13 $pop_1 = \texttt{ExchangeImmigrants}(pop_1, \Psi_2, \eta, \vartheta)$;
14 $pop_2 = \texttt{ExchangeImmigrants}(pop_2, \Psi_1, \eta, \vartheta)$;
15 $gen = gen + 1$;
16 **end**
17 **return** $\texttt{rp}^*$;

tion (line 5) and then used to update the surrogate model (line 7). Finally, the model is utilised to evaluate the auxiliary population (line 8). After evaluation, *APTGP* uses tournament selection to select η of the best individuals from each population as the transferable knowledge (lines 9-10). Then, a new population is bred from each population with the crossover,

mutation and reproduction operators (lines 11–12). Finally, the process of knowledge exchange is performed (lines 13–14).

In *APTGP*, knowledge quality and novelty are the two main aspects that govern the knowledge exchange process. Hence, in *APTGP* the exchange mechanism selects high-quality individuals to transfer if the they are not present in the target population, which requires searching both populations efficiently. To increase the search efficiency, first a hashed summary of the populations are created using the approach proposed in Section 4.4.2. Further details on the selection of the individuals and the exchange mechanism are given in Sections 6.4.4 and 6.4.5 respectively.

6.4.2 Initialisation

The main population and the auxiliary population are initialised by the top unique routing policies from the source knowledge $\mathcal{S}$. For this purpose, any duplicates in $\mathcal{S}$ are removed based on their phenotypic behaviours, by the `RemoveDuplicates` function. The `RemoveDuplicates` function is described in Algorithm 4.3 (page 124). Note that the algorithm can take any set of routing policies as input, and will be used by the `ExchangeImmigrants`() function as well. After duplicate removal, *APTGP* selects the best of the unique individuals, in terms of their fitness value for the source problem, to initialise two separate populations with the same individuals. As a result, two populations are initially the same.

6.4.3 Surrogate Model for Auxiliary Population

In *APTGP*, the auxiliary population does not utilise the main fitness function and instead, a surrogate model is utilised to give an estimation of the fitness value with a lower computational cost. In our approach, we utilised the KNN-based surrogate model described in Algorithm 4.5 of Section 4.4.2. In KNN-based surrogate models, a pool of routing policies are retained as the sample of the fitness values that the GP individuals can

Algorithm 6.2: `UpdateSurrogate`(Υ , Pop)

Input: Υ The surrogate pool to update; Pop A set of routing policies
Output: Υ The updated surrogate pool

```
1 Pop' ← RemoveDuplicates (Pop)
2 Υ ← Υ ∪ Pop'
3 if |Υ| > 2 * |Pop| then
4 |   Υ ← RemoveOldest (Υ)
5 end
6 return Υ
```

have. Then, to estimate the fitness of an individual, the nearest policy in the KNN pool is found and its fitness is returned as the fitness of the individual. In this chapter, we limit the size of the surrogate to be twice the population size based on the recommendation by Hildebrandt et al. [105]. Because the surrogate model is needed to be updated with the most recent individuals that have been evaluated for the target problem, we propose the function in Algorithm 6.2 for updating the surrogate pool. The design of this algorithm is motivated by the suspicion that the GP population may contain duplicates. In this algorithm, the `UpdateSurrogate` method first removes the duplicates from the given population before adding it to the pool and, if the pool size is larger than twice the population size, the pool is trimmed by discarding the oldest routing policies in it.

In Algorithm 6.2, Υ represents the KNN pool of routing policies. Initially, Υ is set to empty. In each generation, after evaluating pop_1, all the phenotypically unique individuals in pop_1 will be added into Υ. The maximal size of Υ is $2 * popsize$. If the number of elements exceeds the maximal size, Υ is trimmed by removing the oldest elements (line 4).

6.4.4 Immigrant Selection

Before the breeding, for each population, we select η immigrants preparing to be transferred to the other population. Each immigrant is selected

by the size-7 tournament selection. These immigrants are stored separately and are transferred to the other population after the breeding operation of GP. The pseudocode is shown in Algorithm 6.3.

Algorithm 6.3: `Immigrants`(pop, η)

Input: pop: a population of routing policies
Input: η: number of immigrants
Output: Ψ: the immigrants from pop

```
1 Ψ = ∅;
2 while |Ψ| < η do
      // Common tournament size for GP is 7
3     Randomly select 7 individuals from pop;
4     Add the best selected individual into Ψ;
5 end
6 return Ψ;
```

6.4.5 Knowledge Exchange

After the breeding, the main population and the auxiliary population exchange knowledge by transferring the selected immigrants to each other. When transferring the immigrants, the following two factors are considered: (1) to maintain diversity, the immigrants should not be a duplicate in the goal population, and (2) the immigrants should replace the less useful individuals in the goal population.

Algorithm 6.4 shows the pseudocode of the knowledge exchange. First, it finds the unique individuals in the goal population pop (line 1) into the unique (uni) and then finds the replaceable duplicated individuals ($replaced$) by subtracting the set of unique individuals from pop (line 2). The individuals to be replaced ($replaced$) is first set to all the duplicated individuals, as they are expected to contribute little to the population. If there are not enough duplicates (i.e. $|replaced| < \eta$), we fill in the remaining places by the size-7 reverse tournament selection policies from

Algorithm 6.4: `ExchangeImmigrants`$(pop, \Psi, \eta, \vartheta)$

Input: pop: the destination population
Input: Ψ: the immigrants from the source population
Input: η: number of immigrants
Input: ϑ: number of trials
Output: The new destination population pop

```
// Select individuals to be replaced in pop
uni = RemoveDuplicates(pop) ;
replaced = pop \ uni ;
while |replaced| < η do
    // Common tournament size for GP is 7
    Randomly select 7 individuals from uni;
    Add the worst selected individual into replaced;
end
replaced = ReverseSortByFitness(replaced) ;
// Refine the immigrants
Ψ' = ∅;
for i = 1 → η do
    for tries = 1 → ϑ do
        // See the Hash function in Algorithm 4.4
        if ∃rp ∈ uni, Hash(Ψ[i]) = Hash(rp) then
            // Standard mutation
            Ψ[i] = Mutate(Ψ[i]) ;
        else
            Ψ' = Ψ' ∪ {Ψ[i]};
            break;
        end
    end
end
// Transfer immigrants
replaced[1 : |Ψ'|] = Ψ';
return pop;
```

uni (lines 3–6). That is, each time we randomly sample 7 individuals from uni, and select the one with the worst fitness into $replaced$. Then, we sort $replaced$ by fitness from worst to best (`ReverseSortByFitness()`) (line 7). Note that `ExchangeImmigrants()` is called between the breeding and evaluation, and the newly generated individuals are not evaluated yet. In this case, we simply use the fitness inherited from the parent for the sorting.

Next, the immigrants are refined to make sure they are not duplicates of the goal population. For each immigrant, if it is a duplicate of the goal population (line 11), then we mutate it to generate a different immigrant (line 12). We keep mutating the immigrant until it becomes a non-duplicate or the maximal number of trials ϑ is reached, and Ψ is refined to Ψ'. Finally, we select the worse individuals (either duplicates or worst fitness) in pop and replace them with Ψ'.

6.4.6 Summary

The main idea of *APTGP* is to maintain two populations in the target instance and interact with each other to improve the search effectiveness. First, the main population is initialised by the unique individuals from the source knowledge, which can make the search start from a better region. Second, the auxiliary population starts from the same region as the main population, and evolves alongside the main population but towards different directions (e.g., by using the surrogate fitness). Third, the main population regularly receives non-duplicate immigrants from the auxiliary population, which can help it jump out of the initial local region and handle the concept drift from the source to the target instance. Finally, the main population migrates its best individuals to the auxiliary population to influence its search towards the best solutions that have been found so far.

It should be noted that our algorithm bears some similarities with the cultural algorithms [199, 181] or the distributed evolutionary algorithm

(the island model). However, the cultural and distributed evolutionary algorithms focus on solving a single problem, while *APTGP* focuses on transfer optimisation and takes advantage of the source individuals during the search process in the target problem. None of the populations in *APTGP* act as a belief space like in the cultural algorithms. Furthermore, in contrast to the cultural and distributed evolutionary algorithms, the populations in *APTGP* are not initialised randomly, none of the populations act as a belief space, all populations are evolved separately and the auxiliary population is evolved with a surrogate model.

6.5 Experimental Studies

To evaluate the effectiveness of *APTGP*, we conduct experiments on a wide range of source and target UCARP instances. To maintain consistency with the experiments in our previous chapters, we perform our experiments on the same set of transfer scenarios defined in Table 3.1. For the same reason, we use the same GP settings in our previous experiments, as is given in Table 3.2.

6.5.1 Parameter Settings

For each scenario, the source problem is solved first with vanilla GP that evolved 1024 individuals for 50 generations and then, $1024 * 50 = 51200$ routing policies are considered as the transferable knowledge. In all experiments, the training is performed with 5 sampled instances that are rotated each generation to prevent overfitting [145]. The solutions are tested with 500 sampled instances. All experiments were conducted for 30 independent runs. All statistical tests are performed with a $\alpha = 0.05$ significance level.

The phenotypic characterisation of the routing policies in *APTGP* requires a set of decision situations $\mathbf{\Omega}$. For each scenario, the path scanning

Table 6.1: Summary of compared existing algorithms

Algorithm	Knowledge transfer mechanism
GATL [129]	Select the best and median trees of each generation into a pool. Choose randomly from the pool to initialise the target GP population.
BestGen-1 [62]	Select k of the best individuals of each generation into a pool. Choose randomly from the pool to initialise the target GP population.
SUFullTree [13]	Select the best individuals of all the generations into a pool. Expand the pool with policies created from the good individuals and evaluate them with a surrogate. Choose the best from the pool to initialise the target GP population.

policy in Figure 4.2 was utilised to serve a few tasks for the target problem and the encountered decision situations were observed and 20 of them were recorded as $\mathbf{\Omega}$. In our experiments, the size of 20 situations provided a reasonable balance between the computational cost of characterising the policies and the accuracy of the characterisation. *APTGP* has two parameters η and ϑ that control its behaviour. We tested *APTGP* with different values of these parameters and achieved the best results with $\eta = 200$ and $\vartheta = 10$, which will be used throughout this chapter.

6.5.2 Results and Discussions

To examine the performance of *APTGP*, we compared it with some of the best available transfer optimisation methods for GP. The selected algorithms include *SUFullTree* [13], BestGen-1 [62] and *GATL*[129]. A summary of the compared algorithms is given in Table 6.1. In all experiments, we utilised the Friedman's statistical test with a confidence level of $\alpha = 0.05$.

Table 6.2 presents the test performance of the compared algorithms, where the best results are highlighted in boldface. As is evident, in almost all cases, *APTGP* had a best test performance among the compared ones. The Friedman test is also conducted to verify statistical significance. The

calculated rank and p-value are given in the bottom rows of Table 6.2. We can see that *APTGP* has the best rank and the p-value shows that the difference between the results is significant. *APTGP* obtained the best mean test performance for all the scenarios in the experiments. To pinpoint the difference, the Conover post-hoc analysis [49] is performed and the p-value of the pairwise comparisons are given in Table 6.3 after being adjusted with the Benjamini-Hochberg method [25]. The very small p-values in the last column of the table show that the difference between *APTGP* and all other algorithm is significant and, considering its rank, this indicates its superiority to the other methods.

Table 6.2: Test performance of 30 independent runs of the compared algorithms (mean $\pm$ std)

Scn.	GPHH	GATL [129]	BestGen-1 [62]	SUFullTree [13]	*APTGP*
1	91.2±0.1	91.2±0.1	91.2±0.1	91.2±0.1	**91.1±0.0**
2	551.0±10.3	550.8±8.1	551.1±10.4	551.5±9.4	**542.9±10.7**
3	598.6±8.8	599.6±9.5	600.5±11.6	601.9±11.0	**595.0±9.7**
4	639.5±11.3	636.0±10.7	640.6±12.1	644.1±15.8	**632.6±7.3**
5	58.2±0.1	58.3±0.1	58.3±0.1	58.2±0.1	**58.1±0.1**
6	424.8±8.5	424.5±8.1	423.9±8.6	423.2±8.6	**417.6±7.2**
7	432.1±7.1	431.0±6.7	431.2±6.3	431.8±8.3	**427.6±5.8**
8	432.6±5.5	432.2±5.2	432.7±4.8	430.0±6.1	**423.3±5.5**
9	576.2±3.9	576.5±3.8	576.8±3.7	576.4±3.3	**572.0±4.6**
10	340.5±4.7	338.1±4.2	337.5±3.1	337.2±2.1	**335.1±4.0**
11	347.2±6.1	347.1±6.0	345.9±4.8	344.4±5.1	**340.6±4.1**
12	551.0±10.3	553.7±10.5	551.8±10.1	552.7±8.9	**544.8±8.0**
13	598.6±8.8	598.5±7.5	597.6±8.2	600.2±10.3	**595.4±6.2**
14	639.5±11.3	640.1±12.2	639.0±11.9	640.8±14.8	**630.9±5.7**
15	340.5±4.7	339.8±3.5	339.9±5.0	337.0±3.3	**334.4±3.2**
16	444.4±4.7	444.5±5.6	443.9±6.5	444.1±4.9	**441.4±5.6**
17	324.3±6.2	322.4±5.7	321.5±5.4	319.8±4.6	**317.1±3.8**
18	360.3±3.1	359.4±4.5	359.4±3.9	356.2±4.2	**352.2±2.4**
19	358.3±2.6	358.2±3.4	358.8±2.7	357.6±5.3	**354.8±2.6**
20	359.0±1.8	359.0±1.3	358.6±1.9	358.3±1.1	**356.6±1.2**

Continuation of Table 6.2

Scn.	GPHH	GATL [129]	BestGen-1 [62]	SUFullTree [13]	*APTGP*
21	340.8±4.4	340.9±4.6	339.4±4.7	337.6±4.6	**332.6±5.9**
22	351.9±3.5	353.1±4.8	352.5±3.5	350.2±3.3	**349.7±2.4**
23	356.6±1.6	356.7±1.7	356.1±1.3	356.3±1.4	**355.7±0.7**
24	310.9±0.5	311.0±0.5	310.7±0.8	308.8±2.8	**307.9±4.0**
25	389.2±0.2	389.1±0.2	389.2±0.2	389.1±0.2	**389.0±0.1**
26	363.1±2.8	363.2±4.1	363.4±2.6	362.2±3.1	**357.8±4.9**
27	342.1±6.2	341.9±5.1	340.9±8.0	338.6±4.7	**335.1±4.6**
28	382.0±5.5	380.2±6.0	381.3±8.0	384.7±6.0	**378.3±6.5**
29	382.8±3.3	382.6±2.2	382.7±4.8	382.1±3.2	**381.0±2.7**
30	351.5±2.5	351.4±1.1	351.7±1.2	351.0±1.6	**349.9±3.2**
31	326.0±4.7	323.5±4.4	325.2±5.0	322.8±4.9	**320.1±4.7**
32	444.4±4.7	445.2±6.8	442.0±7.3	441.2±5.8	**439.9±6.4**
33	448.2±0.5	448.4±0.8	448.2±0.9	448.9±1.8	**448.0±0.6**
34	384.6±4.4	385.8±6.5	386.9±5.0	384.3±5.1	**382.9±4.6**
35	369.3±1.8	369.2±2.8	369.8±3.8	368.4±2.6	**367.1±1.9**
36	321.4±5.2	323.7±5.5	323.8±5.1	321.1±2.5	**319.5±1.7**
37	166.2±2.0	166.0±1.4	165.2±1.5	165.1±2.0	**163.8±1.4**
38	376.1±7.6	379.8±7.8	377.8±7.5	372.0±9.2	**366.6±5.9**
39	415.7±9.2	416.6±8.4	414.3±8.9	412.3±7.4	**409.0±8.3**
40	347.2±6.1	347.1±6.7	345.8±4.4	344.5±5.0	**340.8±4.9**
41	351.5±2.5	351.7±2.3	352.0±2.3	351.2±1.9	**350.3±3.1**
42	165.9±1.8	165.7±1.5	165.7±1.6	165.2±1.2	**163.9±1.4**
43	462.6±6.0	460.8±5.3	460.2±5.4	456.7±6.5	**451.8±7.3**
44	426.6±3.3	427.1±2.5	427.0±2.6	425.8±3.8	**424.3±4.3**
45	499.0±3.9	498.8±4.5	497.6±4.4	497.5±3.3	**496.5±3.4**
Rank	3.94	3.82	3.61	2.62	**1.00**
Friedman's p-value					**0**

Figure 6.2 presents the convergence curve of the algorithms for a few scenarios. As can be seen, in all cases, *APTGP* starts with a better initial state in the target instance, and manages to maintain a better performance throughout the entire evolutionary process. We observed similar patterns in almost all other scenarios.

In Figure 6.3, the distribution of the fitness value of the solutions obtained with each algorithm is given in the form of violin plots for a few representative scenarios. It can be seen that *APTGP* obtained the best (lowest) distributions. Similar patterns were observed in other scenarios too.

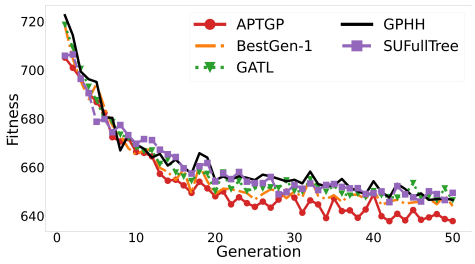
6.5.3 Program Size

Figure 6.4 presents the distribution of the program sizes (i.e., number of nodes in the tree) in the final population obtained by the compared GP algorithms. From the figure, it is clear to see that *APTGP* obtained larger trees (65 nodes on average versus 58 nodes by GPHH). This suggests that *APTGP* managed to capture more complex and sophisticated interactions between the features for decision making in UCARP.

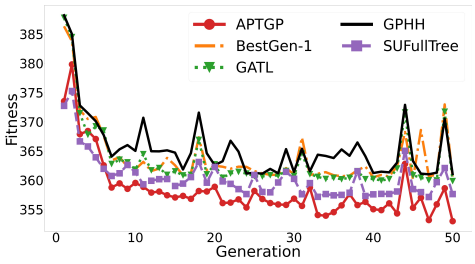
Figure 6.5 presents an example of the evolved routing policies. As is evident, the policy does not have a trivial size which makes it difficult to interpret but at the same time, it represents the complexity of the processing it performs on the state of the environment. Nevertheless, it is possible to gain some high-level insights from this policy. In this policy, the most frequent terminals is CR (Cost to Refill). This indicates that this policy considers this information about the environmental state to be more important. This is consistent with our domain expertise that during the course of serving tasks, a majority of the total cost pertains to the cost returning to depot to refill.

Table 6.3: Post-hoc comparison of the compared existing algorithms with adjusted p-values

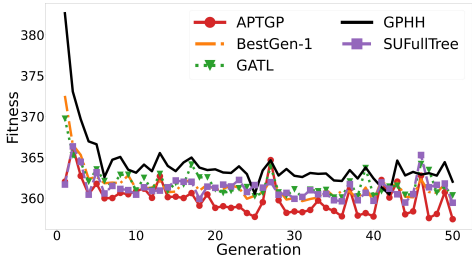
	GATL	BestGen-1	SUFullTree	APTGP
GPHH	0.67	0.33	**0**	**0**
GATL	–	0.54	**0**	**0**
BestGen-1	–	–	**0**	**0**
SUFullTree	–	–	–	**0**



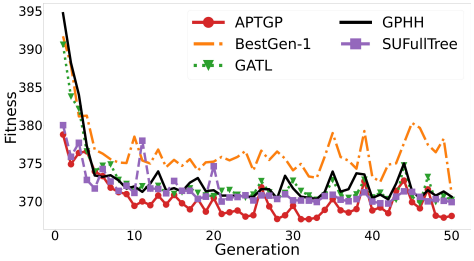
(a) *from Ugdb23, from 10 to Ugdb12 with 8 vehicles (Scn. 14)*



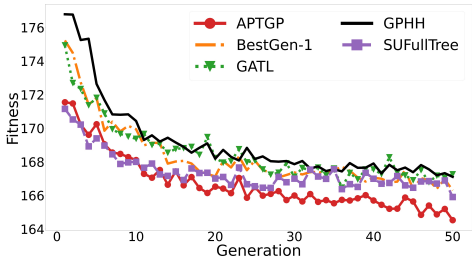
(b) *from Ugdb7 with 5 vehicles to Ugdb1, 6 vehicles (Scn. 18)*



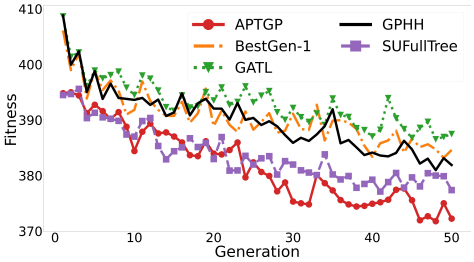
(c) *Ugdb3, from 5 to 7 vehicles (Scn. 20)*



(d) *Ugdb6, from 5 to 7 vehicles (Scn. 35)*

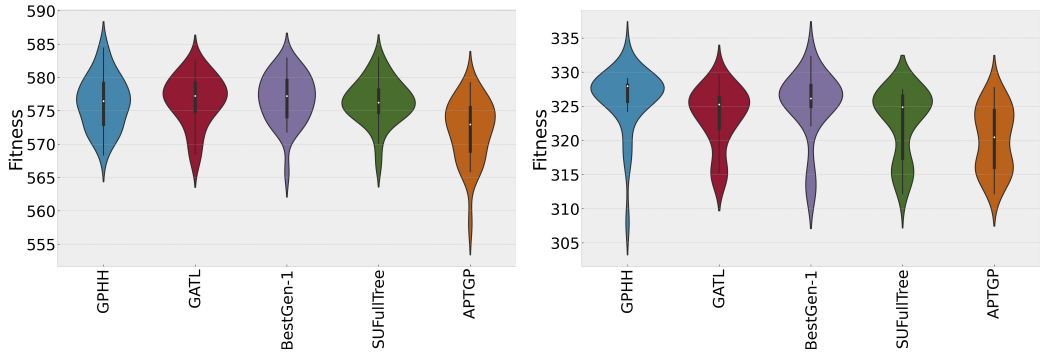


(e) *Ugdb21, from 6 to 4 vehicles (Scn. 37)*

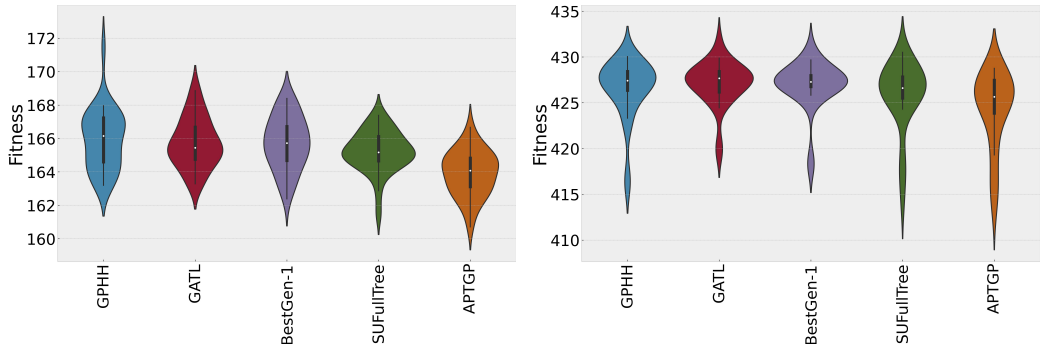


(f) *Ugdb7, from 5 to 4 vehicles (Scn. 38)*

Fig. 6.2 *Convergence curve of APTGP and some existing transfer methods.*



(a) *from Uegl-1-C with 8 vehicles to Ugdb13, 5 vehicles (Scn. 9)* **(b)** *Ugdb3, from 5 to 6 vehicles (Scn. 31)*



(c) *Ugdb21, from 6 to 5 vehicles (Scn. 42)* **(d)** *From Uval8A to val8Adm1 with 3 vehicles(Scn. 42)*

Fig. 6.3 *Violin plots of the performance of APTGP and the compared transfer algorithms.*

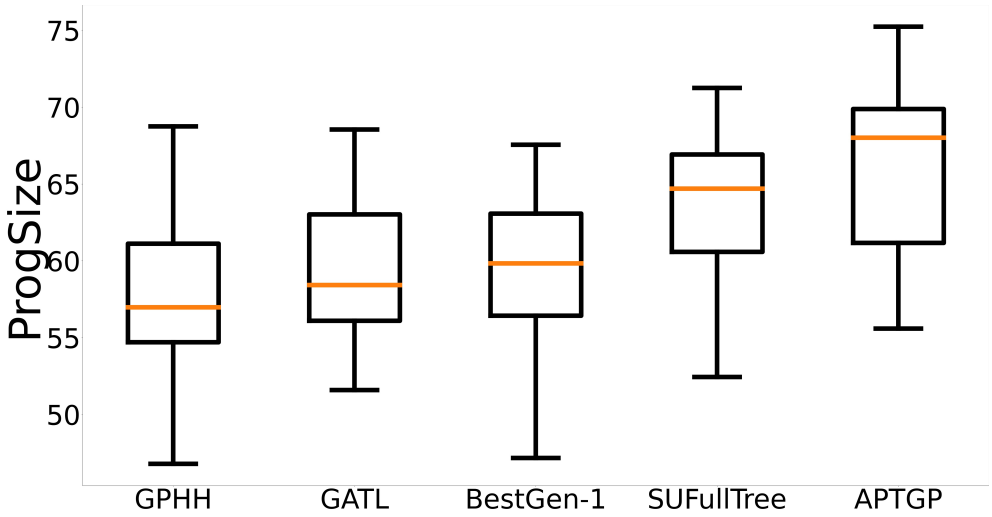


Fig. 6.4 *Size of the programs evolved with the compared algorithms.*

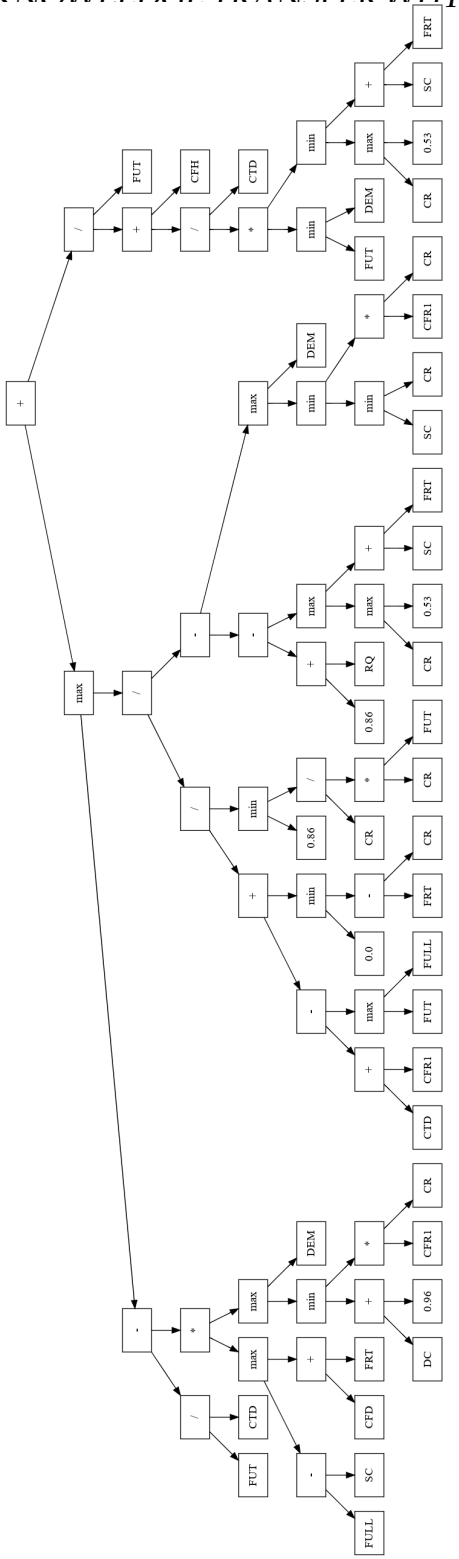


Fig. 6.5 *A programs evolved with APTGP.*

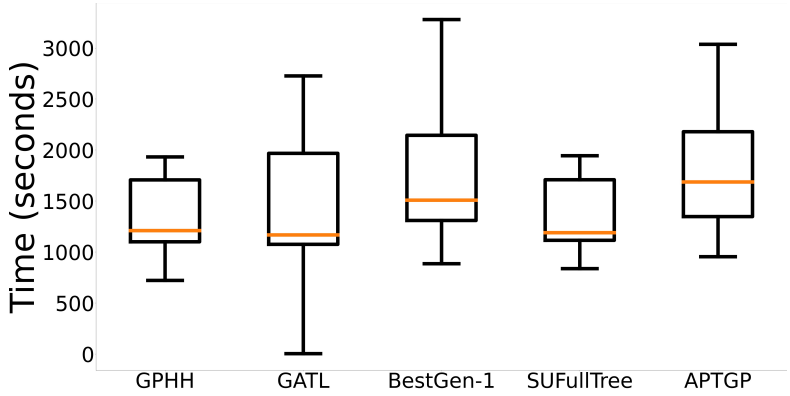


Fig. 6.6 *Training time of the compared algorithms.*

6.5.4 Training Time

Figure 6.6 presents the summary of the training time for each of the examined algorithms. According to this figure, the training time of *APTGP* is slightly longer than the existing methods. In *APTGP*, the initialisation operator and the knowledge exchange mechanism incur an additional computational cost and hence, the increased training time is expected. However, the increment in time does not hinder the applicability of the algorithm and, considering its superior performance, the additional cost can be considered acceptable.

6.5.5 Further Analysis

Component Analysis

APTGP consists of the following main novel components:

1. Initialisation with the transferred knowledge;
2. An auxiliary population that is evolved alongside the main population;
3. A knowledge exchange mechanism that promotes diversity and quality of the populations;

4. A surrogate model that allows the algorithm to evaluate more individuals; and

5. A duplicate removal mechanism that removes redundant individuals when exchanging immigrants between populations.

In order to investigate the contributions of each component to the effectiveness of *APTGP*, we conducted additional experiments with the following versions of *APTGP* in which one or more of the components were disabled.

- *APTGP*-NT: the *APTGP* with No knowledge Transfer in initialisation. It randomly generates the initial GP population for the target instance.

- *APTGP*-NA: the *APTGP* with No Auxiliary population. After the initialisation with transfer, it runs the traditional GP for the target instance.

- *APTGP*-SE: the *APTGP* with a Simple Exchange scheme, which replaces all the low-quality individuals with the immigrants from the other population;

- *APTGP*-ND: the *APTGP* with No Duplicate removal when initialising *APTGP*;

- *APTGP*-SA: the Surrogate Assisted *APTGP*, which has no explicit auxiliary population, but generates an extra 1024 offspring from the main population in each generation as the auxiliary population.

All the variants of *APTGP* except *APTGP*-NA generates 2048 offspring in each generation, evaluates 1024 offspring by the actual evaluation, while the other 1024 offspring by the KNN surrogate model. *APTGP*-NA uses a traditional GPHH process in the target instance, which generates and evaluates 1024 offspring in each generation.

Table 6.4 presents the test performance of *APTGP* and different variants in all the transfer scenarios. For each scenario (row), the entry with the best mean value is highlighted in boldface. We also conducted the Friedman test, and show the ranks of each compared algorithm and the p-value at the bottom of the table. As can be seen, *APTGP* has the best test performance (lowest rank of 1.68) and the p-value is close to zero, indicating that the differences among the compared algorithms are statistically significant. For post-hoc pairwise comparison, the adjusted p-values of the Conover pairwise comparisons [49] are given in Table 6.5.

Table 6.4: Test performance of 30 independent runs of *APTGP* and its variants (mean $\pm$ std).

Scn.	APTGP-NT	APTGP-NA	APTGP-SE	APTGP-ND	APTGP-SA	APTGP
1	91.1±0.1	91.1±0.1	91.1±0.1	91.1±0.1	91.1±0.1	**91.1±0.0**
2	544.7±5.2	554.3±9.4	556.8±10.2	547.4±7.2	546.7±9.0	**542.9±10.7**
3	595.8±6.4	604.6±9.2	604.2±9.4	**594.6±8.4**	596.0±10.1	595.0±9.7
4	**627.8±5.4**	647.8±13.8	648.4±16.6	634.0±7.7	633.0±9.8	632.6±7.3
5	58.1±0.1	58.2±0.1	58.3±0.2	58.1±0.1	58.1±0.1	**58.1±0.1**
6	417.7±7.0	424.3±9.5	426.0±8.9	419.8±7.6	418.4±7.6	**417.6±7.2**
7	**425.0±5.5**	430.6±7.7	432.3±7.4	427.8±5.5	428.2±5.6	427.6±5.8
8	425.8±4.6	431.8±5.5	431.0±6.8	423.6±7.6	425.1±7.0	**423.3±5.5**
9	572.6±3.0	576.3±3.0	578.1±3.8	573.4±3.9	574.7±11.0	**572.0±4.6**
10	**335.0±4.4**	337.4±2.0	338.0±1.8	335.3±3.4	335.3±3.5	335.1±4.0
11	341.1±3.9	343.1±3.9	345.4±4.4	341.8±5.1	**339.9±4.1**	340.6±4.1
12	**544.7±5.2**	550.6±8.2	557.4±11.8	549.0±7.0	545.0±7.6	544.8±8.0
13	595.8±6.4	604.0±9.3	604.7±11.2	596.1±9.8	595.6±9.2	**595.4±6.2**
14	**627.8±5.4**	642.0±11.6	649.8±13.1	634.5±8.3	631.4±9.1	630.9±5.7
15	335.0±4.4	337.5±2.1	337.9±3.3	334.5±4.0	335.2±3.1	**334.4±3.2**
16	442.2±4.0	444.7±5.6	443.5±7.6	442.5±4.3	**439.7±8.5**	441.4±5.6
17	**317.0±3.4**	321.0±5.1	321.8±5.0	318.0±4.3	317.6±4.3	317.1±3.8
18	354.5±3.8	356.9±4.2	356.4±3.8	352.3±3.0	**352.0±4.0**	352.2±2.4
19	355.9±2.9	359.1±5.5	358.5±5.0	357.3±3.8	355.9±4.1	**354.8±2.6**
20	356.8±1.6	358.0±1.3	358.3±1.1	356.9±1.5	**356.6±2.3**	356.6±1.2
21	337.2±5.3	338.1±5.0	340.5±3.4	336.7±5.8	334.2±5.6	**332.6±5.9**

Continuation of Table 6.2

Scn.	APTGP-NT	APTGP-NA	APTGP-SE	APTGP-ND	APTGP-SA	APTGP
22	349.8±1.8	350.6±3.2	350.9±2.3	350.8±2.9	**349.0±2.9**	349.7±2.4
23	355.6±0.1	356.4±1.5	356.7±1.7	**355.6±0.1**	355.8±1.0	355.7±0.7
24	310.9±0.8	308.8±2.9	309.7±1.3	309.1±2.6	**307.8±3.1**	307.9±4.0
25	389.0±0.1	389.1±0.1	389.1±0.2	**389.0±0.1**	389.1±0.1	389.0±0.1
26	359.0±3.1	362.0±2.6	362.8±2.6	361.1±3.1	359.5±3.9	**357.8±4.9**
27	335.7±3.9	339.9±6.8	340.3±6.1	337.2±8.5	335.5±4.6	**335.1±4.6**
28	379.8±5.2	382.9±6.6	385.1±5.3	379.9±8.8	379.7±6.2	**378.3±6.5**
29	382.1±2.0	381.3±4.0	383.8±3.5	381.4±3.9	**378.9±5.2**	381.0±2.7
30	350.7±2.3	351.1±1.1	351.0±1.6	350.8±1.6	350.8±2.2	**349.9±3.2**
31	323.5±4.9	324.0±4.6	323.9±4.5	320.6±5.6	321.2±5.2	**320.1±4.7**
32	442.2±4.0	439.6±6.3	442.6±4.4	439.3±6.3	**438.9±6.7**	439.9±6.4
33	448.0±0.5	448.5±1.3	448.6±1.4	448.2±1.4	**447.8±0.5**	448.0±0.6
34	382.7±3.4	384.6±5.2	385.6±4.9	384.6±5.5	**381.1±4.1**	382.9±4.6
35	367.8±0.9	369.2±2.9	369.1±2.5	367.8±2.2	367.7±2.0	**367.1±1.9**
36	**318.5±1.2**	322.5±3.8	322.9±4.2	320.3±2.2	320.2±1.9	319.5±1.7
37	163.8±1.3	165.1±1.4	165.6±1.7	164.5±1.3	**163.7±1.4**	163.8±1.4
38	366.6±5.7	374.7±7.6	375.5±7.3	369.5±6.0	369.8±6.4	**366.6±5.9**
39	**407.7±7.1**	412.4±7.1	415.7±7.1	411.6±5.6	409.6±6.8	409.0±8.3
40	341.1±3.9	346.8±5.9	346.7±5.1	341.9±4.9	341.4±4.8	**340.8±4.9**
41	350.7±2.3	351.6±2.6	351.7±2.5	350.5±4.2	351.1±1.7	**350.3±3.1**
42	164.6±1.0	164.7±1.6	165.4±1.6	164.5±1.5	164.0±1.4	**163.9±1.4**
43	457.4±7.4	456.9±7.8	458.9±6.2	453.8±7.4	453.8±6.8	**451.8±7.3**
44	**424.2±4.3**	426.0±2.9	426.1±3.3	425.9±2.3	424.7±3.9	424.3±4.3
45	**496.2±3.4**	497.7±3.4	497.3±3.9	496.5±3.7	496.8±2.7	496.5±3.4
Rank	2.62	5.1	5.72	3.37	2.6	1.68
Friedman's p-value						**1.11e-16**

From the table, it can be seen that *APTGP* significantly outperformed *APTGP*-NA, *APTGP*-SE and *APTGP*-ND. The advantage of *APTGP* over *APTGP*-NA indicates that without using the auxiliary population in the search process, the performance of *APTGP* degrades significantly. This is an indicator of the important role that the auxiliary population plays in

Table 6.5: Post-hoc comparison of the *APTGP* and its variants.

	APTGP-NA	APTGP-SE	APTGP-ND	APTGP-SA	APTGP
APTGP-NT	**1.7e-08**	**7.8e-13**	0.07	0.95	**0.02**
APTGP-NA	–	0.079	**7.8e-05**	**1.5e-08**	**3.7e-14**
APTGP-SE	–	–	**2.2e-08**	**7.40e-13**	**4.1e-19**
APTGP-ND	–	–	–	0.067	5.6e-05
APTGP-SA	–	–	–	–	**0.02**

the algorithm.

On the other hand, using an auxiliary population with a simple knowledge exchange mechanism is not enough for achieving significant improvements, since *APTGP*-SE and *APTGP*-NA are statistically similar and *APTGP*-SE was ranked even worse than *APTGP*-NA.

The advantage of *APTGP* over *APTGP*-ND indicates that the act of duplicate removal increases the amount of useful knowledge that is inherited from the source which in turn, highlights the negative impact that the presence of duplicates in the source knowledge can have on the performance on the target instance.

Compared with *APTGP*-NT, the p-value was 0.02, which is very close to the significance level the Bonferroni correction ($0.05/4 = 0.0125$ for the four comparisons between APTGP and other four algorithms). From Table 6.4, we can also see that the rank of *APTGP*-NT is 2.62, closer to the rank of *APTGP* than *APTGP*-NA, *APTGP*-SE and *APTGP*-ND, and obtained the best test performance on 10 scenarios. This shows that even without the initial knowledge transfer, *APTGP* can still be quite effective due to the use of auxiliary population for knowledge transfer during the search process. Meanwhile, the considerable difference between *APTGP* and *APTGP*-NT (rank 1.68 vs rank 2.62) shows the effectiveness of the knowledge transfer in the initial population of the target instance.

APTGP-SA showed the closest test performance to *APTGP*. Its rank was 2.6, and the p-value was 0.02, slightly larger than the corrected signif-

icance level of 0.0125. It achieved the best test performance on 11 scenarios. However, we can still see the advantage of *APTGP* over *APTGP*-SA, as *APTGP* still has a much better rank (1.68 vs 2.6) and performed the best on 21 scenarios. This verifies the effectiveness of using the auxiliary population instead of directly generating extra offspring and evaluate using the surrogate model.

It is also interesting to note that *APTGP*-NA and *APTGP*-SE had very similar performances. Both algorithms transfer unique individuals, but *APTGP*-SE additionally performs a simple knowledge exchange that does not consider the possible presence of duplicates. As a result, not only the algorithm does not benefit from the exchange of knowledge, its slightly inferior performance indicates that the exchange was even harmful. This could be attributed to possibility that the exchange mechanism may have increased duplicates in the main population. The damage of this effect is to the point that *APTGP*-NT, which does not utilise any transferred knowledge, has a significantly better performance than *APTGP*-SE. On the other hand, *APTGP*-NT performs rather similarly to *APTGP*-NA.

Phenotypic Diversity

The GPHH method for solving UCARP is known to suffer from the loss of population diversity during the evolutionary process [11, 9]. This property can negatively impact the quality of knowledge transfer since the transfer of duplicates reduces the amount of useful knowledge that can be transferred and may run into the risk of trapping the search process in poor local optima. This understanding was one of the key considerations for designing *APTGP*. In this subsection, we investigate how effective *APTGP* was in maintaining and increasing the diversity during the search process. For this purpose, we employ the entropy measure [32] for calculating the population diversity.

Figure 6.7 presents the distribution of the entropy of the populations during the evolutionary process of the compared algorithms. For each

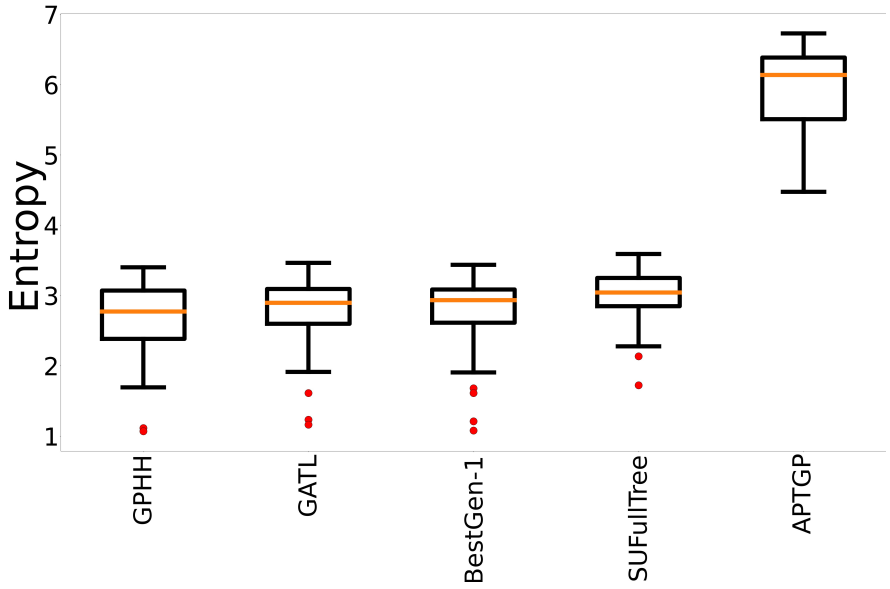


Fig. 6.7 *Population entropy of the compared algorithms.*

algorithm, the entropy of the populations in all the generations of all the 30 runs are taken into account. From Figure 6.7, we can clearly see that *APTGP* managed to reach a much higher entropy (better diversity) of the populations than the other algorithms.

To understand how the population entropy changed over the course of evolution, the average population entropy over the 30 runs is plotted in Figure 6.8 against GP generation for a few scenarios (we observed similar patterns for other scenarios as well). As can be easily seen from the figure, all the compared algorithms started with a high degree of diversity. However, as the evolution proceeded, all the other algorithms except *APTGP* lost their diversity in the population rather quickly. On the other hand, although *APTGP* also lost some of its diversity, it managed to maintain a high entropy (diversity) over time. Another interesting observation in Figure 6.8 is that the initial entropy of *APTGP* and SUFullTree is higher that other methods. This can be explained by considering the fact that both *APTGP* and SUFullTree algorithms place a high emphasis on creating a

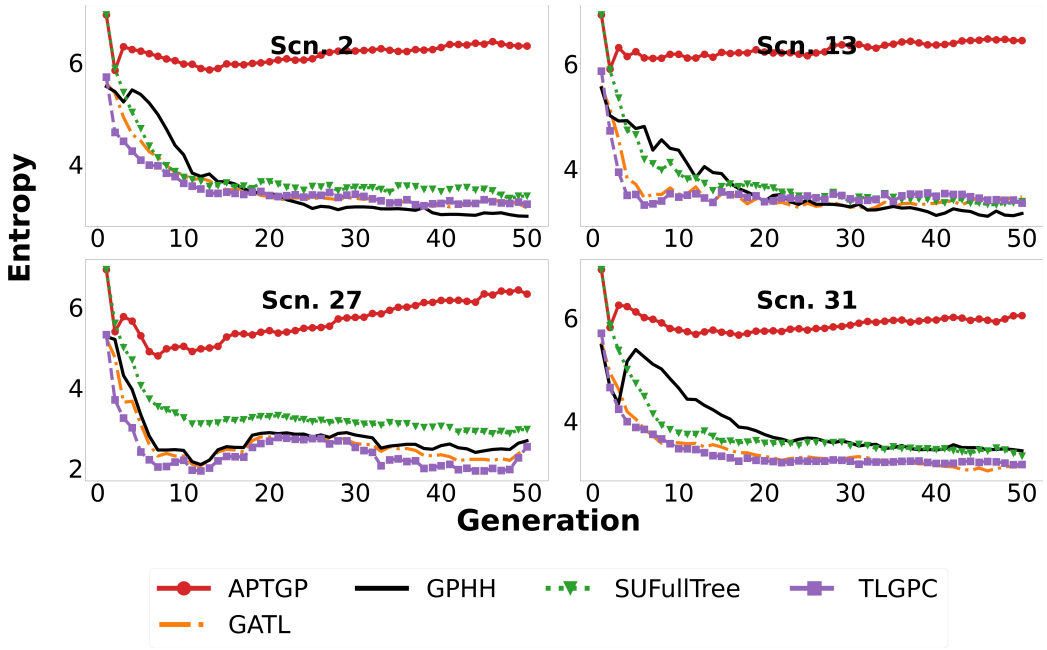


Fig. 6.8 *The curve of the population entropy of the compared algorithms for some representative scenarios.*

very diverse initial population.

6.5.6 Duplicate Removal

In the knowledge exchange phase of *APTGP* (i.e. Algorithm 6.4), when selecting individuals in a population to be replaced with the incoming immigrants from the other population, the duplicates in the population are first considered for replacement. If the population does not contain enough duplicates, then low-quality unique individuals are selected to be replaced. Figure 6.9 presents the number of duplicates selected and replaced in each population over the generations of *APTGP*, averaged over all runs on all the scenarios. As can be seen, throughout the evolutionary process, there exist a large number of duplicates in the population. The number of duplicates decrease slowly throughout the evolution as is

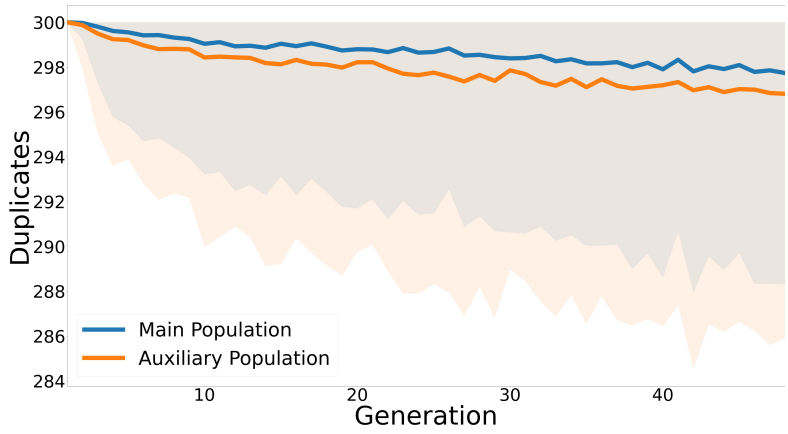
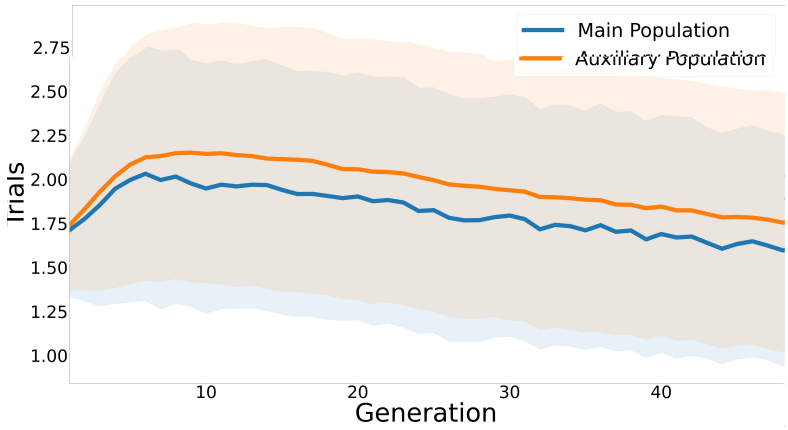


Fig. 6.9 *Average number of the removed duplicates*

suggested by the downward trend of the plot and the increased standard deviation. On average, 298.67 duplicates were replaced in each generation of *APTGP*, with a standard deviation of 7.14. Overall, we can see that there are a large amount of duplicates in the population of *APTGP*, despite the pressure of the duplicate removal mechanism in *APTGP*.

According to Algorithm 6.1, the duplicate replacement is performed at the end of each generation and after the breeding operation. Since the



(a) *Average number of trials to find a unique immigrant*

Fig. 6.10 *Dynamics of the duplicate removal mechanism.*

population diversity is maintained at a high level at the end of each generation, as discussed in Subsection 6.5.5, the breeding operator is the main reason for introducing duplicates into the population. This observation indicates the need for designing more effective breeding operators for solving UCARP.

Figure 6.9 suggests that one may remove more duplicates and achieve better performance by increasing the value of the η parameter. However, when we investigated this possibility, we did not observe any significant improvement in the results. On the contrary, there were cases in which the performance became slightly worse. One possible reason for this phenomenon is that as the value of η increases, more unique individuals are likely to be discarded in favour of the transferred immigrants. In such cases, *APTGP* does benefit from the possible improvement in diversity that immigrants could provide and it can just benefit from the exchange of the knowledge that immigrants contain. However, as the GP evolution proceeds, the quality of the population also increases. As a result, when unique individuals are replaced with the immigrants, it is more likely that the population replaces good individuals with the immigrants which are not guaranteed to be good (considering that the immigrants may go through mutation before being transferred).

Figure 6.10a presents the average number of trials (mutations) for *APTGP* to obtain a unique individual as an immigrant for knowledge exchange. As we can see from the figure, in the early generations, the algorithm is more likely to find out that the candidate immigrant is redundant in the target. However, as the algorithm proceeds, this likelihood decreases slightly because the diversity in both populations increases (as is seen in the entropy plots in Figure 6.8).

6.6 Chapter Summary

In this chapter, a novel transfer optimisation algorithm for GP, called *APTGP*, was proposed to evolve routing policies for UCARP. In *APTGP*, all the routing policies that GP has examined for solving a source problem are considered as the transferable knowledge. First, we propose to remove duplicates before using the source individuals to initialise the GP population for the target problem. Then, during the search process, the transferred knowledge is retained and evolved in a separate auxiliary population alongside the main population. The auxiliary population is evolved with a surrogate model that is learned from the main population. The main purpose of the auxiliary population is to help GP address the issue of losing its population diversity and increase its exploration capabilities. To achieve this, an elaborate knowledge exchange mechanism is devised to share high-quality and unique individuals between the main and auxiliary populations.

The effectiveness of *APTGP* has been verified by comparing with the state-of-the-art algorithms on a wide range of transfer scenarios. Our analysis demonstrated that *APTGP* managed to significantly outperform all the state-of-the-art GP algorithms with knowledge transfer. Additionally, we demonstrated that *APTGP* helped GP overcome the limitation of losing population diversity. Furthermore, we conducted a detailed set of control experiments to verify the effectiveness and the contribution of each novel component of *APTGP*.

Chapter 7

Conclusions

The focus of this thesis is on devising novel transfer optimisation algorithms for GPHH to solve UCARP. The ultimate goal of this work to improve the effectiveness of GPHH for evolving vehicle routing policies and solving a problem using the knowledge that has been gained from a previously-solved related problem. To achieve this goal, we examined the literature of transfer optimisation methods for evolutionary algorithms and identified the lack of phenotypic diversity and the presence of duplicates as the main challenges that one may face in order to perform knowledge transfer successfully for UCARP. The identification of the challenges helped us concentrate our efforts on devising transfer optimisation algorithms that could handle the challenges successfully. We evaluated the effectiveness of each proposed algorithm using the UCARP simulations with a vast set of measurements and analyses.

In this chapter, the achieved objectives of this thesis are highlighted and are followed by the main conclusions that we have drawn from our studies. Following the conclusion, a set of discussions are presented to provide deeper insights into the key issues in this research area. Finally, this chapter outlines a collection of potential research directions for future work that are based on the studies in this thesis.

7.1 Achieved Objectives

In this thesis, the following research goals were pursued and achieved.

- Chapter 3 of this thesis investigated and identified the challenges in performing effective knowledge transfer for GPHH to solve UCARP. In order to achieve this goal, we considered the literature of the transfer optimisation methods for evolutionary algorithms and upon examining the literature, it was noticed that the existing methods for GP did not evaluate the importance of a sub-tree for transfer and did not try to learn the probability distribution of the solutions. To address these gaps, we proposed two new (sub-)tree-based algorithms in which two measures were proposed for identifying the subtrees that suitable for transfer. Additionally, we proposed an algorithm for learning the probability distribution of the good source solutions. To analyse the performance of the proposed algorithms, we also considered a set of state-of-the-art methods and evaluated them for solving a large set of experiments. Our experimental results showed that the UCARP knowledge source, i.e. the set of all individuals that were discovered for solving a source problem, may contain a large number of duplicates, the majority of which belonged to the set of high-quality solutions. These results highlighted a major challenge in the way of performing successful knowledge transfer for UCARP. This discovery guided the direction of the research in this thesis and was addressed in later chapters.

- The goal in Chapter 4 was to propose a set of approaches to overcome the identified issues in Chapter 3. More specifically, Chapter 4 proposed a set of novel transfer optimisation algorithms in which the possible presence of duplicates in the knowledge source was considered and handled. First, it is speculated that if the knowledge source contains duplicates, the naïve transfer of source individuals will transfer the duplicates to the target population and this will also

afflict the search process for solving the target problem. Therefore, to overcome this issue, a collection of methods are proposed in Chapter 4 to overcome the lack of population diversity due to the transfer. After this, the chapter focuses on handling the presence of duplicates in the knowledge source more directly. In order to do so, the chapter first proposes a novel approach to detecting and removing phenotypic duplicates from the transferred knowledge source and refine it into a set of unique individuals. Because the removal of duplicates reduced the number of high-quality individuals, the chapter proposed learning a surrogate model from the transferred individuals and utilise it for creating additional high-quality unique individuals that increases the set of reusable individuals. Our experimental results showed that the removal of duplicates can improve the quality of the knowledge transfer.

- Chapter 5 proposed another novel approach to transfer optimisation for solving UCARP. The proposed algorithm also considers the possible presence of duplicates in the knowledge source. Additionally, the proposed algorithm in this chapter also utilises a method for removing the duplicates and reusing the genetic materials of the unique individuals for initialisation of GP for solving the target problem. However, the main goal in this chapter is to utilise the transferred knowledge after the initialisation phase. Although the current literature of transfer optimisation entails a set of algorithms that reuse the transferred genetic materials during the search for solving the target problem, the focus of these existing algorithms is on the genotypic features of the knowledge source while the final solutions for a UCARP instance, i.e. the vehicle routes, belong to the phenotypic space. As a result, the existing knowledge transfer methods, which reuse the transferred knowledge after initialisation, are unlikely to perform effectively for UCARP, as was depicted in Chapter 3 for the TLPC algorithm [116]. Consequently, Chapter 5 of this

thesis proposed a novel algorithm that transfers the phenotypic information of a UCARP knowledge source and equips GP search process with this information. The transferred knowledge allows GP crossover and mutation operators to avoid creating individuals that are not likely to have a good performance for the target problem and by doing so, it helps GP spend its computational budget on evaluating more novel solutions for the target problem. The experimental results showed that prohibiting the search process from recreating the source solutions allows GP discover and evaluate new solutions and consequently, increase the search performance significantly.

- In Chapter 6, another novel approach to knowledge transfer is proposed for solving UCARP. In accordance with the findings in Chapter 3, this chapter also takes into account the possibility that the transferred set of individuals may contain phenotypic duplicates and hence, handles the issue accordingly. However, the main consideration in Chapter 6 is that the issue of duplicates can happen when GP is used to solve the target problem. Accordingly, the main goal in this chapter is to utilise the transferred knowledge in such a way that helps GP avoid being afflicted with the issue of low population diversity. For this purpose, this chapter proposed an algorithm in which the transferred individuals are maintained in a separate auxiliary population. The auxiliary population is evolved alongside the main population and with the help of a surrogate model. The main purpose of the auxiliary population is to help the main population maintain its diversity and to achieve this, an elaborate mechanism was proposed for exchanging knowledge between the main and the auxiliary populations. In the design of the proposed knowledge exchange mechanism, the focus was placed on the novelty of the knowledge that is being transferred from one population to another; that is, the exchange mechanism tries to ensure that individuals are sent to the receiving population that are not already present

in that population. Also, when the transfer mechanism allows the transfer of an individual, the receiving population replaces one of its duplicates in favour of the incoming individual.

7.2 Main Conclusions

This section describes the main conclusions of this thesis, which are drawn from the four major contribution chapters, i.e., Chapter 3 to Chapter 6.

7.2.1 The Applicability of Transfer Optimisation for Solving UCARP

As it was reviewed in Chapters 1 and 2, in real-world scenarios UCARP instances are subject to change over time frequently. This feature promotes the need for effective transfer optimisation methods that can extract reusable common knowledge from a previously solved problem to facilitate the search process for solving the related new problem. However, to the best of our knowledge, there exist no study on the applicability of transfer optimisation for solving UCARP. As a result, one of the main contributions of this thesis is its investigation of the potentials that transfer optimisation can have for solving UCARP and the identification of the challenges that may exist in this direction.

In the context of transfer optimisation and transfer learning, two of the fundamental questions are the questions of what the transferable knowledge is and how the knowledge should be transferred [96, 151, 186, 82]. In this context, these question implies the need to know what the nature of the transferable knowledge is, how the knowledge should be extracted from the source and how it should be reused. Accordingly, to be successful at knowledge transfer, it is of significant importance to know what features of the knowledge source can impact the quality of knowledge transfer.

To answer these questions, we selected a set of existing transfer optimisation algorithms and examined their effectiveness for solving UCARP. Upon our examination, we realised that none of the existing methods were able to demonstrate significant improvement over the performance of GPHH without any knowledge transfer. Furthermore, in our experiments, the algorithms that considered the whole GP tree as the transferable knowledge performed slightly better than the algorithms that just transferred sub-trees, but the difference was not significant.

Additionally, we realised that a set of shortcomings in the existing algorithms. Suspecting that the unsatisfactory performances of the existing methods could be due to these shortcomings, we also proposed a set of novel algorithms to address these shortcomings. However, our experiments revealed that the proposed algorithms also could not perform better than the existing methods. This result prompted us to investigate the reasons for this bad performances of the transfer optimisation algorithms deeper.

Accordingly, in this work, we identified that the GP search process for solving a UCARP instance is susceptible to losing its phenotypic diversity due to the presence of phenotypic duplicates. When the duplicates occupy a significant portion of the population, the set of useful knowledge becomes limited. Therefore, on one hand, the transfer of duplicates, if allowed, will amount to the transfer of redundant genetic materials. On the other hand, if the duplicates are removed, the knowledge set may not contain adequate amount of reusable individuals. Therefore, we identified the presence of duplicates in UCARP knowledge source as one of the main challenges towards successful transfer of knowledge for solving UCARP.

7.2.2 Diversity-Driven Knowledge Transfer

It is possible that a GP knowledge source may not contain adequate amount of reusable materials to be utilised for transfer. This could happen due to

various reasons, including the abundant presence of duplicates. In this case, the limited amount of transferable knowledge clearly reduces the effectiveness of the transfer and the existing transfer optimisation algorithms are not equipped with any solution for this problem.

One of the main contributions of this thesis is the development of the methods that can handling the potential issue of the presence of duplicates in the knowledge source. In this thesis, we proposed two category of algorithms for handling this issue. In the first category, the knowledge source is not modified but instead, it is used as it is. Accordingly, the potential problem of transferring the duplicates is handled afterwards. In the second category, the duplicates are removed from the knowledge source, before reusing the knowledge source. The removal of the duplicates can lead to a smaller knowledge set that may contain few high-quality individuals. To handle this, in our proposed approaches, we utilised the knowledge from a solved source to train a surrogate model that can estimate the fitness of routing policies for solving the source problem. Although the surrogate model can only provide an approximation of the fitness value, its fast and computationally inexpensive nature allows evaluating a large number of individuals, which could not be achieved if the actual fitness evaluation of UCARP based on simulation was utilised. Consequently, the learned surrogate model allows augmenting the set of transferred source individuals with newly generated policies. Although these new individuals are not evaluated within a complete simulation to assess their actual fitness, the surrogate model can provide a good approximation for it. It should be noted that in the context of transfer optimisation, the actual fitness value loses its prominent importance. This is because, in majority of the available algorithms, including the algorithms in this thesis, the actual fitness is used for detecting if the individual is a good candidate for transfer or not and an approximate fitness value can also convey this information.

Another benefit of using a surrogate model is that the fast fitness ap-

proximation offered by it allows assessing a large number of individuals. This feature, in turn, allows us to discard phenotypic duplicates and assess new individuals until enough number of unique yet high-quality individuals are created for transfer.

Consequently, a major contribution of this thesis is the proposal of two categories of algorithms for handling the presence of duplicates in the knowledge source. Through our experimental studies, we confirmed that trying to handle the negative impacts of duplicates in the knowledge source after the transfer will not increase the performance of the knowledge transfer. On the other hand, removing the duplicates before the transfer can decrease the number of high-quality source individuals. To overcome this issue, we proposed training a surrogate model with the source individuals and use it for creating high-quality individuals that can augment the knowledge pool.

7.2.3 Knowledge-Guided Search

In the existing literature of transfer optimisation methods for GP, the transferred knowledge is either not utilised effectively after the initialisation phase or it is used in a limited fasion. Additionally, in cases that the knowledge is used after initialisation, the focus is mostly placed on reusing the genetic information. This focus on the reuse of genetic materials becomes problematic in case of problems, such as UCARP, in which the genotypic space is different from the phenotypic space and the relationship between the spaces is not trivial to comprehend.

Consequently, a major contribution of this thesis is the proposal of a novel algorithm for the transfer of the phenotypic characteristics of GP individuals that can be used effectively during the search for solving a target problem. The proposed approach for reusing the transferred phenotypic information is significantly different from the existing transfer optimisation algorithms. In our approach, the transferred information is used for

creating a tabu list and the search process for solving the target problem is banned from creating individuals that have the same phenotypic behaviour as any item in the transferred tabu list. This prevents GP from repeating the creation of solutions that are unlikely to be optimal and helps it focus on discovering new solutions. Because of the relatedness of the source and target problems, this approach can be interpreted as an effort to prevent GP from searching the regions of the search space that are less likely to solutions, which in turn, results in guiding GP to explore and discover new regions of the search space.

7.2.4 Knowledge Transfer with Auxiliary Population

As it was discussed in Chapter 3 of this thesis, the search process for solving a UCARP instance is to subject to lose the phenotypic diversity of its population. One consequence of this phenomenon is the negative effect it can have on the quality of knowledge transfer. However, a more immediate consequence of this issue is that the target problem can also be afflicted the same way and it may lose its diversity. Even if GP is initialised with a diverse set of individuals, the diversity can be lost after a few generations.

Accordingly and as a major contribution of this thesis, we proposed a new algorithm that uses the transferred knowledge to help GP maintain its population diversity and not be afflicted with the presence of duplicates. For this purpose, after using the unique high-quality individuals of the source to initialise GP for solving the target problem, the transferred individuals are maintained in a separate auxiliary population and are evolved with a surrogate model. This evolution helps the auxiliary population become more adapted towards solving the target problem. As a result, when knowledge exchange is performed between the main and the auxiliary population, the immigrants from the auxiliary population are not very unsuitable and unfit in the main population. The exchange of immigrants between the two population is designed so that immigrants replace

the duplicates and in this way, help GP discard duplicates in favour of unique new individuals.

7.3 Further Discussions

Section 7.2 presented a summary of the main findings of this thesis. In this section, the general issues covered in this thesis and related topics are discussed further.

7.3.1 Relatedness of Problems

In the context of transfer optimisation, the act of knowledge transfer is justifiable if and only if there exist a good degree of similarity between the source and target problems and there exists a clear relationship between the problems that justifies the nature of knowledge [96]. Otherwise, the act of knowledge transfer may lead to negative transfer and hurt the search process for solving the target problem.

For the case of UCARP, to the best of our knowledge, there does not exist any concrete method for measuring the degree of similarity between any given two problems and, as a result, measuring the similarity of problems remains one of the main challenges to successful transfer of knowledge for solving UCARP. Mei et al. [163] demonstrated that any change in a characteristics of a problem, e.g. number of vehicles, can lead to a problem that is significantly different from the original one. However, the degree of difference (i.e. lack of similarity) is not clear. In the absence of a concrete similarity measure, we devised an approximate measure based on the correlation between the performance of 1000 random routing policies for solving the problems in Section 3.5. However, this approach is an approximate measure that depends on random individuals. Additionally, it is not clear if the fitness value is a good representative that could cover all features of the problems. As a result of this, in this thesis, we utilised

this approximate measure with caution and used just as a hint.

As a result, measuring the exact degree of similarity between UCARP problems is an open subject that needs further research. In the context of transfer optimisation, one immediate benefit of having an accurate degree of similarity is that it illuminates if the source and target problems are related and hence, prevents from negative transfer between unrelated problems. Another advantage of this measure is that it could allow more developing more elaborate algorithms that could regulate the amount of knowledge transfer between problems based on the degree of similarity between them, especially when there are more than one knowledge source to transfer from, as is discussed in Section 7.3.2.

7.3.2 Multiple Sources of Knowledge

In this work, all the devised algorithms, and all the experiments to verify them, considered only a single knowledge source, that is, a single solved problem to transfer from. An interesting direction for future work could be the case of having a large number of sources when only some of them (possibly a very small fraction of sources) are related to the target while others are not. A possible approach to this problem is to modify the proposed algorithms so that the similarity of the source problem to the target problem is assessed before performing the act of knowledge transfer and, discard the sources that have low similarity to the target problem. One other possible approach can be to measure the similarity of different source problems and base the amount of knowledge that is transferred from each source on the similarity of that source to the target problem. As was discussed in Section 7.3.1, this also highlights the need for having a more accurate measure of similarity between UCARP problems.

7.3.3 Surrogate Models

In Chapters 4 of this work, we made extensive use of surrogate models. In our work, these models were trained on the transferred source knowledge and then, utilised to estimate the fitness of new individuals. In this regard, one shortcoming of this approach is that since the models are trained based on the individuals that were found for solving the source problem, they are trained based on the areas of the search space that were investigated for solving the source problem. Consequently, when a new individual from the investigated regions is assessed with the model, the performance of the model is satisfactory. However, when an individual from an uninvestigated region is encountered, the performance of the surrogate model is very unpredictable. As a result, if the individual has a good fitness, the surrogate model may not be able to detect it. This issue can potentially damage the quality of knowledge transfer and it has the potential to be investigated further. Furthermore, due to their simplicity, the surrogate models in this work were based on the KNN method. However, this is not the only surrogate type and it will be interesting to investigate the effectiveness of other surrogate types for transfer optimisation.

7.4 Future Work

7.4.1 Multi-task Learning for UCARP

Multi-task Learning, MTL, is the approach of solving multiple related problems simultaneously and together [95, 22, 76]. In MTL approaches, the goal in solving the problems together is that during the search process for solving each problem, the processes can share the common knowledge that they find between each other and in this way, increase the effectiveness and efficiency of the search for each problem, compared to when each search is performed independently. Accordingly, similar to transfer optimisation, the act of knowledge sharing is an important component of

MTO. Although MTL algorithms have shown to be very effective in solving an extensive problems from different fields [72, 140, 61, 188], to the best of our knowledge, the work by Ardeh et al. [12] is only work that has performed MTL for UCARP. The experimental results obtained by Ardeh et al. indicated that MTL is a promising area of research that deserves further investigations.

7.4.2 Cultural Algorithms

In Chapter 6, we proposed a transfer optimisation algorithm in which the transferred knowledge is evolved alongside the main population and supports its evolution. As was also discussed in Section 6.4.6, this approach has similarities with the category of cultural algorithms [155, 51]. Although the properties of our proposed algorithm in Chapter 6 differentiates the algorithm from the category of cultural algorithms, the good performance of our algorithm and its existing similarities with cultural methods indicate that cultural algorithms are viable research area that deserve further investigation. Accordingly, it would be interesting to study the potential of developing novel cultural algorithms for solving UCARP, especially, in the context of transfer optimisation and multi-task learning. As an example, one particular case in this direction would to be consider the common knowledge into the belief space of cultural algorithms and use the facilities that the network provide to evolve the common knowledge and expand the collection of reusable common knowledge.

7.4.3 Population Diversity

In Chapter 3 of this thesis, it was discovered that lack of diversity in the knowledge source was one of the main challenges in performing effective transfer optimisation for UCARP. The concept of diversity was one of the main areas of focus throughout Chapters 4–6 and our experimental results indicated its contribution to the performance of the search process. How-

ever, to the best of our knowledge, the issue of population diversity has not been investigated before for UCARP. Our experimental results, especially in Chapter 6, indicate that improving the population diversity has a positive correlation with the performance of GPHH for solving UCARP. However, this area has not been investigated rigorously. Particularly, investigating the effect of promoting the phenotypic and genotypic diversity on the search performance, the correlation between phenotypic and genotypic diversity and the effect of utilising different phenotypic measures on the quality of GPHH results would be very interesting and fruitful.

7.4.4 Knowledge Transfer for Non-GP Approaches to Solving UCARP

In this thesis, the main approach for solving UCARP was the approach of using GP as a hyper heuristic. Accordingly, all the transfer optimisation methods in this work were based the GPHH method. However, other evolutionary algorithms, such as EDA [223] and memetic search [224], have been also devised for solving UCARP. To the best of our knowledge, transfer optimisation has not been researched in the context of these algorithms. Accordingly, one potential direction worth investigation is developing transfer optimisation algorithms for such methods and compare their performances to the GPHH-based algorithms. Additionally, transferring knowledge from a source problem that is solved with GPHH to the non-GPHH methods and vice versa is another interesting subject of research.

Furthermore, recently neural network-based reinforcement learning algorithms have recently been successful at solving routing problems [194, 24, 175, 56]. However, to the best of our knowledge, no attempt has ever been made at solving UCARP with any reinforcement learning algorithms. One potential approach would be to consider a neural network to represent the task priority function, which is represented by GP trees in the

GPHH approach, and devise appropriate training algorithms. Investigating the transfer learning potential of the neural-based approach and devising algorithms for transferring knowledge from/to GPHH algorithms are another interesting lines of research in this direction.

7.4.5 Knowledge Transfer for Other Combinatorial Problems

The knowledge transfer algorithms that were developed in this work were designed for GPHH to solve UCARP. However, GPHH is the state-of-the-art approach for other combinatorial optimisation problems (e.g. DFJSS) [59]. Hence, one potential direction for future work is to adapt the proposed methods for solving other combinatorial problems. In this regard, it should be noted that the algorithms were designed for solving UCARP, and therefore, they may entail steps that are specific to UCARP (e.g. the phenotypic characterisation of routing policies with decision situations). Consequently, the first step in applying the proposed algorithms to other problems will be to identify the UACRP-specific components of the proposed algorithms and replace them with relevant steps for solving other combinatorial problems. Another important consideration is the fact that the design of the proposed algorithms was heavily influenced by the observation that the GPHH process for solving UCARP suffers from the lack of population diversity. As a result, another step in applying the proposed algorithms to other problems is to identify how the lack of diversity in the GP population, if present, impacts the performance of GP.

7.4.6 The Effect of Population Size

Population size is an important parameter of the genetic algorithm [108]. Previous studies have investigated the impact of this parameter on the performance of evolutionary algorithms [198, 211, 195]. In general, it can be concluded that the effect of the population on the performance of EC

algorithms is complicated [65, 201, 222]. On the one hand, if the population size is small, the search capability of GP will be restricted and hence, the performance will be impacted negatively. On the other hand, and in the case of UCARP, having a large population is not likely to improve the performance significantly because the majority of the population will be occupied with duplicates.

However, if some mechanisms are devised to control the presence of duplicates, then two scenarios are possible, the first of which is that the computational budget (i.e. number of fitness evaluations) is fixed; that is, the overall number of fitness evaluations is the same as in our experiments. In this case increasing the population size is likely to reduce the effectiveness of the breeding operators and result in worse performance because GP individuals will go through fewer generations of evolution. The second possible scenario is to allow GP to have an increased computational budget. In this case, since the presence of duplicates are controlled, the computational budget will not be wasted on evaluating duplicates and hence, the performance of the algorithm is likely to increase.

Nevertheless, as the previous studies indicated, the effect of population size on the performance of GP is complicated. Consequently, it will be beneficial to investigate the effect in the context of GP for solving UCARP and understand how it will impact the performance of the knowledge transfer mechanisms.

Bibliography

[1] ABBASIFARD, M. R., GHAHREMANI, B., AND NADERI, H. A Survey on Nearest Neighbor Search Methods. *International Journal of Computer Applications 95*, 25 (jun 2014), 39–52.

[2] AKAY, B., KARABOGA, D., GORKEMLI, B., AND KAYA, E. A survey on the Artificial Bee Colony algorithm variants for binary, integer and mixed integer programming problems. *Applied Soft Computing 106* (jul 2021), 107351.

[3] AL-HELALI, B., CHEN, Q., XUE, B., AND ZHANG, M. A Hybrid GP-KNN Imputation for Symbolic Regression with Missing Values. In *AI 2018: Advances in Artificial Intelligence* (Cham, 2018), T. Mitrovic, B. Xue, and X. Li, Eds., Springer International Publishing, pp. 345–357.

[4] AL-HELALI, B., CHEN, Q., XUE, B., AND ZHANG, M. Multi-Tree Genetic Programming with New Operators for Transfer Learning in Symbolic Regression with Incomplete Data. *IEEE Transactions on Evolutionary Computation* (2021), 1–1. doi: 10.1109/TEVC.2021.3079843.

[5] ALLEN, S., BURKE, E. K., HYDE, M., AND KENDALL, G. Evolving Reusable 3D Packing Heuristics with Genetic Programming. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation* (New York, NY, USA, 2009), GECCO '09, ACM, pp. 931–938.

[6] ALVAREZ, I. M., BROWNE, W. N., AND ZHANG, M. Reusing learned functionality to address complex boolean functions. In *Simulated Evolution and Learning* (2014), vol. 8886, Springer International Publishing, pp. 383–394.

[7] ALVAREZ, I. M., BROWNE, W. N., AND ZHANG, M. Compaction for code fragment based learning classifier systems - Redux. *2016 IEEE Congress on Evolutionary Computation, CEC 2016* (2016), 2217–2224.

[8] AMPONSAH, S. K., AND SALHI, S. The investigation of a class of capacitated arc routing problems: the collection of garbage in developing countries. *Waste Management 24*, 7 (2004), 711–721.

[9] ARDEH, M. A., MEI, Y., AND ZHANG, M. A Novel Genetic Programming Algorithm with Knowledge Transfer for Uncertain Capacitated Arc Routing Problem. In *PRICAI 2019: Trends in Artificial Intelligence*. Springer, 2019, pp. 196–200.

[10] ARDEH, M. A., MEI, Y., AND ZHANG, M. Genetic Programming Hyper-heuristic with Knowledge Transfer for Uncertain Capacitated Arc Routing Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (2019), ACM, pp. 334–335.

[11] ARDEH, M. A., MEI, Y., AND ZHANG, M. Transfer Learning in Genetic Programming Hyper-heuristic for Solving Uncertain Capacitated Arc Routing Problem. In *Congress on Evolutionary Computation* (Wellington, New Zealand, 2019), pp. 49–56.

[12] ARDEH, M. A., MEI, Y., AND ZHANG, M. A Novel Multi-Task Genetic Programming Approach to Uncertain Capacitated Arc Routing Problem. In *Proceedings of the ACM Genetic and Evolutionary Computation Conference* (2021), ACM, pp. 759–767.

[13] ARDEH, M. A., MEI, Y., AND ZHANG, M. Surrogate-Assisted Genetic Programming with Diverse Transfer for the Uncertain Capacitated Arc Routing Problem. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2021).

[14] ARDEH, M. A., MEI, Y., AND ZHANGZ, M. Diversity-driven Knowledge Transfer for GPHH to Solve Uncertain Capacitated Arc Routing Problem. In *Proceedings of the IEEE Symposium Series on Computational Intelligence* (2020), pp. 2407–2414.

[15] ARYA, S., MOUNT, D. M., NETANYAHU, N. S., SILVERMAN, R., AND WU, A. Y. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM 45*, 6 (nov 1998), 891–923.

[16] AZARI, S., ZHANG, M., XUE, B., AND PENG, L. Genetic Programming for Preprocessing Tandem Mass Spectra to Improve the Reliability of Peptide Identification. In *2018 IEEE Congress on Evolutionary Computation (CEC)* (2018), IEEE, pp. 1–8.

[17] AZZOUZ, A., ENNIGROU, M., JLIFI, B., AND GHÉDIRA, K. Combining Tabu search and genetic algorithm in a multi-agent system for solving flexible job shop problem. In *Proceedings of Special Session - Revised Papers, 11th Mexican International Conference on Artificial Intelligence 2012: Advances in Artificial Intelligence and Applications, MICAI 2012* (2012), pp. 83–88.

[18] BACK, T., FOGEL, D. B., AND MICHALEWICZ, Z. *Handbook of Evolutionary Computation*, 1st ed. IOP Publishing Ltd., GBR, 1997.

[19] BADER-EL-DEN, M., AND POLI, R. Generating SAT Local-Search Heuristics Using a GP Hyper-Heuristic Framework. In *Artificial Evolution* (Berlin, Heidelberg, 2008), N. Monmarché, E.-G. Talbi, P. Col-

let, M. Schoenauer, and E. Lutton, Eds., Springer Berlin Heidelberg, pp. 37–49.

[20] BALDACCI, R., AND MANIEZZO, V. Exact methods based on node-routing formulations for undirected arc-routing problems. *Networks 47*, 1 (2006), 52–60.

[21] BALI, K. K., GUPTA, A., FENG, L., ONG, Y. S., AND SIEW, T. P. Linearized domain adaptation in evolutionary multitasking. In *Proceedings of IEEE Congress on Evolutionary Computation* (jul 2017), pp. 1295–1302.

[22] BALI, K. K., ONG, Y. S., GUPTA, A., AND TAN, P. S. Multifactorial Evolutionary Algorithm with Online Transfer Parameter Estimation: MFEA-II. *IEEE Transactions on Evolutionary Computation 24*, 1 (feb 2020), 69–83.

[23] BARTOLINI, E., CORDEAU, J.-F., AND LAPORTE, G. An Exact Algorithm for the Capacitated Arc Routing Problem with Deadheading Demand. *Operations Research 61*, 2 (2013), 315–327.

[24] BELLO, I., PHAM, H., LE, Q. V., NOROUZI, M., AND BENGIO, S. Neural Combinatorial Optimization with Reinforcement Learning. In *5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings* (nov 2016), International Conference on Learning Representations, ICLR.

[25] BENJAMINI, Y., AND HOCHBERG, Y. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological) 57*, 1 (1995), 289–300.

[26] BEULLENS, P., MUYLDERMANS, L., CATTRYSSE, D., AND VAN OUDHEUSDEN, D. A guided local search heuristic for the capaci-

tated arc routing problem. *European Journal of Operational Research 147*, 3 (2003), 629–643.

[27] BHATIA, N., AND VANDANA. Survey of Nearest Neighbor Techniques. *International Journal of Computer Science and Information Security 8*, 2 (jul 2010).

[28] BRAMEIER, M. F., AND BANZHAF, W. *Linear Genetic Programming*. Springer, 2007.

[29] BRANDAO, J., AND EGLESE, R. A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers & Operations Research 35*, 4 (2008), 1112–1126.

[30] BRANKE, J., AND SCHMIDT, C. Faster convergence by means of fitness estimation. *Soft Computing 9*, 1 (jan 2005), 13–20.

[31] BURKE, E. K., GENDREAU, M., HYDE, M., KENDALL, G., OCHOA, G., ÖZCAN, E., AND QU, R. Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society 64*, 12 (dec 2013), 1695–1724.

[32] BURKE, E. K., GUSTAFSON, S., AND KENDALL, G. Diversity in Genetic Programming: An Analysis of Measures and Correlation with Fitness. *IEEE Transactions on Evolutionary Computation 8*, 1 (feb 2004), 47–62.

[33] BURKE, E. K., HYDE, M., KENDALL, G., AND WOODWARD, J. A Genetic Programming Hyper-Heuristic Approach for Evolving 2-D Strip Packing Heuristics. *IEEE Transactions on Evolutionary Computation 14*, 6 (dec 2010), 942–958.

[34] BURKE, E. K., HYDE, M. R., AND KENDALL, G. Evolving Bin Packing Heuristics with Genetic Programming. In *Parallel Problem Solving*

from Nature - PPSN IX (Berlin, Heidelberg, 2006), T. P. Runarsson, H.-G. Beyer, E. Burke, J. J. Merelo-Guervós, L. D. Whitley, and X. Yao, Eds., Springer Berlin Heidelberg, pp. 860–869.

[35] BURKE, E. K., HYDE, M. R., KENDALL, G., OCHOA, G., OZCAN, E., AND WOODWARD, J. R. *Exploring Hyper-heuristic Methodologies with Genetic Programming*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 177–201.

[36] BURKE, E. K., HYDE, M. R., KENDALL, G., AND WOODWARD, J. Automatic Heuristic Generation with Genetic Programming: Evolving a Jack-of-all-trades or a Master of One. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation* (New York, NY, USA, 2007), GECCO '07, ACM, pp. 1559–1565.

[37] BURKE, E. K., HYDE, M. R., KENDALL, G., AND WOODWARD, J. Automating the Packing Heuristic Design Process with Genetic Programming. *Evol. Comput. 20*, 1 (mar 2012), 63–89.

[38] CAMPBELL, J. F., AND LANGEVIN, A. *Roadway Snow and Ice Control*. Springer US, Boston, MA, 2000, pp. 389–418.

[39] CARUANA, R. Multitask Learning. *Machine Learning 28*, 1 (1997), 41–75.

[40] CHANDRA MOHAN, B., AND BASKARAN, R. A survey: Ant Colony Optimization based recent research and implementation on several engineering domain. *Expert Systems with Applications 39*, 4 (mar 2012), 4618–4627.

[41] CHEN, L., GENDREAU, M., HÀ, M. H., AND LANGEVIN, A. A robust optimization approach for the road network daily maintenance routing problem with uncertain service time. *Transportation Research Part E: Logistics and Transportation Review 85* (2016), 40–51.

[42] CHEN, L., HÀ, M. H., LANGEVIN, A., AND GENDREAU, M. Optimizing road network daily maintenance operations with stochastic service and travel times. *Transportation Research Part E: Logistics and Transportation Review 64* (2014), 88–102.

[43] CHEN, L. Y., LEE, P. M., AND HSIAO, T. C. A sensor tagging approach for reusing building blocks of knowledge in Learning Classifier Systems. *2015 IEEE Congress on Evolutionary Computation, CEC 2015 - Proceedings* (2015), 2953–2960.

[44] CHEN, Q. Feature Selection to Improve Generalization of Genetic Programming for High-Dimensional Symbolic Regression.

[45] CHEN, Y., HAO, J.-K., AND GLOVER, F. A hybrid metaheuristic approach for the capacitated arc routing problem. *European Journal of Operational Research 253*, 1 (aug 2016), 25–39.

[46] CHRISTIANSEN, C. H., LYSGAARD, J., AND WØHLK, S. A branch-and-price algorithm for the capacitated arc routing problem with stochastic demands. *Operations Research Letters 37*, 6 (2009), 392–398.

[47] CHU, T. H., AND NGUYEN, Q. U. Reducing code bloat in Genetic Programming based on subtree substituting technique. In *Proceedings of the Asia Pacific Symposium on Intelligent and Evolutionary Systems* (dec 2017), vol. 2017-Janua, Institute of Electrical and Electronics Engineers Inc., pp. 25–30.

[48] CHUGH, T., JIN, Y., MIETTINEN, K., HAKANEN, J., AND SINDHYA, K. A Surrogate-Assisted Reference Vector Guided Evolutionary Algorithm for Computationally Expensive Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation 22*, 1 (feb 2018), 129–142.

[49] CONOVER, W. J. *Practical Nonparametric Statistics*, vol. 350. John Wiley & Sons, 1998.

[50] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. No Title. In *Introduction to Algorithms*, 3rd ed. MIT Press and McGraw-Hill, 2009, ch. Hash Table, p. 1320.

[51] CORTÉS RIVERA, D., LANDA BECERRA, R., AND COELLO COELLO, C. A. Cultural algorithms, an alternative heuristic to solve the job shop scheduling problem. *Engineering Optimization 39*, 1 (jan 2007), 69–85.

[52] COWLING, P., COWLING, G., AND SOUBEIGA, E. A Hyperheuristic Approach to Scheduling a Sales Summit. In *Practice and Theory of Automated Timetabling III* (2001), E. Burke and W. Erben, Eds., Springer Berlin Heidelberg, pp. 176–190.

[53] CRAWSHAW, M. Multi-Task Learning with Deep Neural Networks: A Survey. *arXiv* (sep 2020).

[54] CUNNINGHAM, P., AND SMYTH, B. Case-based reasoning in scheduling: Reusing solution components. *International Journal of Production Research 35*, 11 (1997), 2947–2962.

[55] DA, B., GUPTA, A., AND ONG, Y.-S. Curbing Negative Influences Online for Seamless Transfer Evolutionary Optimization. *IEEE Transactions on Cybernetics* (2018), 1–14.

[56] DAI, H., KHALIL, E. B., ZHANG, Y., DILKINA, B., AND SONG, L. Learning Combinatorial Optimization Algorithms over Graphs. *Advances in Neural Information Processing Systems 2017-December* (apr 2017), 6349–6359.

[57] DATAR, M., INDYK, P., IMMORLICA, N., AND MIRROKNI, V. S. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Annual Symposium on Computational Geometry* (New York, New York, USA, 2004), Association for Computing Machinery, pp. 253–262.

[58] DAVIS, J. P., EISENHARDT, K. M., AND BINGHAM, C. B. Developing theory through simulation methods. *Academy of Management Review 32*, 2 (2007), 480–499.

[59] DE LORENZO, A., BARTOLI, A., CASTELLI, M., MEDVET, E., AND XUE, B. Genetic programming in the twenty-first century: a bibliometric and content-based analysis from both sides of the fence. *Genetic Programming and Evolvable Machines* (2019), 181–204.

[60] DENZINGER, J., FUCHS, M., AND FUCHS, M. High Performance ATP Systems by Combining Several AI Methods. In *Proceedings of the 15th International Joint Conference on Artifical Intelligence - Volume 1* (San Francisco, CA, USA, 1997), IJCAI'97, Morgan Kaufmann Publishers Inc., pp. 102–107.

[61] DING, J., YANG, C., JIN, Y., AND CHAI, T. Generalized Multitasking for Evolutionary Optimization of Expensive Problems. *IEEE Transactions on Evolutionary Computation 23*, 1 (feb 2019), 44–58.

[62] DINH, T. T. H., CHU, T. H., AND NGUYEN, Q. U. Transfer learning in genetic programming. In *Congress on Evolutionary Computation* (2015), pp. 1145–1151.

[63] DRAKE, J. H., HYDE, M., IBRAHIM, K., AND OZCAN, E. A genetic programming hyper-heuristic for the multidimensional knapsack problem. *Kybernetes 43*, 9/10 (2014), 1500–1511.

[64] DURASEVIĆ, M., AND JAKOBOVIĆ, D. A survey of dispatching rules for the dynamic unrelated machines environment. *Expert Systems with Applications 113* (dec 2018), 555–569.

[65] DÍAZ-ÁLVAREZ, J., CASTILLO, P. A., DE VEGA, F. F., CHÁVEZ, F., AND ALVARADO, J. Population size influence on the energy consumption of genetic programming. *Measurement and Control 55*, 1-2 (2022), 102–115.

[66] EGLESE, R. W., AND LI, L. Y. O. *A Tabu Search based Heuristic for Arc Routing with a Capacity Constraint and Time Deadline*. Springer US, Boston, MA, 1996, pp. 633–649.

[67] ELBES, M., ALZUBI, S., KANAN, T., AL-FUQAHA, A., AND HAWASHIN, B. A survey on particle swarm optimization with emphasis on engineering and network applications. *Evolutionary Intelligence 12*, 2 (jun 2019), 113–129.

[68] ELIASON, S. R. *Maximum likelihood estimation : logic and practice*. Sage, 1993.

[69] ESTER, M., KRIEGEL, H.-P., SANDER, J., AND XU, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (1996), AAAI Press, pp. 226–231.

[70] EYDI, A., AND JAVAZI, L. Model and solution approach for multi objective-multi commodity capacitated arc routing problem with fuzzy demand. *Journal of Industrial and Systems Engineering 5*, 4 (2012), 208–229.

[71] FENG, L., HUANG, Y., TSANG, I. W., GUPTA, A., TANG, K., TAN, K. C., AND ONG, Y.-S. Towards Faster Vehicle Routing by Transferring Knowledge From Customer Representation. *IEEE Transactions on Intelligent Transportation Systems* (sep 2020).

[72] FENG, L., HUANG, Y., ZHOU, L., ZHONG, J., GUPTA, A., TANG, K., AND TAN, K. C. Explicit Evolutionary Multitasking for Combinatorial Optimization: A Case Study on Capacitated Vehicle Routing Problem. *IEEE Transactions on Cybernetics 51*, 6 (2021), 3143–3156.

[73] FENG, L., ONG, Y. S., LIM, M. H., AND TSANG, I. W. Memetic Search with Interdomain Learning: A Realization between CVRP

and CARP. *IEEE Transactions on Evolutionary Computation 19*, 5 (2015), 644–658.

[74] FENG, L., ONG, Y. S., TAN, A. H., AND TSANG, I. W. Memes as building blocks: a case study on evolutionary optimization + transfer learning for routing problems. *Memetic Computing 7*, 3 (2015), 159–180.

[75] FENG, L., ZHOU, L., GUPTA, A., ZHONG, J., ZHU, Z., TAN, K.-C., AND QIN, K. Solving Generalized Vehicle Routing Problem With Occasional Drivers via Evolutionary Multitasking. *IEEE Transactions on Cybernetics* (dec 2019).

[76] FENG, L., ZHOU, L., ZHONG, J., GUPTA, A., ONG, Y. S., TAN, K. C., AND QIN, A. K. Evolutionary Multitasking via Explicit Autoencoding. *IEEE Transactions on Cybernetics 49*, 9 (sep 2019), 3457–3470.

[77] FISHER, R. D., AND THOMPSON, G. L. Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules. In *Factory Scheduling Conference* (1961).

[78] FLEURY, G., LACOMME, P., AND PRINS, C. Evolutionary Algorithms for Stochastic Arc Routing Problems. Springer, Berlin, Heidelberg, 2004, pp. 501–512.

[79] FLEURY, G., LACOMME, P., PRINS, C., AND RAMDANE CHERIF-KHETTAF, W. Robustness evaluation of solutions for the Capacitated Arc Routing Problem. In *Simulation and Planning in High Autonomy Systems* (Lisbon, Portugal, 2002), pp. 290–295.

[80] FOGEL, D. B. An Overview of Evolutionary Programming. 89–109.

[81] FORBES, C., EVANS, M., HASTINGS, N., AND PEACOCK, B. *Statiscitcal Distributions*, 4. edition ed. John Wiley & Sons, 2011.

[82] FU, W., XUE, B., ZHANG, M., AND GAO, X. Transductive Transfer Learning in Genetic Programming for Document Classification. In *Simulated Evolution and Learning* (Cham, 2017), Y. Shi, K. C. Tan, M. Zhang, K. Tang, X. Li, Q. Zhang, Y. Tan, M. Middendorf, and Y. Jin, Eds., Springer International Publishing, pp. 556–568.

[83] FUKUNAGA, A. S. Automated Discovery of Local Search Heuristics for Satisfiability Testing. *Evolutionary Computation 16*, 1 (mar 2008), 31–61.

[84] GAN, J., FENG, J., FANG, Q., AND NG, W. Locality-sensitive hashing scheme based on dynamic collision counting. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (New York, New York, USA, 2012), ACM Press, pp. 541–552.

[85] GEIGER, C. D., UZSOY, R., AND AYTUĞ, H. Rapid Modeling and Discovery of Priority Dispatching Rules: An Autonomous Learning Approach. *Journal of Scheduling 9*, 1 (feb 2006), 7–34.

[86] GLOVER, F. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research 13*, 5 (jan 1986), 533–549.

[87] GLOVER, F. Tabu Search—Part I. *ORSA Journal on Computing 1*, 3 (aug 1989), 190–206.

[88] GLOVER, F. Tabu Search—Part II. *ORSA Journal on Computing 2*, 1 (feb 1990), 4–32.

[89] GLOVER, F., KELLY, J. P., AND LAGUNA, M. Genetic algorithms and tabu search: Hybrids for optimization. *Computers and Operations Research 22*, 1 (jan 1995), 111–134.

[90] GOLDEN, B., DEARMON, J., AND BAKER, E. Computational experiments with algorithms for a class of routing problems. *Computers & Operations Research 10*, 1 (jan 1983), 47–59.

[91] GOLDEN, B. L., AND WONG, R. T. Capacitated arc routing problems. *Networks 11*, 3, 305–315.

[92] GONG, M., TANG, Z., LI, H., AND ZHANG, J. Evolutionary multitasking with dynamic resource allocating strategy. *IEEE Transactions on Evolutionary Computation 23*, 5 (oct 2019), 858–869.

[93] GREISTORFER, P. A tabu scatter search metaheuristic for the arc routing problem. *Computers & Industrial Engineering 44*, 2 (2003), 249–266.

[94] GUPTA, A., MAŃDZIUK, J., AND ONG, Y.-S. Evolutionary multitasking in bi-level optimization. *Complex & Intelligent Systems 1*, 1-4 (dec 2015), 83–95.

[95] GUPTA, A., ONG, Y. S., AND FENG, L. Multifactorial Evolution: Toward Evolutionary Multitasking. *IEEE Transactions on Evolutionary Computation 20*, 3 (2016), 343–357.

[96] GUPTA, A., ONG, Y. S., AND FENG, L. Insights on Transfer Optimization: Because Experience is the Best Teacher. *IEEE Transactions on Emerging Topics in Computational Intelligence 2*, 1 (2018), 51 – 64.

[97] GUPTA, R., AND RATINOV, L. Text categorization with knowledge transfer from heterogeneous data sources. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2* (Chicago, Illinois, 2008), AAAI Press, pp. 842–847.

[98] HAN, X., LIANG, Y., LI, Z., LI, G., WU, X., WANG, B., ZHAO, G., WU, C., AND HAN, X. An Efficient Genetic Algorithm for Optimization Problems with Time-Consuming Fitness Evaluation. *International Journal of Computational Methods 12*, 1 (2015), 12.

[99] HAR-PELED, S. A replacement for Voronoi diagrams of near linear size. In *Annual Symposium on Foundations of Computer Science - Proceedings* (2001), IEEE Computer Society, pp. 94–103.

[100] HASLAM, E., XUE, B., AND ZHANG, M. Further investigation on genetic programming with transfer learning for symbolic regression. In *Congress on Evolutionary Computation* (2016), IEEE, pp. 3598–3605.

[101] HAUPTMAN, A., ELYASAF, A., AND SIPPER, M. Evolving Hyper Heuristic-Based Solvers for Rush Hour and FreeCell. In *Third Annual Symposium on Combinatorial Search* (aug 2010).

[102] HAUSCHILD, M., SASTRY, K., PELIKAN, M., AND GOLDBERG, D. E. Using previous models to bias structural learning in the hierarchical BOA. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation* (2008), ACM, pp. 415–422.

[103] HAUSCHILD, M. W., AND PELIKAN, M. Intelligent bias of network structures in the hierarchical BOA. In *Proceedings of the 11th Annual Genetic and Evolutionary Computation Conference* (New York, USA, 2009), ACM Press, pp. 413–420.

[104] HERTZ, A., LAPORTE, G., AND MITTAZ, M. A tabu search heuristic for the capacitated arc routing problem. *Operations Research 48*, 1 (2000), 129–135.

[105] HILDEBRANDT, T., AND BRANKE, J. On using surrogates with genetic programming. *Evolutionary Computation 23*, 3 (2015), 343–367.

[106] HIRABAYASHI, R., SARUWATARI, Y., AND NISHIDA, N. Tour construction algorithm for the capacitated arc routing-problems. *Asia-Pacific Journal of Operational Research 9*, 2 (1992), 155–175.

[107] HONG, Y.-S., LEE, H., AND TAHK, M.-J. Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks. *Engineering Optimization 35*, 1 (feb 2003), 91–102.

[108] HU, T., AND BANZHAF, W. The role of population size in rate of evolution in genetic programming. In *Genetic Programming* (Berlin, Hei-

delberg, 2009), L. Vanneschi, S. Gustafson, A. Moraglio, I. De Falco, and M. Ebner, Eds., Springer Berlin Heidelberg, pp. 85–96.

[109] HUANG, Q., FENG, J., FANG, Q., NG, W., AND WANG, W. Query-aware locality-sensitive hashing scheme for lp norm. *VLDB Journal 26*, 5 (oct 2017), 683–708.

[110] HUSSAIN, M., BIRD, J. J., AND FARIA, D. R. A study on cnn transfer learning for image classification. In *Advances in Computational Intelligence Systems* (Cham, 2019), A. Lotfi, H. Bouchachia, A. Gegov, C. Langensiepen, and M. McGinnity, Eds., Springer International Publishing, pp. 191–202.

[111] HYDE, M., OZCAN, E., AND BURKE, E. K. Multilevel Search for Evolving the Acceptance Criteria of a Hyper-Heuristic. In *Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2009)* (Dublin, Ireland, 2009), pp. 798–801.

[112] INDYK, P., AND MOTWD, R. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing - STOC '98* (New York, New York, USA, 1998), ACM Press, pp. 604–613.

[113] IPPOLITI, E. *Heuristic Reasoning*. Studies in Applied Philosophy, Epistemology and Rational Ethics. Springer International Publishing, 2014.

[114] IQBAL, M., BROWNE, W. N., AND ZHANG, M. Extracting and using building blocks of knowledge in learning classifier systems. *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference - GECCO '12* (2012), 863.

[115] IQBAL, M., BROWNE, W. N., AND ZHANG, M. Reusing building blocks of extracted knowledge to solve complex, large-scale boolean

problems. *IEEE Transactions on Evolutionary Computation 18*, 4 (2014), 465–480.

[116] IQBAL, M., XUE, B., AL-SAHAF, H., AND ZHANG, M. Cross-Domain Reuse of Extracted Knowledge in Genetic Programming for Image Classification. *IEEE Transactions on Evolutionary Computation 21*, 4 (2017), 569–587.

[117] IQBAL, M., XUE, B., AND ZHANG, M. Reusing Extracted Knowledge in Genetic Programming to Solve Complex Texture Image Classification Problems. In *PAKDD 2016: Advances in Knowledge Discovery and Data Mining* (Cham, 2016), J. Bailey, L. Khan, T. Washio, G. Dobbie, J. Z. Huang, and R. Wang, Eds., Springer International Publishing, pp. 117–129.

[118] IQBAL, M., ZHANG, M., AND BROWNE, W. Automatically defined functions for learning classifier systems. *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation - GECCO '11* (2011), 375.

[119] IQBAL, M., ZHANG, M., AND XUE, B. Improving classification on images by extracting and transferring knowledge in genetic programming. *IEEE Congress on Evolutionary Computation* (2016), 3582–3589.

[120] JIN, Y. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation 1*, 2 (2011), 61–70.

[121] K BURKE, E., HYDE, M. R., KENDALL, G., OCHOA, G., AND OZCAN, E. *A classification of hyper-heuristic approaches*. 2010, pp. 449–468.

[122] KAHNEMAN, D., SLOVIC, P., AND TVERSKY, A. *Judgment Under Uncertainty: Heuristics and Biases*. Cambridge University Press, 1982.

[123] KELLER, R. E., AND POLI, R. Linear genetic programming of parsimonious metaheuristics. In *2007 IEEE Congress on Evolutionary Computation* (sep 2007), IEEE, pp. 4508–4515.

[124] KENDALL, M. G. A New Measure of Rank Correlation. *Biometrika 30*, 1-2 (jun 1938), 81–93.

[125] KENDY ARAKAKI, R., AND LUIZ USBERTI, F. An efficiency-based path-scanning heuristic for the capacitated arc routing problem. *Computers and Operations Research 103* (mar 2019), 288–295.

[126] KIM, S. H., AND BOUKOUVALA, F. Machine learning-based surrogate modeling for data-driven optimization: a comparison of subset selection for regression techniques. *Optimization Letters 2019 14:4 14*, 4 (may 2019), 989–1010.

[127] KINO, S., HARADA, T., AND THAWONMAS, R. Proposal of Surrogate Model for Genetic Programming Based on Program Structure Similarity. *Annual Conference of the Society of Instrument and Control Engineers of Japan* (sep 2020), 808–813.

[128] KLEINBERG, J. M. Two algorithms for nearest-neighbor search in high dimensions. In *Proceedings of the Annual ACM Symposium on Theory of Computing* (New York, USA, 1997), ACM, pp. 599–608.

[129] KOÇER, B., AND ARSLAN, A. Genetic transfer learning. *Expert Systems with Applications 37*, 10 (2010), 6997–7002.

[130] KOZA, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT, Cambridge, USA, 1992.

[131] KUMAR, R., JOSHI, A. H., BANKA, K. K., AND ROCKETT, P. I. Evolution of Hyperheuristics for the Biobjective 0/1 Knapsack Problem by Multiobjective Genetic Programming. In *Proceedings of the*

10th Annual Conference on Genetic and Evolutionary Computation (New York, NY, USA, 2008), GECCO '08, ACM, pp. 1227–1234.

[132] KUSHILEVITZ, E., OSTROVSKY, R., AND RABANI, Y. Efficient search for approximate nearest neighbor in high dimensional spaces. In *Conference Proceedings of the Annual ACM Symposium on Theory of Computing* (New York, New York, USA, 1998), ACM, pp. 614–623.

[133] LACOMME, P., PRINS, C., AND RAMDANE-CHÉRIF, W. A genetic algorithm for the capacitated arc routing problem and its extensions. In *Applications of Evolutionary Computing* (Berlin, Heidelberg, 2001), E. J. W. Boers, Ed., Springer, pp. 473–483.

[134] LAPORTE, G., MUSMANNO, R., AND VOCATURO, F. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science 44*, 1 (2010), 125–135.

[135] LEE, N. K., LI, X., AND WANG, D. A comprehensive survey on genetic algorithms for dna motif prediction. *Information Sciences 466* (2018), 25–43.

[136] LI, X., AND GAO, L. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *International Journal of Production Economics 174* (apr 2016), 93–110.

[137] LI, Z., LIN, X., ZHANG, Q., AND LIU, H. Evolution strategies for continuous optimization: A survey of the state-of-the-art. *Swarm and Evolutionary Computation 56* (2020), 100694.

[138] LIAW, R.-T., AND TING, C.-K. Evolutionary Manytasking Optimization Based on Symbiosis in Biocoenosis. *Proceedings of the AAAI Conference on Artificial Intelligence 33*, 01 (jul 2019), 4295–4303.

[139] LIM, D., JIN, Y., ONG, Y. S., AND SENDHOFF, B. Generalizing surrogate-assisted evolutionary computation. *IEEE Transactions on Evolutionary Computation 14*, 3 (jun 2010), 329–355.

[140] LIN, J., LIU, H.-L., TAN, K. C., AND GU, F. An Effective Knowledge Transfer Approach for Multiobjective Multitasking Optimization. *IEEE Transactions on Cybernetics* (mar 2020), 1–11.

[141] LIN, J., LIU, H.-L., XUE, B., ZHANG, M., AND GU, F. Multi-objective Multi-tasking Optimization Based on Incremental Learning. *IEEE Transactions on Evolutionary Computation* (2019).

[142] LIN, J., ZHAO, L., WANG, Q., WARD, R., AND WANG, Z. J. Dt-let: Deep transfer learning by exploring where to transfer. *Neurocomputing 390* (2020), 99–107.

[143] LING, X., XUE, G.-R., DAI, W., JIANG, Y., YANG, Q., AND YU, Y. Can chinese web pages be classified with english data source? In *Proceeding of the 17th international conference on World Wide Web - WWW '08* (New York, New York, USA, 2008), ACM Press, p. 969.

[144] LIU, J., TANG, K., AND YAO, X. Robust Optimization in Uncertain Capacitated Arc Routing Problems: Progresses and Perspectives. *IEEE Computational Intelligence Magazine 16*, 1 (2021), 63–82.

[145] LIU, Y., AND MEI, Y. Automated Heuristic Design Using Genetic Programming Hyper-Heuristic for Uncertain Capacitated Arc Routing Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2017), ACM, pp. 290–297.

[146] LIU, Y., MEI, Y., ZHANG, M., AND ZHANG, Z. A predictive-reactive approach with genetic programming and cooperative coevolution for the uncertain capacitated arc routing problem. *Evolutionary Computation 28*, 2 (2019), 289–316.

[147] LONGO, H., DE ARAGÃO, M. P., AND UCHOA, E. Solving capacitated arc routing problems using a transformation to the cvrp. *Computers & Operations Research 33*, 6 (2006), 1823–1837.

[148] LORIEN Y. PRATT. Discriminability-Based Transfer between Neural Networks. In *Advances in Neural Information Processing Systems 5, [NIPS Conference]* (1992), Morgan Kaufmann, pp. 204–211.

[149] LOSHCHILOV, I., SCHOENAUER, M., AND SEBAG, M. Self-adaptive surrogate-assisted covariance matrix adaptation evolution strategy. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference - GECCO '12* (New York, New York, USA, 2012), ACM Press, p. 321.

[150] LOUIS, S. J., AND MCDONNELL, J. Learning with case-injected genetic algorithms. *IEEE Transactions on Evolutionary Computation 8*, 4 (2004), 316–328.

[151] LU, J., BEHBOOD, V., HAO, P., ZUO, H., XUE, S., AND ZHANG, G. Transfer learning using computational intelligence: A survey. *Knowledge-Based Systems 80* (2015), 14–23.

[152] LUKE, S. Code Growth Is Not Caused by Introns. In *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference* (Las Vegas, Nevada, USA, 2000), D. Whitley, Ed., pp. 228–235.

[153] MACLACHLAN, J., MEI, Y., BRANKE, J., AND ZHANG, M. An improved Genetic Programming Hyper-heuristic for the Uncertain Capacitated Arc Routing Problem. In *AI 2018: Advances in Artificial Intelligence* (2018), pp. 1–12.

[154] MACLACHLAN, J., MEI, Y., BRANKE, J., AND ZHANG, M. Genetic Programming Hyper-Heuristics with Vehicle Collaboration for Uncertain Capacitated Arc Routing Problems. *Evolutionary Computation* (2019).

[155] MAHERI, A., JALILI, S., HOSSEINZADEH, Y., KHANI, R., AND MIRYAHYAVI, M. A comprehensive survey on cultural algorithms. *Swarm and Evolutionary Computation 62* (apr 2021), 100846.

[156] MANAZIR, A., AND RAZA, K. Recent Developments in Cartesian Genetic Programming and its Variants. *ACM Computing Surveys 51*, 6 (jan 2019).

[157] MARSHALL, R. J. Adapting a Hyper-heuristic to Respond to Scalability Issues in Combinatorial Optimisation, 2015.

[158] MCKAY, R. I., HOAI, N. X., WHIGHAM, P. A., SHAN, Y., AND O'NEILL, M. Grammar-based Genetic Programming: a survey. *Genetic Programming and Evolvable Machines 11*, 3-4 (sep 2010), 365–396.

[159] MEI, Y., LI, X., AND YAO, X. Cooperative Coevolution With Route Distance Grouping for Large-Scale Capacitated Arc Routing Problems. *IEEE Transactions on Evolutionary Computation 18*, 3 (jun 2014), 435–449.

[160] MEI, Y., NGUYEN, S., XUE, B., AND ZHANG, M. An Efficient Feature Selection Algorithm for Evolving Job Shop Scheduling Rules With Genetic Programming. *IEEE Transactions on Emerging Topics in Computational Intelligence 1*, 5 (2017), 339–353.

[161] MEI, Y., TANG, K., AND YAO, X. A Global Repair Operator for Capacitated Arc Routing Problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 39*, 3 (jun 2009), 723–734.

[162] MEI, Y., TANG, K., AND YAO, X. Capacitated arc routing problem in uncertain environments. In *Congress on Evolutionary Computation* (2010), IEEE, pp. 1–8.

[163] MEI, Y., AND ZHANG, M. Genetic Programming Hyper-heuristic for Multi-vehicle Uncertain Capacitated Arc Routing Problem. In

Proceedings of the Genetic and Evolutionary Computation Conference Companion (2018), ACM, pp. 141–142.

[164] MIHALKOVA, L., HUYNH, T., AND MOONEY, R. J. Mapping and revising Markov logic networks for transfer learning. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 1* (Vancouver, British Columbia, Canada, 2007), AAAI Press, pp. 608–614.

[165] MIHALKOVA, L., AND MOONEY, R. Transfer Learning with Markov Logic Networks. In *ICML workshop on structural knowledge transfer for machine learning* (2006).

[166] MIKROELEKTRONIK, . F., THIELE, L., AND BLICKLE, T. Genetic Programming and Redundancy. In *Genetic Algorithms Within the Framework of Evolutionary Computation*. Max-Planck-Institut fur Informatik, Saarbrucken, German, 1994, pp. 33–38.

[167] MILLER, J. F. *Cartesian Genetic Programming*, vol. 43. Springer, Berlin, Heidelberg, 2011.

[168] MOHAN, J., LANKA, K., AND RAO, A. N. A Review of Dynamic Job Shop Scheduling Techniques. *Procedia Manufacturing 30* (jan 2019), 34–39.

[169] MOHRI, M., ROSTAMIZADEH, A., AND TALWALKAR, A. *Foundations of Machine Learning*, 2nd ed. The MIT Press, 2018.

[170] MOR, A., AND SPERANZA, M. G. Vehicle routing problems over time: a survey. *4OR 18*, 2 (2020), 129–149.

[171] MUÑOZ, L., TRUJILLO, L., AND SILVA, S. Transfer Learning in Constructive Induction with Genetic Programming. *Genetic Programming and Evolvable Machines 21* (2020), 529–569.

[172] MURPHY, K. P. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[173] MURRE, J. Transfer of learning in backpropagation networks and in related neural network models. In *Connectionist Models of Memory and Language*. LondonUCL Press, 1995, pp. 73–93.

[174] MUYLDERMANS, L., AND PANG, G. *Chapter 10: Variants of the Capacitated Arc Routing Problem*. pp. 223–253.

[175] NAZARI, M., OROOJLOOY, A., TAKÁČ, M., AND SNYDER, L. V. Reinforcement Learning for Solving the Vehicle Routing Problem. In *NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montreal, Canada, 2018), pp. 9861–9871.

[176] NGUYEN, S., MEI, Y., AND ZHANG, M. Genetic programming for production scheduling: a survey with a unified framework. *Complex & Intelligent Systems 3*, 1 (2017), 41–66.

[177] NGUYEN, S., ZHANG, M., ALAHAKOON, D., AND TAN, K. C. Visualizing the evolution of computer programs for genetic programming [Research Frontier]. *IEEE Computational Intelligence Magazine 13*, 4 (nov 2018), 77–94.

[178] NGUYEN, S., ZHANG, M., ALAHAKOON, D., AND TAN, K. C. People-Centric Evolutionary System for Dynamic Production Scheduling. *IEEE Transactions on Cybernetics 51*, 3 (mar 2021), 1403–1416.

[179] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND CHEN TAN, K. Hybrid evolutionary computation methods for quay crane scheduling problems. *Computers & Operations Research 40*, 8 (2013), 2083–2093.

[180] NGUYEN, S., ZHANG, M., AND TAN, K. C. Surrogate-Assisted Genetic Programming With Simplified Models for Automated Design of Dispatching Rules. *IEEE Transactions on Cybernetics 47*, 9 (sep 2017), 2951–2965.

[181] NGUYEN, T.-T., AND YAO, X. An experimental study of hybridizing cultural algorithms and local search. *International journal of neural systems 18 1* (2008), 1–17.

[182] O'NEILL, D., AL-SAHAF, H., XUE, B., AND ZHANG, M. Common subtrees in related problems: A novel transfer learning approach for genetic programming. In *Proceedings of IEEE Congress on Evolutionary Computation* (2017), pp. 1287–1294.

[183] ONG, Y. S., AND GUPTA, A. Evolutionary Multitasking: A Computer Science View of Cognitive Multitasking. *Cognitive Computation 8*, 2 (apr 2016), 125–142.

[184] ONG, Y. S., ZHOU, Z., AND LIM, D. Curse and blessing of uncertainty in evolutionary algorithm using approximation. In *2006 IEEE Congress on Evolutionary Computation, CEC 2006* (2006), pp. 2928–2935.

[185] OZCAN, E., BILGIN, B., AND KORKMAZ, E. A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis 12* (2008), 3–23.

[186] PAN, S. J., AND YANG, Q. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering 22*, 10 (oct 2010), 1345–1359.

[187] PANDEY, H. M. Performance Evaluation of Selection Methods of Genetic Algorithm and Network Security Concerns. *Procedia Computer Science 78* (jan 2016), 13–18.

[188] PARK, J., MEI, Y., NGUYEN, S., CHEN, G., AND ZHANG, M. Evolutionary multitask optimisation for dynamic job shop scheduling using niched genetic programming. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2018), vol. 11320 LNAI, Springer Verlag, pp. 739–751.

[189] PARK, J., AND SANDBERG, I. W. Universal Approximation Using Radial-Basis-Function Networks. *Neural Computation 3*, 2 (jun 1991), 246–257.

[190] PEARL, J. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984.

[191] PEARN, W. L. Augment-insert algorithms for the capacitated arc routing problem. *Comput. Oper. Res. 18*, 2 (1991), 189–198.

[192] PELIKAN, M., AND HAUSCHILD, M. W. Distance-based bias in model-directed optimization of additively decomposable problems. In *Proceedings of the 14th International Conference on Genetic and Evolutionary Computation* (New York, USA, 2012), ACM Press, pp. 273–280.

[193] PELIKAN, M., HAUSCHILD, M. W., AND LANZI, P. L. Transfer learning, soft distance-based bias, and the hierarchical BOA. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2012), vol. 7491 LNCS, Springer, Berlin, Heidelberg, pp. 173–183.

[194] PENG, B., WANG, J., AND ZHANG, Z. A Deep Reinforcement Learning Algorithm Using Dynamic Attention Model for Vehicle Routing Problems. *Communications in Computer and Information Science 1205 CCIS* (feb 2020), 636–650.

[195] PISZCZ, A., AND SOULE, T. Genetic programming: Optimal population sizes for varying complexity problems. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation* (New York, NY, USA, 2006), GECCO '06, Association for Computing Machinery, p. 953–954.

[196] POLACEK, M., DOERNER, K. F., HARTL, R. F., AND MANIEZZO, V. A variable neighborhood search for the capacitated arc routing problem with intermediate facilities. *Journal of Heuristics 14*, 5 (oct 2008), 405–423.

[197] PRINS, C., AND BOUCHENOUA, S. *A Memetic Algorithm Solving the VRP, the CARP and General Routing Problems with Nodes, Edges and Arcs*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 65–85.

[198] RAJAKUMAR, B., AND GEORGE, A. Apoga: An adaptive population pool size based genetic algorithm. *AASRI Procedia 4* (2013), 288–296. 2013 AASRI Conference on Intelligent Systems and Control.

[199] REYNOLDS, R. G. An introduction to cultural algorithms. In *Evolutionary Programming — Proceedings of the Third Annual Conference* (USA, 1994), A. V. Sebald and L. J. Fogel, Eds., World Scientific Press, pp. 131–139.

[200] REZAABBASIFARD, M., GHAHREMANI, B., AND NADERI, H. A Survey on Nearest Neighbor Search Methods. *International Journal of Computer Applications 95*, 25 (jun 2014), 39–52.

[201] ROEVA, O., FIDANOVA, S., AND PAPRZYCKI, M. Influence of the population size on the genetic algorithm performance in case of cultivation process modelling. In *Federated Conference on Computer Science and Information Systems* (2013), IEEE, pp. 371–376.

[202] ROSTAMI, M., KOLOURI, S., EATON, E., AND KIM, K. Deep transfer learning for few-shot sar image classification. *Remote Sensing 11*, 11 (2019). DOI: 10.3390/rs11111374.

[203] RUNARSSON, T. P. Constrained Evolutionary Optimization by Approximate Ranking and Surrogate Models. Springer, Berlin, Heidelberg, 2004, pp. 401–410.

[204] RUSSELL, S., AND NORVIG, P. *Artificial Intelligence: A Modern Approach*, 3 ed. Prentice Hall, 2010.

[205] S, N., Y, M., B, X., AND M, Z. A Hybrid Genetic Programming Algorithm for Automated Design of Dispatching Rules. *Evolutionary computation 27*, 3 (2019), 467–596.

[206] SAŁUSTOWICZ, R., AND SCHMIDHUBER, J. Probabilistic Incremental Program Evolution: Stochastic search through program space. In *Lecture Notes in Computer Science*. Springer, 1997, pp. 213–220.

[207] SANTOS, L., COUTINHO-RODRIGUES, J., AND CURRENT, J. R. An improved heuristic for the capacitated arc routing problem. *Computers & Operations Research 36*, 9 (2009), 2632–2637.

[208] SANTOS, L., COUTINHO-RODRIGUES, J., AND CURRENT, J. R. An improved ant colony optimization based algorithm for the capacitated arc routing problem. *Transportation Research Part B: Methodological 44*, 2 (feb 2010), 246–266.

[209] SHAKHNAROVICH, G., DARRELL, T., AND INDYK, P. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)*. The MIT Press, 2006.

[210] SONG, A., CHEN, D., AND ZHANG, M. Bloat control in genetic programming by evaluating contribution of nodes. In *Proceedings of theconference on Genetic and evolutionary computation* (New York, New York, USA, 2009), ACM Press, p. 1893.

[211] SPINOSA, E., AND POZO, A. Controlling the population size in genetic programming. In *Advances in Artificial Intelligence* (Berlin, Heidelberg, 2002), G. Bittencourt and G. L. Ramalho, Eds., Springer Berlin Heidelberg, pp. 345–354.

[212] TAN, B., MA, H., MEI, Y., AND ZHANG, M. A Cooperative Coevolution Genetic Programming Hyper-Heuristic Approach for On-line Resource Allocation in Container-based Clouds. *IEEE Transactions on Cloud Computing* (2020).

[213] TAN, C., SUN, F., KONG, T., ZHANG, W., YANG, C., AND LIU, C. A survey on deep transfer learning. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (oct 2018), vol. 11141 LNCS, Springer Verlag, pp. 270–279.

[214] TAYLOR, M. E., WHITESON, S., AND STONE, P. Transfer learning for policy search methods. In *Proceedings of the Twenty-Third International Conference on Machine Learning* (2006), pp. 1–4.

[215] TIGHINEANU, P., SKUBCH, K., BAIREUTHER, P., REISS, A., BERKENKAMP, F., AND VINOGRADSKA, J. Transfer learning with gaussian processes for bayesian optimization, 2021.

[216] TIRKOLAEE, E. B., MAHDAVI, I., ESFAHANI, M. M. S., AND WEBER, G.-W. A hybrid augmented ant colony optimization for the multi-trip capacitated arc routing problem under fuzzy demands for urban solid waste management. *Waste Management & Research 38*, 2 (2020), 156–172. PMID: 31405349.

[217] TONG, H., HUANG, C., MINKU, L. L., AND YAO, X. Surrogate models in evolutionary single-objective optimization: A new taxonomy and experimental study. *Information Sciences 562* (jul 2021), 414–437.

[218] TORREY, L., AND SHAVLIK, J. Transfer learning. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. IGI Global, 2010, pp. 242–264.

[219] TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. Model reduction for real-time fluids. In *ACM SIGGRAPH 2006 Papers on - SIGGRAPH '06* (New York, New York, USA, 2006), vol. 25, ACM Press, p. 826.

[220] TRUJILLO, L., MUÑOZ, L., LÓPEZ, U., AND HERNÁNDEZ, D. E. Untapped Potential of Genetic Programming: Transfer Learning and Outlier Removal. In *Genetic Programming Theory and Practice XVI. Genetic and Evolutionary Computation*. Springer, 2019, pp. 193–207.

[221] USBERTI, F. L., FRANÇA, P. M., AND FRANÇA, A. L. M. The open capacitated arc routing problem. *Computers & Operations Research 38*, 11 (nov 2011), 1543–1555.

[222] VRAJITORU, D. *Large Population or Many Generations for Genetic Algorithms? Implications in Information Retrieval*. Physica-Verlag HD, Heidelberg, 2000, pp. 199–222.

[223] WANG, J., TANG, K., LOZANO, J. A., AND YAO, X. Estimation of the Distribution Algorithm With a Stochastic Local Search for Uncertain Capacitated Arc Routing Problems. *IEEE Transactions on Evolutionary Computation 20*, 1 (2016), 96–109.

[224] WANG, J., TANG, K., AND YAO, X. A memetic algorithm for uncertain Capacitated Arc Routing Problems. In *Proceedings of the Workshop on Memetic Computing* (2013), IEEE, pp. 72–79.

[225] WANG, S., MEI, Y., PARK, J., AND ZHANG, M. Evolving Ensembles of Routing Policies using Genetic Programming for Uncertain Capacitated Arc Routing Problem. In *IEEE Symposium Series on Computational Intelligence* (dec 2019), Institute of Electrical and Electronics Engineers Inc., pp. 1628–1635.

[226] WANG, S., MEI, Y., AND ZHANG, M. Novel ensemble genetic programming hyper-heuristics for uncertain capacitated arc routing

problem. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2019), ACM, pp. 1093–1101.

[227] WANG, S., MEI, Y., AND ZHANG, M. A Multi-Objective Genetic Programming Approach with Self-Adaptive α Dominance to Uncertain Capacitated Arc Routing Problem. In *Proceedings of IEEE Congress on Evolutionary Computation* (aug 2021), IEEE, pp. 636–643.

[228] WANG, S., MEI, Y., AND ZHANG, M. Towards Interpretable Routing Policy: A Two Stage Multi-Objective Genetic Programming Approach with Feature Selection for Uncertain Capacitated Arc Routing Problem. In *Proceedings of the Symposium Series on Computational Intelligence* (2021), IEEE, pp. 2399–2406.

[229] WANG, S., MEI, Y., AND ZHANG, M. Two-stage multi-objective genetic programming with archive for uncertain capacitated arc routing problem. In *Proceedings of the Genetic and Evolutionary Computation Conference* (jun 2021), Association for Computing Machinery, Inc, pp. 287–295.

[230] WANG, S., MEI, Y., ZHANG, M., AND YAO, X. Genetic programming with niching for uncertain capacitated arc routing problem. *IEEE Transactions on Evolutionary Computation 26*, 1 (2021), 73–87.

[231] WANG, X., JIN, Y., SCHMITT, S., OLHOFER, M., AND ALLMENDINGER, R. Transfer learning based surrogate assisted evolutionary bi-objective optimization for objectives with different evaluation times. *Knowledge-Based Systems 227* (2021), 107190.

[232] WEBER, R., SCHEK, H.-J., AND BLOTT, S. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB)* (1998).

[233] WEI, T., AND ZHONG, J. A Preliminary Study of Knowledge Transfer in Multi-Classification Using Gene Expression Programming. *Frontiers in Neuroscience 13* (2020), 1396.

[234] WEISE, T., DEVERT, A., AND TANG, K. A developmental solution to (dynamic) capacitated arc routing problems using genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference* (New York, USA, 2012), ACM Press, p. 831.

[235] WEISS, K., KHOSHGOFTAAR, T. M., AND WANG, D. A survey of transfer learning. *Journal of Big Data 3*, 1 (dec 2016), 9.

[236] WILLEMSE, E. J., AND JOUBERT, J. W. Constructive heuristics for the Mixed Capacity Arc Routing Problem under Time Restrictions with Intermediate Facilities. *Computers & Operations Research 68* (apr 2016), 30–62.

[237] WINEBERG, W., AND OPPACHER, F. The Benefits of Computing with Introns. In *Genetic Programming 1996*. The MIT Press, may 1996.

[238] WØHLK, S. *A Decade of Capacitated Arc Routing*. Springer, 2008, pp. 29–48.

[239] WUNDERLICH, J., COLLETTE, M., LEVY, L., AND BODIN, L. Scheduling Meter Readers for Southern California Gas Company. *Interfaces 22*, 3 (1992), 22–30.

[240] YSKA, D., MEI, Y., AND ZHANG, M. Genetic Programming Hyper-Heuristic with Cooperative Coevolution for Dynamic Flexible Job Shop Scheduling. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10781 LNCS* (2018), 306–321.

[241] ZHANG, F., MEI, Y., NGUYEN, S., TAN, K. C., AND ZHANG, M. Multitask Genetic Programming-Based Generative Hyperheuristics:

A Case Study in Dynamic Scheduling. *IEEE Transactions on Cybernetics* (2021).

[242] ZHANG, F., MEI, Y., NGUYEN, S., AND ZHANG, M. A preliminary approach to evolutionary multitasking for dynamic flexible job shop scheduling via genetic programming. In *GECCO 2020 Companion - Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion* (New York, NY, USA, jul 2020), Association for Computing Machinery, Inc, pp. 107–108.

[243] ZHANG, F., MEI, Y., NGUYEN, S., AND ZHANG, M. Evolving Scheduling Heuristics via Genetic Programming With Feature Selection in Dynamic Flexible Job-Shop Scheduling. *IEEE Transactions on Cybernetics* (oct 2020).

[244] ZHANG, F., MEI, Y., NGUYEN, S., AND ZHANG, M. Genetic Programming with Adaptive Search Based on the Frequency of Features for Dynamic Flexible Job Shop Scheduling. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 12102 LNCS* (apr 2020), 214–230.

[245] ZHANG, F., MEI, Y., NGUYEN, S., AND ZHANG, M. Guided Subtree Selection for Genetic Operators in Genetic Programming for Dynamic Flexible Job Shop Scheduling. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 12101 LNCS* (apr 2020), 262–278.

[246] ZHANG, F., MEI, Y., NGUYEN, S., AND ZHANG, M. Collaborative Multifidelity-Based Surrogate Models for Genetic Programming in Dynamic Flexible Job Shop Scheduling. *IEEE Transactions on Cybernetics* (2021).

[247] ZHANG, F., MEI, Y., NGUYEN, S., AND ZHANG, M. Correlation Coefficient based Recombinative Guidance for Genetic Programming Hyper-heuristics in Dynamic Flexible Job Shop Scheduling. *IEEE Transactions on Evolutionary Computation* (2021).

[248] ZHANG, F., MEI, Y., NGUYEN, S., ZHANG, M., AND TAN, K. C. Surrogate-Assisted Evolutionary Multitask Genetic Programming for Dynamic Flexible Job Shop Scheduling. *IEEE Transactions on Evolutionary Computation* (2021).

[249] ZHANG, F., MEI, Y., AND ZHANG, M. Genetic Programming with Multi-tree Representation for Dynamic Flexible Job Shop Scheduling. In *Australasian Joint Conference on Artificial Intelligence* (2018), Springer, pp. 472–484.

[250] ZHANG, F., MEI, Y., AND ZHANG, M. Surrogate-Assisted Genetic Programming for Dynamic Flexible Job Shop Scheduling. In *Proceedings of the Australasian Joint Conference on Artificial Intelligence (AI)* (2018), Springer, Cham, pp. 766–772.

[251] ZHANG, M., ZHANG, Y., AND SMART, W. Program Simplification in Genetic Programming for Object Classification. In *Knowledge-Based Intelligent Information and Engineering Systems* (Berlin, Heidelberg, 2005), R. Khosla, R. J. Howlett, and L. C. Jain, Eds., Springer Berlin Heidelberg, pp. 988–996.

[252] ZHANG, Y., AND YANG, Q. A Survey on Multi-Task Learning. *IEEE Transactions on Knowledge and Data Engineering* (2021). Doi: 10.1109/TKDE.2021.3070203.

[253] ZHENG, X., QIN, A. K., GONG, M., AND ZHOU, D. Self-Regulated Evolutionary Multitask Optimization. *IEEE Transactions on Evolutionary Computation 24*, 1 (feb 2020), 16–28.

[254] ZHONG, J., FENG, L., CAI, W., AND ONG, Y. Multifactorial Genetic Programming for Symbolic Regression Problems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (jul 2018).

[255] ZHOU, J., ZENG, S., AND ZHANG, B. Two-stage knowledge transfer framework for image classification. *Pattern Recognition 107* (2020), 107529.

[256] ZHOU, L., FENG, L., GUPTA, A., ONG, Y. S., LIU, K., CHEN, C., SHA, E., YANG, B., AND YAN, B. W. Solving dynamic vehicle routing problem via evolutionary search with learning capability. In *2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings* (jul 2017), Institute of Electrical and Electronics Engineers Inc., pp. 890–896.

[257] ZHOU, L., FENG, L., TAN, K. C., ZHONG, J., ZHU, Z., LIU, K., AND CHEN, C. Toward Adaptive Knowledge Transfer in Multifactorial Evolutionary Computation. *IEEE Transactions on Cybernetics* (mar 2020), 1–14.