Genetic Programming Hyper-Heuristic for Multi-Vehicle Uncertain Capacitated Arc Routing Problem

Yi Mei
Victoria University of Wellington
New Zealand
yi.mei@ecs.vuw.ac.nz

Mengjie Zhang
Victoria University of Wellington
New Zealand
mengjie.zhang@ecs.vuw.ac.nz

ABSTRACT

This paper investigates evolving routing policy for general Uncertain Capacitated Arc Routing Problems (UCARP) with any number of vehicles, and for the first time, designs a novel model for online decision making (i.e. *meta-algorithm*) for multiple vehicles in service simultaneously. Then, we develop a GPHH based on the meta-algorithm. The experimental studies show the GPHH can evolve much better policies than the state-of-the-art manually designed policy. In addition, the reusability of the evolved policies dramatically decreases when the number of vehicles changes, which suggests a retraining process when a new vehicle is brought or an existing vehicle breaks down.

CCS CONCEPTS

• **Theory of computation** → *Routing and network design problems*;

KEYWORDS

genetic programming, hyper-heuristic, uncertain capacitated arc routing problem

ACM Reference Format:
Yi Mei and Mengjie Zhang. 2018. Genetic Programming Hyper-Heuristic for Multi-Vehicle Uncertain Capacitated Arc Routing Problem. In *GECCO '18 Companion: Genetic and Evolutionary Computation Conference Companion, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3205651.3205661

1 INTRODUCTION

The Uncertain Capacitated Arc Routing Problem (UCARP) [3] has a wide range of real-world application in logistics and transportation domains. The problem aims to serve edges (arcs) in a connected graph with a set of routes (cycles), where the total cost of the routes is minimised under certain constraints. In UCARP, the travel cost between vertices in the graph and demand of tasks is unknown in advance, and is revealed during the process of executing the services. In this case, a preplanned solution may become worse or even infeasible.

Routing policy is an effective heuristic approach due to its ability to make real-time decisions (e.g. decide the task for a vehicle to serve next). Starting with a set of empty routes, a routing policy gradually builds the routes by repeatedly selecting a task from the unserved tasks for a vehicle to serve next.

Manually designing a routing policy requires much human effort and domain knowledge, and the manually designed policies are usually not good enough. Genetic Programming Hyper-Heuristic (GPHH) has been successfully applied to automatically evolve effective routing policies for UCARP (e.g. [2, 4]). However, in the existing studies, the routes are built sequentially, and the next route is opened after the previous one is closed. In other words, the existing studies were restricted to the single-vehicle case, as there is always a single vehicle on its way to serve the tasks. The existing works cannot handle the general multi-vehicle UCARP, in which there are multiple vehicles on the road simultaneously.

In this paper, we aim to develop a GPHH method for the general UCARP, in which the routes are built in parallel rather than sequentially. To the best of our knowledge, this paper is the first attempt to design routing policies for multi-vehicle UCARP.

2 PROPOSED APPROACH

In the GPHH, a routing policy is represented as a Lisp tree, which will be used as a priority function to select the task from a pool of candidate tasks for a vehicle to serve next. A routing policy is evaluated by a *meta-algorithm*. Given a sampled UCARP instance and a routing policy, a meta-algorithm generates a feasible solution. Then, given a training set $\mathcal{I}_{\text{train}}$, the fitness of a routing policy $h(\cdot)$ is defined as the average total cost of the solutions generated by $h(\cdot)$ on all the instances in $\mathcal{I}_{\text{train}}$. Existing meta-algorithms can be applied to only single-vehicle UCARP. In this paper, we will design a new meta-algorithm that can generate a solution for general multi-vehicle UCARP.

2.1 Meta-algorithm for Multi-Vehicle UCARP

The meta-algorithm is a discrete event simulation system that models a multi-vehicle decision making process. It consists of a *system state* and a *priority queue of events*. The system state includes the unserved and unassigned tasks so far, the remaining demand fraction of each task, the current partial routes and their currently served demands. At each step of the simulation, an event in the queue is triggered, and the state and event queue are updated. The simulation is stopped when all the tasks have been served and all the vehicles have returned to the depot.

There are three types of events during the UCARP decision making process as follows.

(1) *Refill event* occurs when a vehicle arrives at a node (e.g. intersection) on its way back to the depot to refill.

GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5764-7/18/07.
https://doi.org/10.1145/3205651.3205661

(2) *Serve event* occurs when a vehicle arrives at a node on its way to serve the next task.
(3) *Refill-and-serve event* occurs after a route failure. It 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.

They have different triggering methods. The detailed source code for the triggering methods and the proposed GPHH can be found from https://github.com/meiyi1986/gpucarp.

3 EXPERIMENTAL STUDIES

The proposed GPHH algorithm was evaluated on the UCARP instances extended from the static *Ugdb* and *Uval* instances [2], which contains 57 instances in total. The experiment contains a *training* and a *test* phase. During the training phase, the routing policies are trained by the GPHH over a set of *training samples*. Then, the routing policy with the best training fitness is applied to a set of unseen *test samples*, and the test performance is evaluated. In the experiment, the training set includes 5 samples, which are re-sampled in each generation using a different random seed. The test set includes 500 samples which are different from the training samples.

The population size is set to 1024, and the maximal depth is set to 8. The crossover, mutation and reproduction rates are 0.8, 0.15 and 0.05. The process stops after 51 generations. For each UCARP instance and m, the GPHH is run 30 times independently.

We have the following two research questions to answer:

(1) How does the number of vehicles affect the effectiveness of the GP-evolved routing policies?
(2) How reusable is a GP-evolved routing policy when the number of vehicles is changed.

To answer the above two research questions, the experimental studies are divided into two experiments.

Experiment 1. We compare the test performance of the GP-evolved routing policies under different numbers of vehicles. For the sake of simplicity, we denote GPHH(p,q) as the GPHH that trains routing policies with p vehicles and tests the best routing policy with q vehicles. In experiment 1, we set $p = q$, i.e. the training and test sets have the same number of vehicles.

First, we compare the GPHH with the path scanning (PS) [1] routing policy, which is a state-of-the-art manually designed routing policy. The overall test performance of the compared policies is shown in Table 1. From the figure, one can see that GPHH(p,p) significantly outperforms the PS policy for all $p = 1, \ldots, 5$. In addition, as p increases, the performance of both path scanning and the GP-evolved policies drops. This is because the decision making processes with more vehicles are more complicated.

Table 1: The overall test performance of the manually designed and GP-evolved policies with $p = 1, \ldots, 5$.

Policy	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
PS	362.48(170.41)	379.89(178.07)	391.25(185.16)	403.99(194.96)	403.59(187.29)
GPHH	**343.00**(5.67)	**349.60**(6.09)	**351.63**(6.00)	**354.36**(6.46)	**357.92**(6.40)

Experiment 2. For each UCARP instance and $q = 1$ and 5, we compared the test performance of the GPHH algorithm trained on different numbers of vehicles using the Wilcoxon's rank sum test with significance level of $\alpha = 0.05$. Tables 2 and 3 show the win-draw-lose results on all the 57 instances. From the tables, it is obvious that for each q, GPHH(q,q) performed the best, which is consistent with intuition. Even with a tiny change in q, the performance is decreased dramatically. This implies that the routing policy needs to be retrained whenever the number of vehicles is changed.

Table 2: The win-draw-lose mutual comparison results for all the 57 instances with $q = 1$.

	GPHH(2,1)	GPHH(3,1)	GPHH(4,1)	GPHH(5,1)
GPHH(1,1)	**55-2-0**	**57-0-0**	**57-0-0**	**57-0-0**
GPHH(2,1)	—	**16-39-2**	**34-23-0**	**45-12-0**
GPHH(3,1)	—	—	**18-37-2**	**29-27-1**
GPHH(4,1)	—	—	—	**16-39-2**

Table 3: The win-draw-lose mutual comparison results for all the 57 instances with $q = 5$.

	GPHH(2,5)	GPHH(3,5)	GPHH(4,5)	GPHH(5,5)
GPHH(1,5)	0-14-43	1-13-43	1-4-52	0-0-57
GPHH(2,5)	—	6-35-16	2-23-32	0-0-57
GPHH(3,5)	—	—	4-26-27	0-0-57
GPHH(4,5)	—	—	—	0-1-56

4 CONCLUSIONS AND FUTURE WORK

In this paper, we developed a GPHH for solving the general multi-vehicle UCARP for the first time. The experimental results showed that the proposed GPHH performed much better than the manually designed policies. In addition, we found that one should always retrain the policies when a new vehicle is bought or an existing vehicle breaks down.

In the future, we will further improve the performance of the GPHH by incorporating feature selection and construction techniques. We will also consider a combination of routing policy and preplanned robust solutions, so that one can take advantage of both offline optimisation and online decision making to achieve better performance. In addition, a main advantage of multi-vehicle UCARP over single-vehicle UCARP is that the vehicle can collaborate with each other in real time. We will explore the possibility of strengthening the real-time collaboration between vehicles to improve the overall quality of the routing plan.

REFERENCES

[1] Philippe Lacomme, Christian Prins, and Wahiba Ramdane-Cherif. 2004. Competitive Memetic Algorithms for Arc Routing Problems. *Annals of Operations Research* 131, 1 (2004), 159–185.
[2] Yuxin Liu, Yi Mei, Mengjie Zhang, and Zili Zhang. 2017. Automated heuristic design using genetic programming hyper-heuristic for uncertain capacitated arc routing problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 290–297.
[3] Y. Mei, K. Tang, and X. Yao. 2010. Capacitated arc routing problem in uncertain environments. In *IEEE Congress on Evolutionary Computation*. 1–8.
[4] Thomas Weise, Alexandre Devert, and Ke Tang. 2012. A Developmental Solution to (Dynamic) Capacitated Arc Routing Problems Using Genetic Programming. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation (GECCO '12)*. ACM, New York, NY, USA, 831–838.