Towards Interpretable Routing Policy: A Two Stage Multi-Objective Genetic Programming Approach with Feature Selection for Uncertain Capacitated Arc Routing Problem

Shaolin Wang, Yi Mei and Mengjie Zhang
Victoria University of Wellington
Wellington, New Zealand
Email: {shaolin.wang, yi.mei, mengjie.zhang}@ecs.vuw.ac.nz

***Abstract*—The Uncertain Capacitated Arc Routing Problem (UCARP) is the dynamic and stochastic counterpart of the well-known Capacitated Arc Routing Problem (CARP). UCARP has a wide range of real-world applications. One of the main challenge in UCARP is to handle the uncertain environment effectively. Routing policy-based approaches are promising technique for solving UCARP as they can respond to the uncertain environment in the real time. However, manually designing effective routing policies is time consuming and heavily replies on domain knowledge. Genetic Programming Hyper-heuristic (GPHH) has been successfully applied to UCARP to automatically evolve effective routing policies. However, the evolved routing policies are usually hard to interpret. In this paper, we aim to improve the potential interpretability of the GP-evolved routing policies by considering both program size and number of distinguished features. To this end, we propose a Two Stage Multi-Objective Genetic Programming Hyper Heuristic approach with Feature Selection (TSFSMOGP). We compared TSFSMOGP with the state-of-the-art single-objective GPHH, a two-stage GPHH with feature selection and a two-stage Multi-Objective GP. The experimental results showed that TSFSMOGP can evolve effective, compact, and thus potentially interpretable routing policies.**

***Keywords*—UCARP; GPHH; Routing policy; Feature Selection.**

I. Introduction

The Capacitated Arc Routing Problem (CARP) [1] is a classic combinatorial problem which a variety of real-world applications such as winter gritting [2], [3] and waste collection [4]. The main goal of CARP is to minimise the total cost of serving a set of *required edges* in a graph using a fleet of vehicles.

CARP has been proven to be NP-hard [5], and there have been extensive studies for solving it [1], [5]. However, most of the previous studies assume that the environment is static. All the parameters (e.g. task demand, travel cost) are fixed and known in advance. In reality, the environment is often dynamic and stochastic in the real world. For example, in snow removal, the amount of snow to be removed on a street varies from one day to another, and cannot be exactly known in advance.

Uncertain Capacitated Arc Routing Problem (UCARP) [6] was proposed to better reflect the real world. In UCARP, the demands and travel costs are uncertain. One of the main challenges caused by the uncertain environment is *route failure*. When the actual demand of a required edge is greater than expected, the remaining capacity of the vehicle is insufficient to complete the service, a route failure occurs. The vehicle has to go back to the depot to replenish in the middle of the service and come back to finish serving the edge again. A route failure might lead to a large recourse cost.

Due to the high computational cost, traditional solution optimisation approaches, such as mathematical programming [7] and evolutionary algorithms [8], cannot effectively handle the route failure. In general, traditional solution optimisation approaches try to optimise a robust solution beforehand. When a route failure occurs, a recourse operator is applied to modify the pre-planned solution. However, it is hardly possible to have a single robust solution that can deal with all the environments. In addition, it is time-consuming to re-optimise the entire remaining solution from scratch in real time.

The routing policy-based approaches [9] are considered as promising techniques to deal with the uncertain environment in UCARP. Unlike other solutions optimisation approaches, routing policies work as heuristics and create a solution based on the latest information on-the-fly. Routing policies can respond to the uncertain environment immediately by guiding a vehicle to determine which task to serve next when it becomes idle [9]. There are some manually designed effective routing policies (e.g. Path Scanning [8]).

The effectiveness of a manually designed routing policy can be affected by numerous factors, e.g. the scenario, the objective(s), and the graph topology [10]. In addition, it is very time-consuming to manually design effective routing policy for a given problem scenario. To this end, Genetic Programming Hyper-heuristic (GPHH) approaches have been applied to UCARP to automatically evolve effective routing policies [11]–[14].

Recently, interpretability becomes a hot topic in AI field. It is important for human users to understand and trust the AI model [15]. Interpretability is also an important aspect when apply GPHH to UCARP. However, most existing studies

on GPHH only focus on the effectiveness but ignore the interpretability. Only few studies [16]–[18] focus on the interpretability aspect by reducing the size of the evolved routing policies. However, they ignore another aspect — number of distinguished features. The number of distinguished features seem to play a major role for improving the interpretability [19], [20]. Intuitively, it is much easier to interpret a GP-evolved routing policy with fewer features.

Feature selection is commonly applied to select feature subset from all features. It can be applied to enhance the quality of the feature space, simplify the learned model, speed up the learning process and improve the quality of the model. Feature selection have been applied to many problems, such as classification, clustering and regression problems [21]. Normally, GP is regarded as an embedded feature selection approach [21]. However, its ability is limited. For example, even best GP individuals still contain many redundancy components. Therefore, a feature subset selection is necessary. In GP, feature weighting and feature ranking techniques are often utilised for feature subset selection [22]–[25].

To further improve the potential interpretability of the GP evolved rules, in this paper we consider both the program size and number of distinguished features, and optimise them along with the effectiveness under a multi-objective GP framework. To this end, this paper proposes a Two Stage Multi-Objective Genetic Programming Hyper Heuristic approach with Feature Selection (TSFSMOGP). The main idea of TSFSMOGP is to divide the GP process into two stages. It focuses on reducing the number of distinguished feature in evolve routing policies by applying feature selection in the first stage, and then reducing the tree size of the evolved routing policies by utilising MOGP in the second stage. This paper has the following research objectives:

1) Develop a Two Stage Multi-Objective Genetic Programming Hyper Heuristic approach with Feature Selection (TSFSMOGP) for UCARP to optimise the effectiveness, program size and number of distinguished features in the rule simultaneously;
2) Verify the effectiveness of TSFSMOGP on a set of UCARP instances;
3) Interpret and analyse the obtained routing policies and solutions evolved by TSFSMOGP.

The rest of this paper is organised as follows. Section II introduces the background. Section III describes the newly proposed TSFSMOGP. Section IV gives the experimental studies. Section V gives the conclusions and possible future directions.

II. Background

A. Uncertain Capacitated Arc Routing Problem

A UCARP instance can be represented by a connected graph $G(V, E)$, where V indicates the set of vertices and E indicates the set of edges. Each edge $e \in E$ has a a positive random deadheading cost $\tilde{dt}(e)$. $\tilde{dt}(e)$ indicates the cost of traversing the edge. $E_R \subseteq E$ indicates the set of edges that need to be served. Each required edge $e_R \in E_R$ has a positive serving cost $s(e_R)$ and a positive random demand $\tilde{d}(e_R)$. Q indicates the capacity of each vehicle. $v_0 \in V$ indicates the depot. The goal is to minimise the total cost of serving all the tasks in E_R. Each vehicle must start and finish at v_0, and the total demand served between two subsequent depot visits cannot exceed the capacity of the vehicle.

There are many random variables in a UCARP instance, each of which can have different samples. In this case, a UCARP instance can have many different samples. In a UCARP instance sample, each random variable has a realised value that is not known in advance.

For example, the realised deadheading cost of an edge is unknown in advance and is revealed during the vehicle traversing over the edge.

The uncertain environment can lead to two kinds of failures, *route failure* and *edge failure*. Route failure occurs when the actual demand of a task exceeds the remaining capacity of the vehicle. It can be repaired by a recourse operator. A typical recourse operator is that the vehicle goes back to the depot to refill and return to the failed task to complete the remaining service. Edge failure occurs when the edge in the route becomes inaccessible. Edge failure can be repaired by finding the shortest path (e.g., Dijkstra's algorithm) under the current situation.

A solution to a UCARP instance sample is represented as $S = (S.X, S.Y)$. $S.X = \{S.X^{(1)}, \ldots, S.X^{(m)}\}$ is a set of vertex sequences, where $S.X^{(k)} = (S.x_1^{(k)}, \ldots, S.x_{L_k}^{(k)})$ stands for the k^{th} route. $S.Y = \{S.Y^{(1)}, \ldots, S.Y^{(m)}\}$ is a set of continuous vectors, where $S.Y^{(k)} = (S.y_1^{(k)}, \ldots, S.y_{L_k-1}^{(k)})$ ($S.y_i^{(k)} \in [0, 1]$) is the fraction of demand served along the route $S.X^{(k)}$. For example, if $S.y_3^{(1)} = 0.7$, then $(S.x_3^{(1)}, S.x_4^{(1)})$ is a task and 70% of its demand is served at position 3 of the route $S.X^{(1)}$. Under the above representation, the problem can be formulated as follows.

$$\min \ \mathrm{E}_{\xi \in \Xi}[C(S_\xi)], \tag{1}$$

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

$$\sum_{k=1}^{m} \sum_{i=1}^{L_k-1} S_\xi.y_i^{(k)} \times S_\xi.z_i^{(k)}(e) = 1, \forall e : d_\xi(e) > 0, \tag{3}$$

$$\sum_{k=1}^{m} \sum_{i=1}^{L_k-1} S_\xi.y_i^{(k)} \times S_\xi.z_i^{(k)}(e) = 0, \forall e : d_\xi(e) = 0, \tag{4}$$

$$\sum_{i=1}^{L_k-1} d_\xi\left(S_\xi.x_i^{(k)}, S_\xi.x_{i+1}^{(k)}\right) \times S_\xi.y_i^{(k)} \leq Q, \forall k = 1, \ldots, m, \tag{5}$$

$$\left(S_\xi.x_i^{(k)}, S_\xi.x_{i+1}^{(k)}\right) \in E, \tag{6}$$

$$S_\xi.y_i^{(k)} \in [0, 1], \tag{7}$$

where in Eqs. (3) and (4), $S_\xi.z_i^{(k)}(e)$ equals 1 if $\left(S_\xi.x_i^{(k)}, S_\xi.x_{i+1}^{(k)}\right) = e$, and 0 otherwise. Eq. (1) is the objective function, which is to minimise the expected total cost $C(S_\xi)$ of the solution S_ξ in all possible environments $\xi \in \Xi$. Eq. (2) indicates that in all S_ξ, the routes start and end at the depot. Eqs. (3) and (4) mean that each task is served

exactly once (the total demand fraction served by all vehicles is 1), while each non-required edge is not served. Eq. (5) is the capacity constraint, and Eqs. (6) and (7) are the domain constraints of $S_\xi.X$ and $S_\xi.Y$.

B. Related Work

The CARP is a classical combinatorial optimisation problem. Most existing approaches focus on the static CARP. Exact approaches were firstly applied to CARP. Exact approaches ensure that the obtained solutions are optimal. Golden and Wong [7] developed an integer linear programming model and solved it using branch-and-cut. However, this approach can only solve small instances due to the NP-hardness of CARP. Exact approaches can become very time consuming as the problem becomes large. To make an improvement, meta-heuristic approaches are used in recent years. They are usually much cheaper than exact approaches. Various meta-heuristic algorithms have been proposed for CARP. Tabu search [26], [27] approach was applied to static CARP. After that, Genetic Algorithm [28] and Memetic Algorithms [8], [29], [30] were proposed to solve the problem better. Lacomme et al. [31] proposed an ant colony schema, and Doerner et al. [32] developed an ant colony optimization. Although the existing approaches showed good performance on static CARP, they cannot handle the route failure effectively in UCARP.

There are two commonly used approaches for solving UCARP, the proactive approaches [33]–[35] and the reactive approaches [11]–[13]. The proactive approaches aim to find robust solutions that can handle all the possible realisations of the random variables. In contrast, reactive approaches mainly focus on using GPHH to evolve routing policies which construct the solution gradually in real time.

Weise et al. [36] first proposed a GPHH for UCARP with a single vehicle and the results showed that the evolved routing policies outperformed manually designed routing policies. To speed up the training process and improve the effectiveness, Liu et al. [11] designed a novel and effective meta-algorithm which filters irrelevant candidate tasks during the decision-making process and achieved better effectiveness than the GPHH in [36]. To better reflect the reality, Mei et al. [13] extended the model from single-vehicle to multiple-vehicle version, so that the solution can be generated with multiple vehicles on the road simultaneously. MacLachlan et al. [12] proposed a novel task filtering method and an effective look-ahead terminal for UCARP. Recently, Ardeh et al. [37], [38] utilised transfer learning to speed up the training process in new Scenarios. Liu et al. [39] proposed a new predictive-reactive approach with Genetic Programming, cooperative coevolution was applied to evolve task sequence and a recourse policy simultaneously.

The existing GPHH approaches managed to obtain effective routing policies for UCARP. Interpretability is also an important aspect in GPHH for UCARP. Only few studies [16]–[18] have been made on this aspect by directly reducing the size of routing policies. The most effective one is the Multi-Objective approach [18]. It can obtain much smaller tree size with comparable result with baseline GPHH. The number of distingguished features is also play a major role for improving the interpretability [19], [20]. Intuitively, routing policies with fewer features will be easier to interpret.

Feature selection is an effective approach for selecting a subset of relevant features. It can remove irrelevant, redundant and misleading features [21]. Feature selection has been successfully applied to solve many problems, such as classification [40], clustering [41] and regression [42]. In UCARP, many characteristics that can be used as features for GP. It is hard to identify which feature is more relevant in a specific scenario. Normally, it is better to include as more features as possible in GP so that GP can evolve better routing policies. However, more features usually lead to larger search space and worse interpretability. Feature selection can be a good technique to handle these issues.

Typically, GP is regarded as an embedded feature selection approach [21]. It can automatically find a feature subset in its best individual in the last population. However, to our preliminary work, even the best individual in the last population still contains some irrelevant features. In GP, feature ranking techniques are often used for feature subset selection [22]–[25]. Friedlander et al. [22] proposed a feature ranking method based on frequency analysis. However, the feature ranking calculated by this method might not be accurate. Mei et al. [24] pointed out that the rank calculated by this method might not be accurate. For example, subtree X - X increases the frequency of feature X. However, subtree X - X makes no contribution to the whole GP tree. To this end, Mei et al. [24] proposed a feature selection approach based on the contribution of each feature (FS-GP) for Job Shop Scheduling Problem (JSS). The result showed that FS-GP can evolve rules with fewer features and achieved comparable performance with baseline GPHH.

In this paper, we aim to propose a new TSFSMOGP which combines FS-GP and MOGP with a two stage framework for UCARP. The TSFSMOGP will consider both tree size of evolved routing policies and the number of distinguishied features.

III. Two Stage Multi-Objective Genetic Programming Hyper Heuristic with Feature Selection

In this section, we describe the Two Stage Multi-Objective Genetic Programming Hyper Heuristic with Feature Selection (TSFSMOGP). First, we give the overall framework of the algorithm. Then, we describe the major components, e.g. individual representation, fitness evaluation, feature selection and multi-objective process, in detail.

A. Overall Framework

The flow chart of TSFSMOGP is given in Figure 1. To improve the potential interpretability of the evolved routing policies, we reduce the number of distinguished features in the routing policies by feature selection. Besides, we apply multi-objective GP to reduce the tree size of the evolved routing

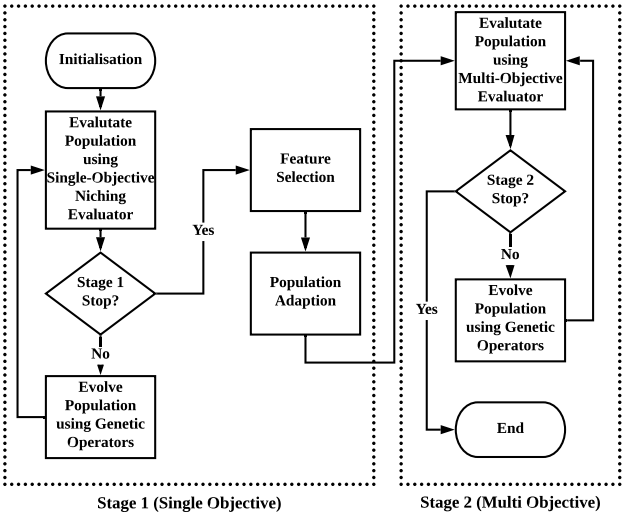

Fig. 1: The flowchart of TSFSMOGP

policies. For the feature selection, the implementation is quite similar with FSGP in [25], and for MOGP, the implementation is quite similar with the TS-GPHH in [18]. TSFSMOGP consists of two stages. The first stage is from generation 0 to *pre-generation*, i.e. 50 generations. The second stage is the generation after *pre-generation*. In the first stage, it is a single-objective optimisation stage which only concentrates on the effectiveness (total-cost). GP proceeds with a niching evaluator to select a diverse set of good individuals to perform feature selection. Once the diverse set of good individuals is obtained, the feature selection mechanism will be invoked to select a set of informative features for evolving routing policies in the second stage. Then, some good individuals will be adapted to the second stage by replacing unselected features in the individuals with some specific constant 1. Thus, we can keep some good individuals from the first stage. In the second stage, the training process is a multi-objective GPHH process in which the terminal set is changed to the selected feature subset. In addition, both effectiveness and tree size are taken account during the GP process in the second stage. Finally, the individual from the Pareto front that has the best effectiveness (i.e. total cost) is returned as the algorithm output.

In summary, the TSFSMOGP contains two consecutive phases. The first phase is the single-objective feature selection phase with niching techniques. The second phase is the multi-objective phase, routing policies are evolved based on the features selected in the first phase and multi-objective evolutionary process.

B. Individual Representation

In TSFSMOGP, each routing policy is essentially an arithmetic priority function, which is represented as a tree in GP. The routing policy is applied to a solution construction process, which is modelled as a decision making process. During the process, the vehicles serve a task at a time. Once

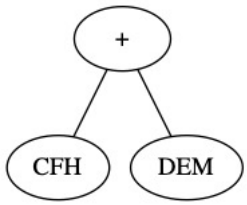

Fig. 2: An example of GP tree.

a vehicle is ready to serve next task, the routing policy is applied to calculate the priority of each unserved task. Then, the most prior feasible task will be selected to serve next. For example, CFH refers to the cost from the current location to the candidate task, DEM refers to the expected demand of the candidate task, if a priority function is "CFH+DEM", then the routing policy prefers tasks that require less serving demand and close to the current location. The process completes when all the tasks have been served, and the routes of the vehicles are returned as the solution constructed by the routing policy.

C. Fitness Evaluation

Given a routing policy rp, the fitness measurement is defined as the average solution quality (i.e. total cost) generated by rp over a set of instance samples $\mathcal{S}$. Specifically,

$$fit(rp) = \frac{1}{|\mathcal{N}|} \sum_{s \in \mathcal{N}} tc(rp, s), \tag{8}$$

where $|\mathcal{N}|$ is the number of instance samples. $tc(rp, s)$ stands for the total cost of the solution obtained by rp on sample s.

We randomly re-sample a subset of training instances for the fitness evaluation. Such instance rotation has been commonly used in other studies [13], [43] and has been demonstrated to be able to improve the generalisation of the evolved solutions.

D. Niching technique and Feature Selection

The effective niching technique based feature selection for JSS was proposed in [24]. Zhang et al. [25] applied this approach to dynamic flexible JSS. In this paper, we apply this idea to UCARP but with a number of adjustments.

1) Niching: It is important to identify which individuals should be used as baseline to analyse the features. On one hand, if we select features based on individuals with bad fitness, it is less possible to select informative features. On the other hand, if we only select features from the best individual, it is very easy to bias to a local optimal. Thus, it is necessary to find a way that can find a diverse set of individuals with good fitness. To this end, niching technique is applied. Niching technique is mainly used in the first stage to find a diverse set of individuals with good fitness. In this case, informative features can be selected based on diverse good individuals to avoid bias to one local optimal.

Clearing method [44], which is a simple yet effective niching algorithm, is applied. Normally, clearing method reduces the number of poor individuals in crowded areas within the search space. In this paper, clearing method is used to obtain

a diverse set of good individuals. In this paper, the clearing method is applied after the traditional fitness evaluation and poor individual is defined as duplicate. According to the suggestions in [24], we set two parameters in clearing method, they are radius γ and capacity k. The radius is used to control the range of each niche and the capacity is used to declare how many individuals should be in each niche.

2) Feature selection: As we mentioned in Section I, it is important to evaluate the contribution of each feature so that informative features can be selected. Following the suggestions in [24], there are three main steps to estimate the contribution of a feature. Firstly, the measured feature is replaced by the constant value 1.0 in all individuals in the diverse set. Then, evaluate the fitness of all new individuals (replaced by constant value 1.0). Secondly, compare the difference between the old individuals and new individuals. This is shown in Eq. 9. If $diff(f) > 0$, it means that the fitness gets worse without feature f. This indicates that f is an important feature. The larger the $diff(f)$ is, the more important the feature f is. After calculation the importance of each feature on all individuals. All the fitnesses are normalised based on Eq. 10 and a voting weight is calculated based on Eq. 11. When the $diff(f) > 0.001$, the voting weight $w(f)$ is assigned to the feature as its voting score. Finally, a feature is selected when its voting score is higher than the half of the total voting weights.

$$diff(f) = fit(f|1) - fit(f) \tag{9}$$

$$fit_{norm(f)} = \frac{1}{1 + fit(f)} \tag{10}$$

$$w(f) = \max(\frac{fit_{norm(f)} - fit_{min}}{fit_{max} - fit_{min}}, 0) \tag{11}$$

E. Multi-Objective GP

MOGP is utilised in the second stage of TSFSMOGP to reduce the tree size. The idea is quite similar with TS-MOGP-s in [18] except that we employ a niching technique to obtain a diverse set of individuals for feature selection in stage one. Tree size and effectiveness are used as two objectives. Tournament selection is modified to select parents based on both objectives. Specifically, given k randomly selected individuals, the non-dominated individual is selected to be the parent. If there are multiple non-dominated individuals, the first one identified during the comparison is selected. The final output of TSFSMOGP is the routing policy with the best effectiveness in the final Pareto Front since effectiveness is still the primary objective in our problem.

IV. Experimental Studies

To evaluate the performance of the proposed approach, we test them on a number of UCARP instances. We compare TSFSMOGP with the GPHH [13] which is the vanilla version that without any feature selection, TSFSGP which applies feature selection for UCARP and TSMOGP [18] which is a multi-objective GPHH approach for UCARP.

TABLE I: The terminal set in the experiments.

Terminal	Description
SC	cost of serving the candidate task
CR	cost from the depot to the current location
DEM	expected demand of the candidate task
DEM1	demand of unserved task that closest to the candidate task
FULL	fullness (served demand over capacity) of the vehicle
CFH	cost from the candidate task to the current location
CFD	cost from the head node of the task to the depot
CFR1	cost from the closest other route to the candidate task
CTT1	the cost from the candidate to its closest remaining task
FRT	fraction of unserved tasks
FUT	fraction of unassigned tasks
CTD	cost from the depot to the candidate task
RQ	remaining capacity of the vehicle
RQ1	remaining capacity of the closest other route to the candidate task
ERC	a random constant value

A. Experiment Setup

We select 6 representative UCARP instances from the commonly used Ugdb and Uval datasets [11]–[13] to evaluate the performance of the proposed approach. The problem size of these instances varies from small (22 tasks and 5 vehicles) to large (97 tasks and 10 vehicles). Therefore, we can evaluate the effectiveness of the proposed algorithms in different problem scenarios. The experiment consists of two phases, training and test phases. In the training phase, we will obtain a routing policy based on the training set T_{train}. During the training phase, all the algorithms use 5 training samples during the evaluation. Note that, for feature selection approaches, there is a feature selection process in the training phase. In the test phase, the routing policy will be tested on an unseen test set, which contains 500 test samples that can avoid testing bias.

The function set is set to $\{+, -, \times, /, \min, \max\}$ where "/" operator is protected divide which returns 1 if divided by 0. In addition, the terminal set is shown in Table I.

The number of generations is 100 for all compared algorithms. Note that, for TSFSMOGP, the first 50 generations are the first stage which implement the feature selection. The population size for all the compared algorithms is 1000. For the initialisation, ramp-half-and-half initialisation is used. The ratio of crossover to mutation to reproduction is 0.8 : 0.15 : 0.05. The tournament selection size is 7. The maximal depth is 8. Elitism size for all compared approaches is 10. For TSFSMOGP, the radius and capacity are set to 0 and 1. The size of the diverse set is set to 30.

All the algorithms are implemented on Evolutionary Computation Java (ECJ) package [45]. Each algorithm was run 30 times independently for each UCARP instance.

B. Results and Discussions

Table II shows the mean and standard deviation of test performance of the compared algorithms. The Wilcoxon rank sum test with significance level of 0.05 is used to compare each algorithm with TSFSMOGP. For each compared algorithm, if its test performance is statistically significantly lower (better) than, higher (worse) than, and comparable with that of TSFSMOGP, the corresponding entry is marked with (+), (-),

TABLE II: The mean and standard deviation of the **test performance** of the compared algorithms. For each method, (+), (-) and (=) indicates it is significantly lower (better) than, higher (worse) than, and comparable with TSFSMOGP.

Instance	GPHH	TSFSGP	TSMOGP	TSFSMOGP
Ugdb1	351.25(14.66)(=)	362.43(62.33)(=)	353.76(10.1)(-)	348.84(8.9)
Ugdb2	367.73(4.77)(=)	368.47(8.1)(=)	371.22(11.52)(=)	368.23(7.92)
Ugdb8	430.79(8.16)(=)	433.62(7.8)(=)	450.48(22.59)(-)	435.2(32.49)
Ugdb23	250.62(2.98)(-)	250.08(2.49)(-)	252.19(2.82)(-)	248.83(2.16)
Uval9D	476.86(10.82)(=)	476.96(10.86)(=)	480.11(19.84)(=)	478.31(11.31)
Uval10D	622.17(16.77)(=)	619.9(6.54)(=)	623.23(8.65)(=)	620.77(12.32)

TABLE III: The mean and standard deviation of **the number of distinguished features** in the routing policies evolved by the compared algorithms.

Instance	GPHH	TSFSGP	TSMOGP	TSFSMOGP
Ugdb1	10.8(1.9)(-)	6.67(2.25)(=)	7.27(2.78)(=)	6.3(2.1)
Ugdb2	11.7(1.3)(-)	7.1(2.11)(=)	8.07(2.08)(-)	6.7(1.9)
Ugdb8	9.3(1.6)(-)	7.73(2.08)(=)	8.0(3.03)(-)	7.1(2.0)
Ugdb23	10.9(1.8)(-)	8.33(2.83)(=)	7.6(2.51)(=)	7.4(2.2)
Uval9D	10.6(2.2)(-)	7.3(2.14)(=)	8.83(2.28)(-)	6.8(1.9)
Uval10D	9.1(1.7)(-)	7.43(2.14)(=)	7.43(2.4)(=)	6.6(1.9)

or (=), respectively. Note that, for TSMOGP and TSFSMOGP, only the individuals with best effectiveness in the final Pareto Front from each run are involved in comparisons.

From Table II, we can see that TSFSMOGP can obtain comparable test performance with GPHH and TSFSGP on 5 out of 6 instances. Specifically, it can outperform GPHH and TSFSGP on Ugdb23. In addition, TSFSMOGP can perform significantly better than TSMOGP on 3 out 6 instances and obtain comparable results on other instances.

Table III shows the mean and standard deviation of the number of distinguished features in the routing policies evolved by the compared algorithms. The Wilcoxon rank sum test with significance level of 0.05 is also applied. From Table III, TSFSMOGP outperforms GPHH on all instances. We can also see that TSFSMOGP performs comparable with TSFSGP as both methods do feature selection at the end of the first stage. There is one interesting observation that TSMOGP can also evolve routing policies with fewer distinguished features than GPHH. This is because the number of features can be reduced when we reduce the tree size of the evolved routing policies. TSFSMOGP can still perform significantly better than TSMOGP on 3 out of 6 instances.

Table IV shows the mean and standard deviation of the number of nodes in the routing policies evolved by the compared algorithms. The Wilcoxon rank sum test with significance level of 0.05 is also applied. From Table IV, it can be seen that the tree size of routing policies evolved by TSFSMOGP is much smaller than GPHH and TSFSGP on all instances. This is expected, because neither GPHH nor TSFSGP applies any method to reduce the tree size. TSMOGP can obtain smaller routing policies than TSFSMOGP. However, this is achieved by sacrificing test performance.

TABLE IV: The mean and standard deviation of **the number of nodes** in the routing policies evolved by the compared algorithms.

Instance	GPHH	TSFSGP	TSMOGP	TSFSMOGP
Ugdb1	83.0(21.64)(-)	86.47(31.25)(-)	31.67(17.07)(=)	39.9(18.4)
Ugdb2	88.2(23.16)(-)	75.53(23.9)(-)	35.93(15.26)(=)	47.5(22.8)
Ugdb8	62.33(19.67)(-)	68.07(20.12)(-)	38.73(25.35)(=)	44.0(18.1)
Ugdb23	81.33(28.09)(-)	75.47(31.23)(-)	33.0(17.41)(+)	50.1(18.4)
Uval9D	80.73(30.09)(-)	66.67(16.94)(-)	45.8(18.37)(=)	49.2(18.8)
Uval10D	67.8(20.45)(-)	78.33(26.89)(-)	35.93(15.84)(+)	51.8(22.8)

TABLE V: The mean and standard deviation for **HV** of the compared algorithms in test process.

Instance	TSMOGP	TSFSMOGP
Ugdb1	0.49(0.2)(=)	0.59(0.11)
Ugdb2	0.4(0.18)(-)	0.53(0.17)
Ugdb8	0.48(0.26)(=)	0.56(0.17)
Ugdb23	0.51(0.21)(=)	0.51(0.13)
Uval9D	0.38(0.18)(-)	0.76(0.13)
Uval10D	0.44(0.19)(-)	0.67(0.13)

TABLE VI: The mean and standard deviation for **IGD** of the compared algorithms in test process.

Instance	TSMOGP	TSFSMOGP
Ugdb1	0.51(0.12)(=)	0.48(0.21)
Ugdb2	0.61(0.15)(-)	0.48(0.18)
Ugdb8	0.77(0.16)(-)	0.32(0.17)
Ugdb23	0.68(0.16)(-)	0.44(0.15)
Uval9D	0.65(0.1)(-)	0.46(0.11)
Uval10D	0.66(0.1)(-)	0.48(0.17)

C. Further Analysis

1) Multi-Objective Indicator: Although we only extract the individual with the best effectiveness from the final Pareto Front from TSFSMOGP and TSMOGP for comparisons, we still want to keep the final Pareto Front for users so that they can select routing policies based on their performance. Note that both indicator are calculated based on three objectives, i.e. number of distinguished features, tree size and test performance. In this case, we compare TSFSMOGP with TSMOGP on two multi-objective indicators, i.e. Hyper Volume (HV) and Inverted Generational Distance (IGD). The Wilcoxon rank sum test with a significance level of 0.05 is used to verify the performance of the proposed approaches. For each $+$, $-$ and $=$ in parentheses refer the Wilcoxon rank sum test significance is significantly lower (better) than, higher (worse) than, and comparable with TSFSMOGP.

From Tables V and VI, we can see that TSFSMOGP outperforms TSMOGP on 3 out of 6 instances and perform comparable with TSMOGP on other instances. TSFSMOGP can also outperform TSMOGP on almost all instances but Ugdb1. This is consistent with the results in Tables II - IV. TSFSMOGP can obtain better effectiveness and few distinguished features and slightly larger tree size than TSMOGP. This is as expected, TSFSMOGP considers all three objectives, and TSMOGP only consider tree size and effectiveness.

2) Semantic Analysis of Evolved Policies: To investigate the complexity and interpretability of the GP-evolved routing policies from TSFSMOGP, we first simplify the rules according to some algebraic simplification rules. Then, a representative routing policy is selected. Eqs. (12) – (15) show a selected policy evolved by TSFSMOGP for the Ugdb23 instance. The policy has a promising test performance (248.54, while the mean of TSFSMOGP is 248.83). The size of the routing policy is much smaller than the baseline GPHH. Besides, there are only 7 out of 15 features in this routing policy.

$$RP = \frac{S_1}{S_2} + S_3 \tag{12}$$

where

$$S_1 = \min(\max(CTT1, FUT), DEM) \tag{13}$$

$$S_2 = 2CTD + \max(SC, CTD) \tag{14}$$

$$S_3 = \max(FRT, CFH) \tag{15}$$

We can identify the following patterns and interpretations from the above routing policy.

- For S_1, it can return one of the three terminals, i.e. $CTT1$, FUT and DEM. There are four possible cases for S_1:
  - $CTT1 < FUT$. This indicates that the candidate task is next to the current location which means that the cost from the current location to its nearest remaining task is 0 (this often happens in the early stage of the service when there are many unassigned tasks and they are next to each other).
    - $CTT1 > DEM$. This indicates that the cost from the current location to its nearest remaining task is greater than the demand of the candidate task. However, in this case, $CTT1$ is not able to greater than $DEM1$ when $CTT1$ = 0.
    - $CTT1 < DEM$. This indicates that the demand of the candidate task is greater than the cost from the current location to its nearest remaining task which is 0. In this case, S_1 returns 0. Then, RP becomes S_3.
  - $CTT1 > FUT$. This indicates that there is no remaining task that is next to the candidate task and there are few unassigned tasks (this often happens in the final stage of the service).
    - $CTT1 > DEM$. This indicates that the cost from the current location to its nearest remaining task is greater than the demand of the candidate task. In this case, RP becomes $\frac{CTT1}{S_2} + S_3$. RP prefers the candidate task that have remaining task near it.
    - $CTT1 < DEM$. This indicates that the demand of the candidate task is greater than the cost from the current location to its nearest remaining task. In this case, RP becomes $\frac{DEM}{S_2} + S_3$, RP prefers the candidate task requires less demand.
- For S_2. As S_2 is the denominator (the larger the denominator, the smaller the fraction), S_2 prefers the candidate task farther from the depot. In addition, S_2 has two possible cases:
  - $SC > CTD$. This indicates that the serving cost of the candidate task is greater than the cost from the candidate task to the depot. In this case, RP prefers the candidate task requires larger serving cost.
  - $SC < CTD$. This indicates that the serving cost of the candidate task is less than the cost from the candidate task to the depot. In this case, RP prefers the candidate task that farther from the depot.
- For S_3, it also has two possible cases:
  - $FRT > CFH$. This can happens when there are a number of remaining tasks and the cost from current location to the candidate task is 0 (the candidate task is next to the current location). In this case, state information FRT is taken account.
  - $FRT < CFH$. Usually, CFH is greater than FRT (range from 0-1). This indicates that the candidate task is not next to the current location. Then, CFH is more important. RP will definitely prefers the candidate task that close to the current location.

V. Conclusions and Future Work

This paper proposed a Two Stage Multi-Objective Genetic Programming Hyper Heuristic approach with Feature Selection (TSFSMOGP) for UCARP. The proposed algorithm manages to improve the interpretability of the evolved routing policies from two different aspects simultaneously, i.e. the number of distinguished features and tree size. In the first stage, niching technique and feature selection technique were utilised to select informative features which can be used in the second stage. In the second stage, the multi-objective method, which uses effectiveness and tree size as two main objectives, was applied to reduce the tree size of evolved routing policies. The experimental results showed that TSFSMOGP can evolve effective, compact (fewer distinguished features and smaller program size), and thus potentially interpretable routing policies.

In the future, we will consider to apply GP simplification method to simplify the evolved routing policies. In addition, we will try to add number of distinguished features as an extra objective in the multi-objective optimisation.

References

[1] M. Dror, *Arc routing: theory, solutions and applications*. Springer Science & Business Media, 2012.

[2] H. Handa, L. Chapman, and X. Yao, "Robust route optimization for gritting/salting trucks: a cercia experience," *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 6–9, 2006.

[3] ——, "Dynamic salting route optimisation using evolutionary computation," in *IEEE Congress on Evolutionary Computation*, 2005, pp. 158–165.

[4] S. 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.

[5] S. Wøhlk, "A decade of capacitated arc routing," in *The vehicle routing problem: latest advances and new challenges*. Springer, 2008, pp. 29–48.

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

[7] B. L. Golden and R. T. Wong, "Capacitated arc routing problems," *Networks*, vol. 11, no. 3, pp. 305–315, 1981.

[8] P. Lacomme, C. Prins, and W. Ramdane-Cherif, "Competitive memetic algorithms for arc routing problems," *Annals of Operations Research*, vol. 131, no. 1-4, pp. 159–185, 2004.

[9] U. Ritzinger, J. Puchinger, and R. F. Hartl, "A survey on dynamic and stochastic vehicle routing problems," *International Journal of Production Research*, vol. 54, no. 1, pp. 215–231, 2016.

[10] J. Jacobsen-Grocott, Y. Mei, G. Chen, and M. Zhang, "Evolving heuristics for dynamic vehicle routing with time windows using genetic programming," in *IEEE Congress on Evolutionary Computation*. IEEE, 2017, pp. 1948–1955.

[11] Y. Liu, Y. Mei, M. Zhang, and Z. 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.

[12] J. MacLachlan, Y. Mei, J. Branke, and M. Zhang, "An improved genetic programming hyper-heuristic for the uncertain capacitated arc routing problem," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2018, pp. 432–444.

[13] 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*, ser. GECCO '18. New York, NY, USA: ACM, 2018, pp. 141–142. [Online]. Available: http://doi.acm.org/10.1145/3205651.3205661

[14] Y. Liu, Y. Mei, M. Zhang, and Z. Zhang, "A predictive-reactive approach with genetic programming and cooperative co-evolution for uncertain capacitated arc routing problem," *Evolutionary Computation*, 2019.

[15] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 2020.

[16] 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*, 2019, pp. 1093–1101.

[17] S. Wang, Y. Mei, J. Park, and M. Zhang, "Evolving ensembles of routing policies using genetic programming for uncertain capacitated arc routing problem," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2019, pp. 1628–1635.

[18] ——, "A two-stage genetic programming hyper-heuristic for uncertain capacitated arc routing problem," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2019, pp. 1606–1613.

[19] M. Narayanan, E. Chen, J. He, B. Kim, S. Gershman, and F. Doshi-Velez, "How do humans understand explanations from machine learning systems? an evaluation of the human-interpretability of explanation," *arXiv preprint arXiv:1802.00682*, 2018.

[20] F. Poursabzi-Sangdeh, D. G. Goldstein, J. M. Hofman, J. W. Vaughan, and H. Wallach, "Manipulating and measuring model interpretability," *arXiv preprint arXiv:1802.07810*, 2018.

[21] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, 2015.

[22] A. Friedlander, K. Neshatian, and M. Zhang, "Meta-learning and feature ranking using genetic programming for classification: Variable terminal weighting," in *2011 IEEE Congress of Evolutionary Computation (CEC)*. IEEE, 2011, pp. 941–948.

[23] S. Ok, K. Miyashita, and S. Nishihara, "Improving performance of gp by adaptive terminal selection," in *Pacific Rim International Conference on Artificial Intelligence*. Springer, 2000, pp. 435–445.

[24] Y. Mei, M. Zhang, and S. Nyugen, "Feature selection in evolving job shop dispatching rules with genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 2016, pp. 365–372.

[25] 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.

[26] R. W. Eglese and L. Y. Li, "A tabu search based heuristic for arc routing with a capacity constraint and time deadline," in *Meta-Heuristics*. Springer, 1996, pp. 633–649.

[27] A. Hertz, G. Laporte, and M. Mittaz, "A tabu search heuristic for the capacitated arc routing problem," *Operations research*, vol. 48, no. 1, pp. 129–135, 2000.

[28] P. Lacomme, C. Prins, and W. Ramdane-Chérif, "A genetic algorithm for the capacitated arc routing problem and its extensions," in *Workshops on Applications of Evolutionary Computation*. Springer, 2001, pp. 473–483.

[29] K. Tang, Y. Mei, and X. Yao, "Memetic algorithm with extended neighborhood search for capacitated arc routing problems," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1151–1166, 2009.

[30] Y. Mei, K. Tang, and X. Yao, "Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, pp. 151–165, 2011.

[31] P. Lacomme, C. Prins, and A. Tanguy, "First competitive ant colony scheme for the carp," in *International Workshop on Ant Colony Optimization and Swarm Intelligence*. Springer, 2004, pp. 426–427.

[32] K. F. Doerner, R. F. Hartl, V. Maniezzo, and M. Reimann, "Applying ant colony optimization to the capacitated arc routing problem," in *International Workshop on Ant Colony Optimization and Swarm Intelligence*. Springer, 2004, pp. 420–421.

[33] G. Fleury, P. Lacomme, and C. Prins, "Evolutionary algorithms for stochastic arc routing problems," in *Workshops on Applications of Evolutionary Computation*. Springer, 2004, pp. 501–512.

[34] J. Wang, K. Tang, and X. Yao, "A memetic algorithm for uncertain capacitated arc routing problems," in *Memetic Computing (MC), 2013 IEEE Workshop on*. IEEE, 2013, pp. 72–79.

[35] 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.

[36] T. Weise, A. Devert, and K. Tang, "A developmental solution to (dynamic) capacitated arc routing problems using genetic programming," in *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. ACM, 2012, pp. 831–838.

[37] M. A. Ardeh, Y. Mei, and M. Zhang, "A novel genetic programming algorithm with knowledge transfer for uncertain capacitated arc routing problem," in *Pacific Rim International Conference on Artificial Intelligence*. Springer, 2019, pp. 196–200.

[38] ——, "Genetic programming hyper-heuristic with knowledge transfer for uncertain capacitated arc routing problem," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2019, pp. 334–335.

[39] Y. Liu, Y. Mei, M. Zhang, and Z. Zhang, "A predictive-reactive approach with genetic programming and cooperative coevolution for the uncertain capacitated arc routing problem," *Evolutionary computation*, vol. 28, no. 2, pp. 289–316, 2020.

[40] A. K. Uysal, "An improved global feature selection scheme for text classification," *Expert systems with Applications*, vol. 43, pp. 82–92, 2016.

[41] A. Lensen, B. Xue, and M. Zhang, "Particle swarm optimisation representations for simultaneous clustering and feature selection," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2016, pp. 1–8.

[42] Q. Chen, M. Zhang, and B. Xue, "Feature selection to improve generalization of genetic programming for high-dimensional symbolic regression," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 792–806, 2017.

[43] T. Hildebrandt, J. Heger, and B. Scholz-Reiter, "Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach," in *Proceedings of Genetic and Evolutionary Computation Conference*. ACM, 2010, pp. 257–264.

[44] A. Pétrowski, "A clearing procedure as a niching method for genetic algorithms," in *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*. IEEE, 1996, pp. 798–803.

[45] S. Luke, L. Panait, G. Balan, S. Paus, Z. Skolicki, J. Bassett, R. Hubley, and A. Chircop, "Ecj: A java-based evolutionary computation research system," *Downloadable versions and documentation can be found at the following url: http://cs. gmu. edu/eclab/projects/ecj*, 2006.