Diversity-driven Knowledge Transfer for GPHH to solve Uncertain Capacited Arc Routing Problem

Mazhar Ansari Ardeh*, Yi Mei† and Mengjie Zhang‡
School of Engineering and Computer Science, Victoria University of Wellington
PO Box 600, Wellington, New Zealand
Email: *mazhar.ansariardeh@ecs.vuw.ac.nz, †yi.mei@ecs.vuw.ac.nz, ‡mengjie.zhang@ecs.vuw.ac.nz

***Abstract*—Uncertain Capacitated Arc Routing Problem (UCARP) is a dynamic combinatorial optimisation problem which can model many real-world logistic systems. Currently, the best available method for solving UCARP is the approach of using Genetic Programing as a hyper-heuristic to evolve routing policies for vehicles automatically. An open challenge in this area is that any change to the features of a solved UCARP instance will make trained policies ineffective for the new problem. As a result, whenever such changes happen, it is required to train new routing policies from scratch. The process of training routing policies is generally expensive. It is desirable to utilise transfer learning methods to reduce the retraining cost. However, earlier studies have identified that performing transfer learning for handling problem changes of UCARP is a challenging task. Lack of diversity and possible convergence to poor local optima are some issues that contribute to this challenge. To address these issues, in this work, we propose a new transfer learning approach with hyper-mutation in GPHH for UCARP to tackle the issue of insufficient diversity of the transferred knowledge. Our experiments demonstrate that the newly proposed approach can increase the effectiveness of knowledge transfer and can allow better handling of UCARP scenario changes through transfer learning.**

I. Introduction

Capacitated Arc Routing Problem (CARP) is a combinatorial optimisation problem with many real-world applications. CARP was first proposed by Golden et al. [1]. CARP designs the routes for a fleet of vehicles with limited capacity to serve a collection of tasks in an environment. In CARP, the environment is modelled with a connected undirected graph $G(V, E)$. Each node of G represents a location in the environment and edges represent the paths between locations. In CARP, some edges are required to be served (so-called tasks) by the vehicles. Each task has a demand value which represents the amount of work that needs to be done for that task. The act of serving tasks also incurs a cost. The goal in CARP is to find a set of routes to serve the tasks with minimal total cost while respecting some predefined constraints [1]. Street watering [2], snow removal [2] and waste collection [3] are some of the applications of CARP. Despite its applicability, the original definition of CARP is very restricted and is not flexible enough for various real-world applications. As a result, different variants of CARP have been proposed [4], [5]. Despite this, most of these variants concentrated on static CARP in which the problem parameters are fixed and are assumed to be known beforehand. However, uncertainty is an inherent feature of many real-world problems and the assumption that problem parameters are fixed and not subject to change does not always hold. For example, in CARP, it is assumed that the cost of serving a task is fixed. In contrast, it is possible, and even very likely for some problems, that the cost could change even while the task is being served (e.g. due to traffic jams).

To address the limitation imposed by the static nature of CARP, Mei et al. [6] extended it with four stochastic parameters. In this uncertain version of CARP, named Uncertain Capacitated Arc Routing Problem (UCAPRP), the presence of tasks and their demand, the presence of paths and the traversal cost of paths are stochastic [6]. Shortly after, Liu et al. [7] proposed an evolutionary method for solving UCARP. Section II-A presents a more detailed description of UCARP and its features.

There exist several categories of approaches for solving UCARP, amongst which the routing policy technique is one of the most flexible ones [7]–[9]. A routing policy is a real-valued mathematical function that takes the state of the environment, vehicles and available tasks as input and assigns a priority value to each task so that an idle vehicle can select the task with the highest priority to serve next. This feature of routing policies make it a very versatile tool for handling the uncertain and dynamic nature of UCARP because it allows the vehicles to make decisions in real-time and based on the most up-to-date information that they have about their environment. As of now, the routing policy approach is considered to be the state-of-the-art for solving UCARP.

Routing policies can be designed for UCARP manually. However, the manual design of routing policies requires extensive domain expertise and can be time-consuming and expensive. However, routing policies can be considered as computer programs and hence, it is possible to use Genetic Programming (GP) for evolving effective routing policies. As a matter of fact, the approach of using routing policies for handling uncertainty in environments and the utilisation of GP for training them is not specific to UCARP. For instance, GP has been used for evolving routing/dispatching rules for tackling the uncertainty in dynamic job-shop scheduling [10], [11] and online resource allocations in cloud containers [12].

After training, routing policies can be used for guiding the decisions of vehicles for as long as the problem features do not change. On the other hand, it has been shown in [6] that if any change to the feature of a problem occurs, such as the

number of vehicles decrease due to vehicle break-down, the routing policy that has been trained for the previous problem will not perform optimally for the new problem. As a result, it is needed to train a new routing policy for the new problem. Although using GP for training routing policy relieves the need for extensive domain expertise, it requires running multiple simulations and consequently, is a time-consuming process. Therefore, it is desirable to be able to reduce the time of the retraining process.

It is reasonable to believe that if the problem is not changed drastically, there is enough similarity between the two problems to make it possible to extract some knowledge from the solved problem to help the effectiveness and efficiency of the training new routing policies for the new problem. This functionality belongs to the realm of transfer learning.

The goal of transfer learning is to salvage some form of knowledge from a problem that is already solved and use the knowledge for alleviating the solving process of a new but related problem. However, Ardeh et al. [9] investigated the potentials of some of the existing transfer learning methods for handling scenario changes of UCARP and concluded that those investigated methods, as effective as they may be for the original problems they were developed for, do not demonstrate satisfactory performance for handling scenario changes of UCARP. In their work, Ardeh et al. identified that lack of diversity and possible convergence to local optima in the source domain are two primary reasons for the unsatisfactory performance of the existing transfer learning methods. Consequently, in this work, we have the following research goals:

- Propose new transfer methods with hyper-mutation for improving the diversity of the GP population after the knowledge transfer.
- Investigate the effects that the increased diversity can have on the quality of knowledge transfer for handling scenario changes of UCARP.

The organisation of this paper is as follows. A detailed review of UCARP, Genetic Programming and transfer learning is given in Section II. The proposed hyper-mutation approach to handling scenario changes of UCARP is described in Section III. Section IV presents the experimental settings and results. The paper is concluded in Section V with some directions for future works.

II. Background

In this section, a comprehensive definition of UCARP will be provided in Subsection II-A. An overview of different approaches to solving UCARP, especially the approach of using GP for evolving routing policies will be given in Subsection II-B. Transfer learning in Evolutionary Computation and, particularly, transfer learning for handling scenario changes of UCARP will be reviewed in Subsection II-C.

A. UCARP

UCARP is an uncertain combinatorial optimisation problem that simulates a fleet of vehicles that need to serve a collection of tasks. In this problem, the vehicles have a fixed capacity Q. In UCARP, the tasks are assumed to be spread over the roads of an environment which is modelled with a graph $G(V, E)$. In this graph, V is the set of nodes that represents the locations of the environment and E is the set of edges that represent the routes of the environment. The graph contains a special node $v_0 \in V$ at which all vehicles are stationed in the beginning.

Two cost values are associated with each edge $e \in E$: 1) a positive and deterministic serving cost $sc(e)$ and, 2) a positive stochastic deadheading cost $dc(e)$. The deadheading cost $dc(e)$ is the cost of traversing the edge e without serving it. The amount of work, $d(e)$, that is needed to be done over an edge e is the called the demand of that task. An edge e for which $d(e) > 0$ is referred to as a task. The goal of UCARP is to find a set of routes for all vehicles to serve all tasks with the minimum total cost. To solve UCARP, a set of constraints must not be violated:

- No task is allowed to be served more than once. However, vehicles are allowed to traverse a task without serving it more than once.
- Vehicles cannot serve more demand than their capacity. If a vehicle becomes full, it needs to return to the depot to replenish its capacity and then resume the serving process.
- All vehicle routes must start and end with the depot node. Nevertheless, the vehicles can return to the depot while serving their tasks to replenish their capacity.

When the stochastic variables of a UCARP instance are sampled from a probability distribution, it is referred to as a sampled UCARP instance. In sampled instances, the actual demand values are revealed after the vehicle finishes serving it. The actual deadheading cost is revealed after the vehicle finishes traversing it.

There are two categories of methods for solving UCARP. *Proactive* methods find a robust solution for the static version of UCARP and then optimise it for the uncertainties of the environment. The works by Mei et al. [6] and Wang et al. [13] and Prins et al. [14] belong to the category of proactive methods. *Reactive* methods typically employ Genetic Programming as a hyper-heuristic (GPHH) to train routing policies for vehicles.

Proactive methods disregard the uncertainty of UCARP when finding robust solutions and in return, try to re-optimise the found solution whenever a failure occurs. As a result, these methods are not as flexible as desired. Motivated by this shortcoming, Liu et al. [7] proposed to equip vehicles with a routing policy so that the vehicles can make decisions in their environment based on the most recent information. Clearly, this approach allows real-time decision making and provides more flexibility. The authors utilised Genetic Programming as a Hyper Heuristic to train routing policies and evaluated their method on benchmark problems described in [6].

B. GPHH for UCARP

Popularized by Koza, Genetic Programming extends the famous Genetic Algorithm (GA) by letting the population members be computer programs. This idea of using the genetic

operators for training computer programs proved to be a very effective approach and has seen extensive applications in a vast variety of problems [15], [16]. When standard GP [17] is employed to search the space of heuristic functions, it is referred to as Genetic Programming Hyper-Heuristic.

Routing policies of UCARP are heuristics in nature and hence, GPHH can be utilised for evolving them. Opting a tree representation for routing policies, this algorithm first initialises a random population of routing policies and then, evolves them by applying the genetic operators of crossover, mutation and reproduction until some stopping criteria are met.

The GPHH approach to solving UCARP requires a meta-algorithm that, given a UCARP instance and a routing policy, can generate the final solution of vehicle routes. For this purpose, an intuitive meta-algorithm is as follows. In the beginning, all vehicles are idle and stationed at the depot. Whenever any vehicle becomes idle, it considers the set of all available tasks. A few filtering methods [18] can be applied to the set to remove inaccessible tasks and improve efficiency. Then, the routing policy is applied to the set of available and accessible tasks with the most recent information about the state of the environment to obtain the priority of each task. Finally, the task with the best priority will be selected to be served. After finishing the task, the vehicle becomes idles and reviews the set of available tasks. If there is no task to serve next, then the problem is solved and the vehicle can return to the depot.

During the course of serving a task, two types of failure may occur. Due to the uncertain nature of the problem, it is possible that the vehicle arrives at a task whose demand is greater than the capacity of the vehicle. In this case, it is said that a *route failure* has happened. In case of a route failure, the vehicle needs to return to the depot to replenish its capacity and then resume serving the task. Also, arriving at a task, the vehicle may find that the deadheading cost of the task is infinite (i.e. the route is inaccessible). This situation is referred to as an *edge failure*. In this case, the vehicle needs to find a detour around the failed edge. Based on the described meta-algorithm, the fitness value of a routing policy is the average total cost of the solutions it generates for a set of training UCARP instances.

The work by Liu et al. [7] was developed for just one vehicle. Mei et al. [18] extended GPHH to train routing policies for multiple vehicles. They also introduced the idea of improving the efficiency of vehicles by filtering out inaccessible tasks. MacLachlan [19] introduced a new state variable (GP terminal) allowed the vehicles to have a better understanding of their environments and make better-informed decisions. Wang et al. [8] proposed a method for evolving an ensemble of the routing policies that can train powerful yet more interpretable policies. Later on, they [20] improved the interpretability by optimising the performance of the policies while minimising their size. MacLachlan et al. [21] considered two collaboration scenarios to allow vehicles to cooperate while serving tasks and showed that the collaboration can increase the effectiveness of the vehicles.

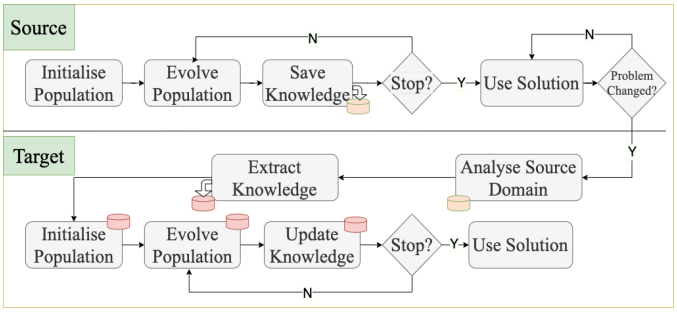


Fig. 1: A general GP-based transfer learning framework

C. Transfer Learning for GP

Many traditional evolutionary algorithms solve problems in isolation and separately [9]. Consequently, they solve all problems from scratch, even if similar problems have been solved before. In contrast, human beings can solve a problem faster and easier when they have had solved similar problems before. This is due to the inherent ability of humans which allows them to extract and retain some form of knowledge after solving a problem and take advantage of it later when similar problems are encountered. Knowing that many evolutionary methods can be computationally-expensive, the transfer learning feature of human minds motivated the invention of similar capabilities for evolutionary methods. Figure 1 presents a high-level flowchart of transfer learning for GP.

O'Neill et al. define transfer learning as "the ability of a system to recognise and apply knowledge and skills learned in previous tasks to novel tasks" [22]. One way that transfer learning methods can help evolutionary algorithms is by enabling them to have an initial population that is better than random initialisation which can help the convergence speed of the methods and even arrive at results that are better than the results that vanilla evolutionary methods could arrive at with similar effort [23]. Compared to the other areas of the machine learning literature, transfer learning is a rather new concept. However, due to its important potentials, it has gathered a lot of attention from scholars. As a result, the number of works in this field is abundant [24], [25]. Therefore, this section will focus on the transfer learning methods that are originally devised for or are applicable to GP and interested readers are referred to [24], [25] for more comprehensive reviews.

A majority of existing transfer learning approaches for Evolutionary Computation (EC) methods are based on the idea of transferring helpful genetic materials from a solved source domain to an unsolved but related target domain. For example, Taylor et al. [26] performed transfer learning for GA by simply transferring the final population of the source domain to the target domain and allowing GA to continue the search. Similarly, Dinh et al. [27] believed that some randomness in the initial population could be beneficial for GP. In their *FullTree-k* method, they selected the best $k\%$ of the final source population to be transferred and initialised the rest of the target population randomly.

Instead of transferring the whole GP individuals, Dinh et

al. [27] also proposed the *SubTree-k* method to consider the final source population, select one of the root subtrees of GP trees and use them as new GP trees to initialise $k\%$ of the target domain population. Iqbal et al. [28] also considered the root subtrees of the final source population as important genetic material but they modified the initialisation operator so that children of the root node are either selected from the transferred subtrees or created randomly.

Ardeh et al. [9] were the first that considered using transfer learning for handling scenario changes of UCARP. In their work, they utilised several existing knowledge transfer methods for EC algorithms and evaluated their potentials. Additionally, they speculated that the subtrees which appear more frequently in the source domain must contain important genetic materials that GP created them frequently in the source domain. Therefore, their *FreqSub-k* method extracted the most frequent subtrees in the source domain and used them as new individuals to initialise $k\%$ of GP population in the target domain. An important aspect of their work was that they identified possible convergence to local optima and lack of diversity in the source domain to be the main challenges to could significantly reduce the effectiveness of the knowledge transfer methods for UCARP.

In a later work, the same authors [29] focused again on handling scenario changes of UCARP. In this work, they considered the contribution that each subtree makes to the fitness of its individual to measure the importance of subtrees. Consequently, they selected the subtrees that contributed the most to their individuals as transferrable knowledge and used them as new individuals to initialise a portion of target GP population. Similar to their previous work, they noted again that lack of diversity in the source domain is the main obstacle to successful knowledge transfer.

Instead of transferring the genetic materials, Ardeh et al. [30] learned the importance of features from a source domain. In a target domain, they modified the initialisation and mutation operator so that terminals of new trees and subtrees were selected with respect to the learned feature weights. Later on, the same authors [31] extended their work and took GP functions into consideration.

Ardeh et al. [9], [29], [30], pointed out an important challenge that can have a major negative role when performing knowledge transfer. During the GP evolution, the genetic operators search the space of potential solutions and try to evolve the population towards the regions of the search space that have better potentials for having the final solutions. As a result, in later generations, GP individuals tend to be focused on these high-potential regions. Consequently, although the GP population will have better average fitness, one side effect of this process is that the phenotypic (and possibly genotypic) diversity of the population will decrease. Additionally, when GP is converged to local or global optima of the source domain, transferring the population to a target domain will place GP in potential local optima of the target domain. This can occur even if GP is converged to the global optima of the source domain because there is no guarantee that the global optima of the source domain will be global optima of the target domain as well. Therefore, when performing transfer learning, GP is faced with a set of individuals that are either fit but lack diversity or, are diverse but lack quality. Transferring low-quality individuals cannot be beneficial and transferring low-diversity ones will run the risk of trapping GP in local optima. Although increasing the quality of diverse individuals is difficult (and possibly, requires further training), helping the diversity of good individuals is possible with far less effort and GP is already equipped with an appropriate tool.

Algorithm 1: Pseudocode for DDGPHH

Input : p_0: initial mutation rate, δ_m: minimum mutation rate, p_r: reproduction rate, $\mathcal{S}$: a source domain.
Output: The best solution

$\Im \leftarrow$ `loadSource` ($\mathcal{S}$)
`// Initialise GP population with a`
`// FullTree transfer (see Section II-C)`
pop $\leftarrow$ `FullTree` ($\Im$)
for $t = 0; t < maxGen$ **do**
  `evaluate` (*pop*)
  pop $\leftarrow$ `crossover` (*pop*, $p_c(t)$)
  pop $\leftarrow$ `mutate` (*pop*, $p_m(t)$)
  pop $\leftarrow$ `reproduce` (*pop*, $p_r(t)$)
  `// Update the rates`
  $p_m(t+1) \leftarrow$ `updateMutationRate` ($p_m(t)$)
  if $p_m(t+1) < \delta_m$ **then**
    $p_m(t+1) \leftarrow \delta_m$
  end
  $t \leftarrow t + 1$
end
return $\mathcal{P}$

III. Diversity-Oriented Knowledge Transfer for GPHH

In a majority of transfer learning methods for EC algorithms, the knowledge transfer process comprises a few high-level design questions [31]: 1) what is the knowledge source? 2) how to represent knowledge? 3) how to extract knowledge? 4) how to use the knowledge in the target domain? Different approaches have been proposed for using the extracted knowledge in a target domain. However, a majority of the methods select the individuals in the GP population (usually the final population) as the knowledge source.

Any evolutionary algorithm is based on the idea of exploring the search space of a problem to find the regions that have good potentials for containing the good solutions and exploiting these regions to discover the global optima. In the context of GP, the crossover operator is the main tool for the exploitation phase. The mutation operator, on the other hand, plays a major role in exploration. In this regard, the role that the initialisation operator plays is also important. When the initialisation is performed randomly, it can contribute to the exploration phase of GP. However, when the GP population is initialised with the transferred individuals of a related source domain, the initialisation operator contributes to the exploitation phase of GP. This is because the operator transfers some information about the areas of the search space that

have previously been explored in the source domain and may have good potentials in the target domain too. As a result, performing transfer learning will bias the search towards exploitation. What makes this situation even more harder it that because of the GP convergence in later generations, the population may contain phenotypically or genotypically similar or even idential individuals which will also contribute to the lack of diversity and reduce the quality of transfer. This feature, combined with the possibility of transferring local optima can hurt the effectiveness of the search.

In order to enhance the exploration phase of GP and reduce the chance of getting stuck in local optima, in this work we propose allowing GP more exploration after a *FullTree* knowledge transfer. In this regard, the mutation operator can be helpful and by increasing the mutation rate (and decreasing the crossover rate accordingly) after knowledge transfer, GP will have a better capability of exploration. The mutation rate can be set to a fixed high rate but having a high mutation rate throughout the search process can be detrimental to GP performance as it does not allow GP enough opportunity to exploit the regions that have been found to be promising. Consequently, in this work, we propose a simple yet effective transfer learning approach based on an adaptive mutation rate with different adaptation strategies. In our approach, the mutation rate is set to an initial value $p_m(0)$ after the knowledge transfer and at each generation t, its value is updated based on the following strategies:

- **Exponential**: At each generation and with an adaptation rate of $0 < \gamma < 1$, the mutation rate is updated with $p_m(t+1) = p_m(t)\gamma^t$. The motivation for this strategy is to let GP start with a high mutation rate to allow it more exploration but drop the rate quickly to prevent the excessive exploration to hurt the search process.
- **Power**: The mutation rate is updated with $p_m(t+1) = p_m(0)/(1+t)^{0.5}$. This strategy works similar to the *exponential* strategy but reduces the mutation rate more slowly than it.
- **Cosine**: The mutation rate is updated with $p_m(t+1) = p_m(0)\cos(\pi t/t_{max})$ in which t_{max} is the total number of generations. This strategy also starts with a large rate but reduces it to allow more exploitation. However, the mutation rate is increased again after a while. The motivation here is to allow GP more exploration at first, then let it exploit the found results and then, do more exploration again to escape from any potential local optima that it may fall into.

Additionally, in order to make sure that mutation is never disabled, we consider a minimum threshold δ_m for the mutation rate. After the mutation rate is updated, we adjust the crossover rate too as $p_c(t) = 1 - p_m(t) - p_r$ in which p_r is the fixed reproduction rate. Figure 2 presents the mutation probabilities that GP will have at each generation based on different update strategies ($p_0 = 0.95, \gamma = 0.95$). Accordingly, the proposed Diversity-Driven GPHH (*DDGPHH*) for the target domain is presented in Table 1.

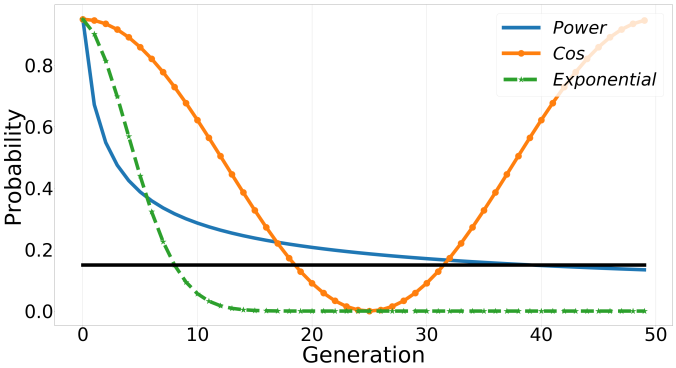


Fig. 2: Mutation probabilities at each generation for different strategies

TABLE I
NUMBER OF VEHICLES ON SOURCE AND TARGET PROBLEMS

Scenario	GDB1-54	GDB1-56	GDB3-54	GDB3-56	GDB6-54
Source	5	5	5	5	5
Target	4	6	4	6	4
Scenario	GDB6-56	GDB7-54	GDB7-56	GDB23-109	GDB23-1011
Source	5	5	5	10	10
Target	6	4	6	9	11

IV. EXPERIMENTAL STUDIES

To evaluate the effectiveness of our newly proposed approach, we conducted a number of experiments. In our experiments, we defined the scenario changes of UCARP to be based on changes in the number of vehicles and kept all other aspects of the problem unchanged. We used the *GDB* dataset for policy training [7]. Table I presents the considered scenarios. We adopted the GP settings that are described in [7], [9] to remain consistent with the literature of UCARP. These settings, as well as, GP terminals are presented in Table II. The GP function set is $\{+, -, *, /, \min, \max\}$ in which the division operator $/$ is protected so that it will return 1 if its denominator is 0. All experiments were performed for 30 independent runs and the results are compared with the unpaired Wilcoxson rank sum test with a 0.05 confidence level.

We experimented with different values of the initial mutation rate p_0 and adaptation rate γ and we obtained the best results with the $p_0 = 0.95$ and $\gamma = 0.95$ values. In addition to the proposed adaptive strategies of Section III, we also considered the nonadaptive approach of setting the mutation rate to a large fixed value that does not change over GP generations to check if the adaptiveness of the mutation rate has any effect or not. For this purpose, after performing a *FullTree* transfer we considered three fixed mutation rates of $0.15, 0.50$ and 0.95 which are refered to as *Fixed-0.15*, *Fixed-0.5* and *Fixed-0.95*. The mutation rate of 0.15 is the default mutation setting for GP, Table II, and hence *Fixed-0.15* pertains to a FullTree transfer plus standard GPHH. *Fixed-0.5* has a high mutation rate but it also allows some exploitation with crossover. *Fixed-0.95* is the extreme case that focuses entirely on exploration and does not allow any exploitation.

TABLE II
GP SETTINGS.

GP 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
GP Setting	**Value**
Population	1024
Crossover rate	0.8
Mutation rate	0.15
Reproduction rate	0.05
Number of generations	50
Elitism	10
Max depth	8

TABLE III
MEANS OF 30 RUNS OF BEST OF GP GENERATIONS FOR DATASET , FROM TO VEHICLES

Scenario	GDB1-54	GDB1-56	GDB3-54	GDB3-56	GDB6-54
No Transfer	357.45±6.2	361.08±3.0	317.76±17.6	327.57±5.7	356.06±19.1
Fixed-0.15	360.6±8.1	359.13±4.8	317.84±9.1	326.65±5.9	351.16±8.4
Fixed-0.5	356.32±5.6	**359.25±8.4**	316.89±7.4	326.38±6.0	356.97±33.8
Fixed-0.95	358.32±7.0	**356.62±4.5**	315.42±11.8	327.24±13.6	352.39±22.5
DDGPHH-Cos	358.36±6.0	**357.2±5.1**	317.59±10.5	**324.77±6.3**	350.76±19.2
DDGPHH-Exp	359.79±7.6	**357.54±4.9**	314.85±5.6	**326.59±9.3**	**348.67±8.0**
DDGPHH-Pow	356.31±5.8	**356.86±4.4**	317.91±10.2	**326.19±5.3**	354.94±22.0
DDGPHH-Rand	357.91±6.5	**357.79±4.9**	**311.26±12.5**	328.44±6.5	350.83±8.4
Scenario	GDB6-56	GDB7-54	GDB7-56	GDB23-109	GDB23-1011
No Transfer	356.21±7.5	381.88±13.7	357.52±2.2	251.83±3.4	249.91±2.6
Fixed-0.15	358.21±16.2	389.59±14.0	357.9±2.1	251.04±2.4	**248.48±1.7**
Fixed-0.5	355.68±7.4	383.3±12.8	357.94±2.4	250.83±2.9	249.54±3.2
Fixed-0.95	357.17±14.3	381.71±9.8	**356.05±0.8**	250.72±3.3	249.02±1.9
DDGPHH-Cos	356.01±4.7	382.6±12.4	358.52±7.4	251.01±3.1	248.81±2.0
DDGPHH-Exp	357.6±13.2	382.44±11.7	**356.82±1.8**	**250.13±2.7**	249.45±2.3
DDGPHH-Pow	356.08±10.0	385.16±10.3	358.18±4.3	250.14±2.5	248.83±1.4
DDGPHH-Rand	356.47±6.0	382.55±12.5	356.98±2.2	251.49±2.9	250.63±3.5

A. Experiment Results

The mean and standard deviation of the final performance obtained by 30 independent runs of the compared the algorithms are presented in Table III. The results that are significantly better than GPHH without any knowledge transfer are presented in boldface.

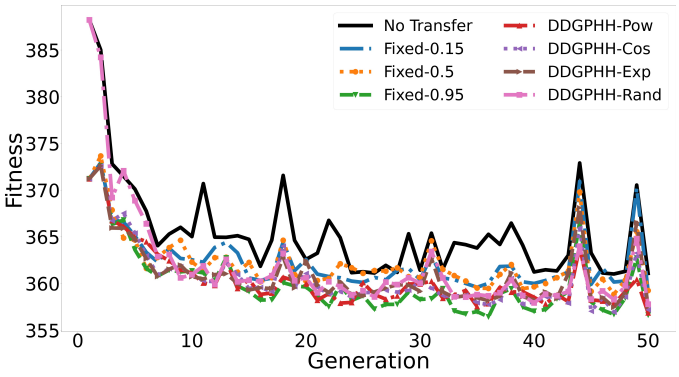
As is evident from Table III, *DDGPHH* with the *exponential* mutation strategy achieved better final performances compared to the GPHH as well as other strategies. As was explained in Section III, this strategy starts with a high mutation rate but the mutation rate decreases quickly, and reach the lower bound after a few generations.. This characteristic separates this strategy from the other ones that also start with a high mutation rate but decrease it more slowly. As a result, the fact that the *exponential* strategy has yielded the best results indicates that performing extra explorations after the transfer of knowledge and then, letting GP take its normal course of search can be beneficial for GPHH. This observation is confirmed further with the results of the *Power* strategy that does not decrease the mutation rate as quickly as the *Exponential* strategy. The *Cos* strategy also starts with a high mutation rate and allows the mutation probability to decrease to a small value so that GP can have the chance of exploiting the regions discovered by the high mutation rate. This strategy then increases the mutation rate to allow GP more explorations. According to Table III, although this strategy can be effective in some cases, it is better to have a strategy that does not increase the probability after decreasing it.

DDGPHH accompanies transfer learning with an increased mutation rate. In order to find out the role that the increased mutation rate can have on any performance improvements, we also consider the case that transfer learning is disabled in *DDGPHH* so that the resultant GP is equipped only with an adaptive mutation rate. This algorithm is referred to as *DDGPHH-Rand*. The *DDGPHH-Rand* method is a version of *DDGPHH-Exp* that an exponential adaption strategy without any knowledge transfer. According to Table III, this method outperformed GPHH in two experiments. These results indicate that the increased diversity can have a positive impact on GP performance for this problem. However, since *DDGPHH-Exp* performs better, it shows that the transferred knowledge has a more prominent contribution.

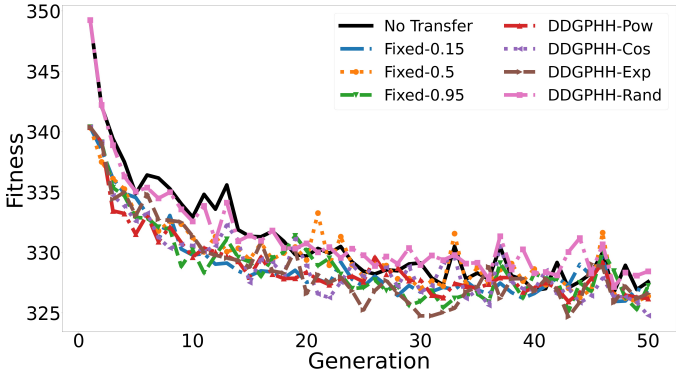
The *Fixed-0.95* and *Fixed-0.5* methods also start with a high probability rate but do not decrease it later and have a fixed mutation probability throughout the search process. As can be seen from Table III, these methods did not perform as effectively as the exponential strategy either. This indicates that having a large mutation rate in every generation and focusing extensively on exploration is not helpful. It is interesting to note that in one of the experiments, the *Fixed-0.15* method performed significantly better than GPHH without transfer. This method performs knowledge transfer but does not change mutation rate. Considering the fact that other *FullTree* methods did not show this performance, it could be inferred that although the simple transfer of population has some potential for helping GP, it is not enough and extravagant mutation rates cannot be very helpful either.

Figure 3 presents the convergence curve of the algorithms. As is seen in the figure, the transfer of knowledge from the source domain helps GP start with a good initial state but after a few generations, vanilla GP manages to catch up with other methods soon. This confirms that transfer learning has lead GP fall in local optima from which it cannot free itself.

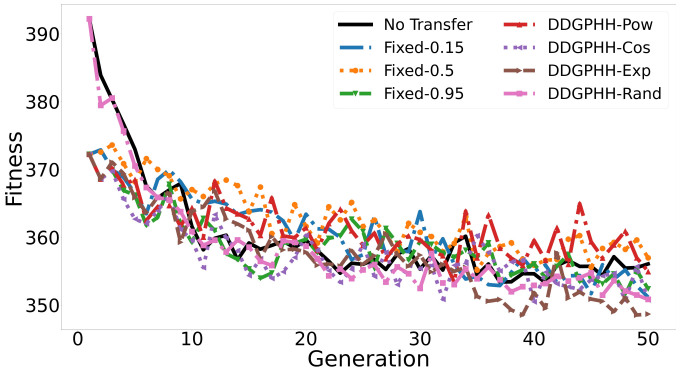
An important goal of transfer learning is to reduce the training cost of optimisation algorithms [31], [32]. Accordingly, Table IV presents the amount of time, in terms of GP generations, that can be saved by applying the transfer learning methods. The numbers are calculated by considering the earliest generation at which the performance of a transfer learning method is statistically similar to the final performance of GP without any transfer learning for at least three consecutive generations. As can be seen, in all cases, transfer learning has helped GP to achieve comparable results to the final performance of GP without transfer much earlier. However,



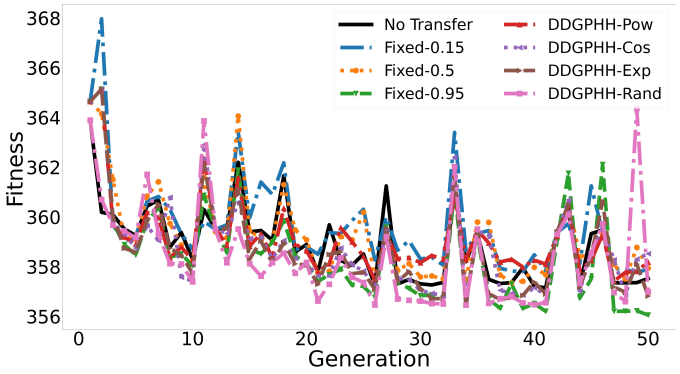
(a) *GDB1* from 5 to 6 vehicles



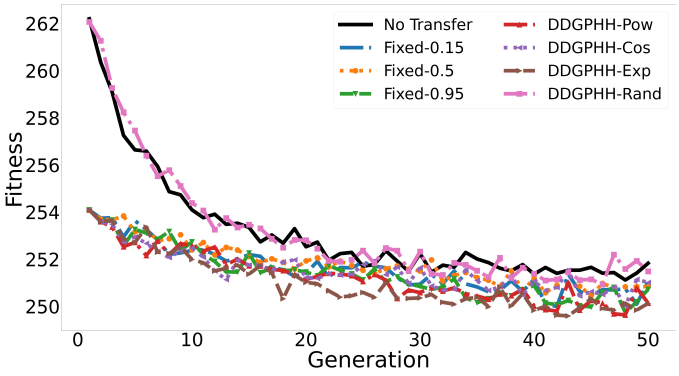
(b) *GDB3* from 5 to 6 vehicles



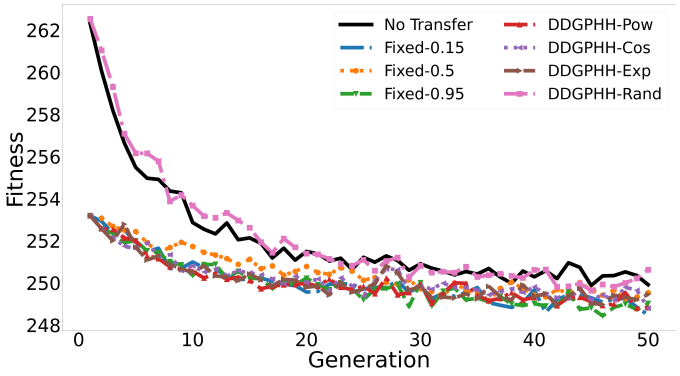
(c) *GDB6* from 5 to 4 vehicles



(d) *GDB7* from 5 to 6 vehicles



(e) *GDB23* from 10 to 9 vehicles



(f) *GDB23* from 10 to 11 vehicles

Fig. 3: Convergence curve of algorithms for different experiments

TABLE IV
AVERAGE IMPROVEMENT IN GP **TRAINING TIME**

	Fixed-0.15	Fixed-0.5	Fixed-0.95	DDGPHH-Pow	DDGPHH-Cos	DDGPHH-Exp
Imprv.	79%	77.66%	87.66%	85.00%	87.33%	86.00

the results in Tables III and IV reveal an interesting pattern. Although most transfer learning methods have been able to achieve the final performance of GPHH much earlier, none of them, except *DDGPHH-Exp* was able to improve the final performance. This is due to the lack of diversity in the initial population, which is consistent with our preliminary studies.

V. CONCLUSIONS

In this paper, we argued that in the context of transfer learning, when the source and target domains are similar, the act of knowledge transfer can contribute to the exploitation phase of GP. One consequence of this phenomenon is that the balance between exploration and exploitation phases of GP can be disturbed due to the transfer of knowledge. This issue becomes more important when we consider the fact that the GP population in the source domain usually converges to the regions of the search space that have a good potential for containing the solution. As a results, the GP population

in the target domain can lack diversity. Furthermore, it is rarely the case that the source and target domains share the same solutions. Hence, the simple transfer of population is likely to make GP get stuck in local optima. To address this issue, in this work, we proposed to increase the diversity of GP population after the knowledge transfer to enhance its exploration capability and decrease the chance of getting stuck in local optima. We achieved this by developing a simple yet effective adaptive mutation rate approach.

Our experimental studies demonstrated that increasing the diversity by simply using a high mutation rate is not very helpful. However, taking the strategy of increasing the mutation rate to a high value for a few generations and then decreasing it shortly has the best potential for helping GP employ the transferred knowledge. With this strategy, GP will have the ability to explore the search space a little more to neutralise the locality imposed by knowledge transfer and then, resume its normal course of exploration and exploitation.

Our experiments demonstrated the importance of considering population diversity after a knowledge transfer. However, the work in this paper is a preliminary effort and in our future work, we intend to utilise more elaborate methods of performing knowledge transfer that does not hurt population diversity.

REFERENCES

[1] B. L. Golden and R. T. Wong, "Capacitated arc routing problems," *Networks*, vol. 11, no. 3, pp. 305–315.

[2] J. F. Campbell and A. Langevin, *Roadway Snow and Ice Control*. Boston, MA: Springer US, 2000, pp. 389–418.

[3] S. K. Amponsah and S. Salhi, "The investigation of a class of capacitated arc routing problems: the collection of garbage in developing countries," *Waste Management*, vol. 24, no. 7, pp. 711–721, 2004.

[4] S. Wøhlk, *A Decade of Capacitated Arc Routing*. Boston, MA: Springer US, 2008, pp. 29–48.

[5] L. Muyldermans and G. Pang, *Chapter 10: Variants of the Capacitated Arc Routing Problem*, pp. 223–253.

[6] Y. Mei, K. Tang, and X. Yao, "Capacitated arc routing problem in uncertain environments," in *IEEE Congress on Evolutionary Computation*, 2010, pp. 1–8.

[7] Y. Liu, Y. Mei, and M. Zhang, "Automated Heuristic Design Using Genetic Programming Hyper-Heuristic for Uncertain Capacitated Arc Routing Problem," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2017, pp. 290–297.

[8] S. Wang, Y. Mei, and M. Zhang, "Novel ensemble genetic programming hyper-heuristics for uncertain capacitated arc routing problem," in *Proceedings of the Genetic and Evolutionary Computation Conference on - GECCO '19*. ACM Press, 2019, pp. 1093–1101.

[9] M. A. Ardeh, Y. Mei, and M. Zhang, "Transfer Learning in Genetic Programming Hyper-heuristic for Solving Uncertain Capacitated Arc Routing Problem," in *IEEE Conference on Evolutionary Computation - Proceedings*, 2019, pp. 49–56.

[10] Y. Mei, S. Nguyen, B. Xue, and M. Zhang, "An Efficient Feature Selection Algorithm for Evolving Job Shop Scheduling Rules With Genetic Programming," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 5, pp. 339–353, oct 2017.

[11] F. Zhang, Y. Mei, and M. Zhang, "A two-stage genetic programming hyper-heuristic approach with feature selection for dynamic flexible job shop scheduling," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 347–355.

[12] B. Tan, H. Ma, and Y. Mei, "A hybrid genetic programming hyper-heuristic approach for online two-level resource allocation in container-based clouds," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, 2019, pp. 2681–2688.

[13] J. Wang, K. Tang, J. A. Lozano, and X. Yao, "Estimation of the Distribution Algorithm With a Stochastic Local Search for Uncertain Capacitated Arc Routing Problems," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 96–109, 2016.

[14] C. Prins and S. Bouchenoua, *A Memetic Algorithm Solving the VRP, the CARP and General Routing Problems with Nodes, Edges and Arcs*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 65–85.

[15] B. Al-Helali, Q. Chen, B. Xue, and M. Zhang, "Multi-tree genetic programming for feature construction-based domain adaptation in symbolic regression with incomplete data," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, ser. GECCO '20. Association for Computing Machinery, 2020, p. 913–921.

[16] Q. Chen, B. Xue, and M. Zhang, "Improving symbolic regression based on correlation between residuals and variables," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, ser. GECCO '20. Association for Computing Machinery, 2020, p. 922–930.

[17] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, USA: MIT Press, 1992.

[18] Y. Mei and M. Zhang, "Genetic Programming Hyper-heuristic for Multi-vehicle Uncertain Capacitated Arc Routing Problem," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 2018, pp. 141–142.

[19] J. Maclachlan, Y. Mei, J. Branke, and M. Zhang, "An improved Genetic Programming Hyper-heuristic for the Uncertain Capacitated Arc Routing Problem," in *AI 2018: Advances in Artificial Intelligence*, 2018, pp. 1–12.

[20] S. Wang, Y. Mei, and M. Zhang, "A multi-objective genetic programming hyper-heuristic approach to uncertain capacitated arc routing problems," in *Proceedings of the IEEE World Congress on Computational Intelligence*. IEEE, mar 2020.

[21] J. MacLachlan, Y. Mei, J. Branke, and M. Zhang, "Genetic programming hyper-heuristics with vehicle collaboration for uncertain capacitated arc routing problems," *Evolutionary Computation*, nov 2019.

[22] D. O'Neill, H. Al-Sahaf, B. Xue, and M. Zhang, "Common subtrees in related problems: A novel transfer learning approach for genetic programming," *2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings*, pp. 1287–1294, 2017.

[23] E. Haslam, B. Xue, and M. Zhang, "Further investigation on genetic programming with transfer learning for symbolic regression," in *IEEE Congress on Evolutionary Computation, Proceeding*, 2016, pp. 3598–3605.

[24] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, oct 2010.

[25] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang, "Transfer learning using computational intelligence: A survey," *Knowledge-Based Systems*, vol. 80, pp. 14–23, 2015.

[26] M. E. Taylor, S. Whiteson, and P. Stone, "Transfer learning for policy search methods," in *Proceedings of the Twenty-Third International Conference on Machine Learning*, 2006.

[27] T. T. H. Dinh, T. H. Chu, and Q. U. Nguyen, "Transfer learning in genetic programming," in *IEEE Congress on Evolutionary Computation*, 2015, pp. 1145–1151.

[28] M. Iqbal, B. Xue, H. Al-Sahaf, and M. Zhang, "Cross-Domain Reuse of Extracted Knowledge in Genetic Programming for Image Classification," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 4, pp. 569–587, 2017.

[29] M. A. Ardeh, Y. Mei, and M. Zhang, "A Novel Genetic Programming Algorithm with Knowledge Transfer for Uncertain Capacitated Arc Routing Problem," in *PRICAI 2019: Trends in Artificial Intelligence*, 2019, pp. 196–200.

[30] ——, "Genetic Programming Hyper-heuristic with Knowledge Transfer for Uncertain Capacitated Arc Routing Problem," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 2019, pp. 334–335.

[31] M. A. Ardeh, Y. Mei, and M. Zhang, "Genetic programming hyper-heuristics with probabilistic prototype tree knowledge transfer for uncertain capacitated arc routing problems (accepted to appear)," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020.

[32] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. IGI Global, 2010, pp. 242–264.