Genetic Programming Hyper-heuristic with Cluster Awareness for Stochastic Team Orienteering Problem with Time Windows

Jericho Jackson, Yi Mei
School of Engineering and Computer Science
Victoria University of Wellington)
Kelburn 6011, New Zealand
Email: {jacksojeri, yi.mei}@ecs.vuw.ac.nz

***Abstract*—This paper looks at the stochastic Team Orienteering Problem with Time Windows, a well-known problem that models the Personalised Tourist Trip Design Probelm. Due to the nature of randomness such as real-time delays, the traditional optimisation approaches are not effective in solving the stochastic problem variant. In this case, genetic programming hyper-heuristics (GPHH) are promising techniques for automatically learning heuristics to make real-time decisions to effectively handle the stochastic environment, however, they still have limitations as the decision making policies use short-sighted information. In this paper, we propose to incorporate global information into the GPHH solution, with a constructed terminal feature based on cluster information to be used by the GPHH, as well as a clustering-aware solution generation process. The experimental studies showed that the newly designed cluster-based feature gave an improvement over the standard GPHH solution. This suggests that incorporating cluster information can be beneficial. Although the clustering-aware solution generation process did not achieve satisfactory performance, the further analysis showed that it could lead to improved performance under certain condition. Overall we demonstrate the effectiveness of using clustering as a global information to enhance the performance of GPHH.**

I. Introduction

The stochastic Team Orienteering Problem with Time Windows (TOPTW) [1] is a difficult optimisation problem that can model many real-world scenarios. One such scenario, the tourist trip design problem [2], asks for a set of itineraries of Places-of-Interest (POIs) for a tourist to visit on their trip that maximises the tourist's satisfaction, under various time, travel and budget constraints. The stochastic TOPTW provides a comprehensive model of the tourist trip design. First, it allows for multiple itineraries to be produced on a problem instance to represent a tourist's multi-day trip to a city. Second, it considers the time of visits to POIs to allow for their opening and closing times to be factored into the itinerary generation. Last, it allows for stochastic visit durations, which represents that tourists can stay longer or shorter at a POI than anticipated.

A solution to the stochastic TOPTW that can efficiently find a near-optimal set of routes would represent significant value to trip planning services, as well as many services in other contexts that the stochastic TOPTW can be applied to.

The TOPTW and related problems have been studied extensively in the past, such as in [1], [3], [4], [5], [6], [7], [8], [9], [10]. Many techniques used in these studies do not apply well to the stochastic TOPTW variant as they develop solutions that can be inflexible to variance in the stochastic variables of the problem [1]. Genetic Programming Hyper-Heuristic (GPHH) evolves a decision making policy that can react to real-time events to produce effective solutions to the stochastic TOPTW. Genetic Programming Hyper-Heuristic (GPHH) has been successfully applied to the stochastic TOPTW [1], as well as in other dynamic scheduling and routing problems [11], [12], [13], [14]. Previous GPHH approaches to the stochastic TOPTW are limited due to the lack of considering the global POI clustering information, which can lead to policies being developed that favour a single high scoring POI over a lower scoring POI that is surrounded by other POIs which can subsequently be visited for minimal travel cost.

In this paper, we propose a solution to the stochastic TOPTW using a combination of GPHH, that has previously been used [1], with global information on POI clustering. To the best of our knowledge, this paper is the first to use a GPHH solution for the stochastic TOPTW that incorporates clustering techniques. This paper proposes a GPHH algorithm involving cluster aware heuristics that produces effective real-time decision making policies for the stochastic TOPTW. This is composed of the following specific objectives.

- Design a new terminal feature that includes clustering information to be used by the GPHH to evolve a policy. This will include constructing features that represent the static or dynamic value of the neighbourhood that a POI is in using clustering algorithms.
- Design the schedule creation procedure, i.e. the algorithm that generates a set of routes, given a stochastic TOPTW instance and a policy. We incorporate the POI clustering information into the schedule creation procedure to generate routes that can recognise the value of dense clusters of POIs in relation to isolated high-value POIs.
- Analyse the effectiveness of GPHH with clustering techniques. The policy and schedule creation procedure will

be evaluated through comparison with a benchmark schedule creation procedure and policy generated by GPHH with no cluster-based features.

The remainder of this paper is structured as follows. Section II gives the problem description and literature review. Section III will outline the proposed GPHH solution and clustering algorithms. Section IV includes the experimental results and analysis. Section V contains the conclusions and future work.

II. Background

A. Problem description

In TOPTW, a set of n POIs $\mathbf{P} = \{1, \dots, n\}$ exist. Each POI $p \in \mathbf{P}$ has a score obtained when it is visited $s(p)$, a duration of visit $d(p)$ and an availability time window $[o(p), c(p)]$. The starting point $p_s \in \mathbf{P}$ and ending point $p_e \in \mathbf{P}$ are also given. For any pair of POIs (p_i, p_j), the travel time from p_i to p_j is $t(p_i, p_j)$. A trip is comprised of m days (tours). Each day has a starting time T_s and an ending time T_e. The problem requires a set of sequences of POIs to be designed so that the total score of the visited POIs is maximised, and the following constraints are met.

1) There are m sequences of POIs, one for each day.
2) Each sequence departs from the starting point p_s at time T_s, and end at p_e no later than T_e.
3) Each POI is visited once at most.
4) A visit at p_i can not start earlier than o_i or later than c_i. If the tourist arrives at p_i earlier than o_i, they have to wait until o_i to begin the visit.

A more detailed mathematical formulation can be seen in [1].

In the stochastic TOPTW, the visit duration $d(p)$ is a random variable rather than a deterministic value. Consequently, the objective becomes maximising the expected total score of visited POIs upon all possible environments.

B. Related Work

The TOPTW and its related variants have been the focus of extensive research in the past. Variations of the TOPTW have been approached using a variety of solutions, including tabu search [3], [15], GPHH [1], simulated annealing [16], iterated local search [4], multitasking GPHH [5], particle swarm optimisation [6], cutting planes [7], artificial bee colony [8], [17], ant colony optimisation [18], [19] and cluster-based heuristics [10].

The iterated local search algorithm [4], [9] is one of the best-known solutions to the TOPTW. It uses manually designed heuristics to construct an initial set of routes and then iteratively improve them by exploring the local search space with partial adjustments to the routes. Whilst this approach has been shown to efficiently generate near-optimal routes for the TOPTW, it is not directly applicable to the stochastic TOPTW, as the algorithm needs to generate a full set of routes with known POI visit times before it can begin its iterative local search. Many studies examining deterministic orienteering problems present the same issue of having solutions that are not directly applicable to stochastic problems. The stochastic TOPTW requires a solution that can be robust to changes in the stochastic environment. Stochastic orienteering problems have been examined in [1], [20], [21], [22], [23]. With the exception of [1], these studies focused on finding robust solutions to the possible situations that could manifest in a stochastic environment, rather than generating solutions in reaction to real-time environment changes.

Dispatching rules have been demonstrated to be effective real-time decision makers for dynamic scheduling and routing problems [24], [25], so long as an effective heuristic is used. Manually designing optimal heuristics to be used as dispatching rules can be difficult, non-intuitive and/or require knowledge of a specific context. As a response to this, GPHH has been applied to evolve effective heuristics that can be used as dispatching rules for these problems [1], [5]. In the Stochastic TOPTW, a dispatching rule evolved by GPHH will evaluate each feasible POI to be visited next and add the POI with the highest priority to the route.

GPHH has previously been successfully applied to the stochastic TOPTW [1], where the terminals used by the GPHH were mostly features representing local information on a candidate POI, such as the time to travel to the POI or expected visit duration at the POI. It investigated the use of two look-ahead features to consider the next POI to be visited, and concluded that including the look-ahead features could improve the effectiveness of the policy developed by GPHH, however there is further potential to incorporate look-ahead information in more effective ways. One reason that the policy produced in [1] may not produce the optimal set of routes is that it mostly considers POIs in isolation. This means that the policy could choose to visit two isolated POIs a certain distance north of the start point with high satisfaction scores, whilst ignoring a larger group a similar distance to the south with slightly lower scores. This could not be optimal, as routes to the larger group could allow more nodes to be visited, giving a greater total satisfaction score. Situations such as this represent the limitations of previous GPHH solutions to orienteering problems and show potential for developing solutions that can incorporate some awareness of the groupings of POIs into its decision making.

The two algorithms presented in [10] are solutions to the TOPTW that incorporate knowledge of clusters of POIs into solution generation. These algorithms first cluster the POIs in a city into groups based on topological distance, and then favour producing routes that visit all POIs in a cluster before moving to the next. One of the algorithms presented in the paper achieved higher quality solutions than an iterative local search algorithm (one of the best-known solutions). This demonstrates the potential of TOPTW solutions including information of the clustering of POIs into schedule creation process. A limitation of this solution is that the cluster-based heuristics used to generate routes were manually designed, and may not be optimal, despite being effective. This highlights a potential for GPHH to evolve a heuristic that includes knowledge of clusters in some way to produce an improved solution to the TOPTW.

III. Proposed GPHH with Cluster Awareness

This section covers the proposed GPHH and clustering framework for the stochastic TOPTW. We propose to develop a cluster-related feature to be included in the terminal set of GPHH, and implement a clustering algorithm in the schedule creation procedure that divides the POIs into clusters and make decisions at the level of clusters rather than individual POIs.

A. Neighbourhood Score Feature

The neighbourhood score (NS) feature is an addition to the terminal features used in [1], and is a constructed feature intended to represent the value a POI has based on its location in relation to other POIs. The intention is that a POI within a cluster of other POIs will have a higher NS than a POI that is, by comparison, isolated. Algorithm 1 shows how the NS of a POI is calculated.

Algorithm 1: Calculation of Neighbourhood Score

```
   Input: A TOPTW instance $I$, a POI $p$ to be evaluated.
   Output: A NS value $ns$ for $p$.
1  $ns \leftarrow 0$;
2  $feasiblePOIs \leftarrow \texttt{getFeasiblePOI}(p)$;
3  for $p' \in feasiblePOIs$ do
4      $ns \leftarrow ns + s(p')/t(p, p')$;
5  end
```

To calculate the NS of a POI p, all the *feasible* POIs that could be immediately visited following p are considered. Each of these contributes to the NS with the satisfaction score gained from a visit to this POI, divided by the time it would take to travel from p to the neighbour. The closer a neighbour is to p and the greater the satisfaction from visiting the neighbour, the higher it will raise the NS of p. This means that any POI closely surrounded by other POIs with high satisfaction scores will have a large NS, and the inclusion of this feature in the terminal set of the GPHH could allow for policies to be developed that schedule tours to high value groups of POIs in favour of isolated high value POIs.

In line 2, a feasible POI p' can be visited following POI p if the following constraints are met:

- p' is unvisited and $p' \neq p$.
- After visiting p, p' can be arrived at before its closing time, i.e. $\text{currTime} + \text{TFV}(p) + t(p, p') \leq c(p')$.
- After visiting p and p', the trip can return to the end point by T_e, i.e. $\max\{\text{currTime} + \text{TFV}(p) + t(p, p'), o(p')\} + d(p') + t(p', p_e) \leq T_e$.

B. Cluster Aware Solution Generation

The schedule creation procedure is described in Algorithm 2. It uses the policy to iteratively add POIs to the schedule, ending each day when no more can be visited. The step of using the clustering information is in line 5.

The new step is used at the beginning of each day. A clustering algorithm is used to group the POIs into clusters. Then the cluster with the highest total score is identified, and the POI with the highest score within that cluster is added as the first visit of the tour. The intention of this is to encourage the schedule creation procedure to start tours with visits to profitable of clusters, as it will allow the full time budget to be spent visiting them with minimal travel time between them. This could be superior to solely using the generated policy to create the schedule as it does not consider the clustering of POIs and could lead to visits to isolated POIs that cause the time budget to be spent on travel time.

The clustering algorithm in the schedule creation procedure will not be used in conjunction with the NS feature, it is simply an alternative method of incorporating POI cluster information as a heuristic. When the schedule creation procedure using the clustering algorithm is experimented on, the NS feature will not be used, and inversely, when the NS feature is experimented on, the schedule creation procedure will not have the clustering step included.

Algorithm 2: Schedule Creation Procedure

```
    Input: A TOPTW instance $I$, a policy $pol$.
    Output: A feasible solution $X = \{X_1, X_2, \ldots, X_m\}$.
1   $day \leftarrow 0$, $t \leftarrow T_s$, $p_c \leftarrow p_s$, $X_{day} \leftarrow (p_s)$;
2   $\Omega \leftarrow \mathbf{P} \setminus \{p_s, p_e\}$, $queue \leftarrow \{(p_s, t)\}$;
3   while $queue \neq \emptyset$ do
4       if $t = T_s$ then
5           Obtain the next $poi$ using Algorithm 3;
6           $X_{day} \leftarrow poi$, $t \leftarrow t_{\text{start}} + d_{\text{sample}}(poi)$;
7       else
            // trigger the next event
8           $(p^*, t^*) \leftarrow \text{poll}(queue)$;
9           Update the feasible unvisited POIs $\Omega' \subseteq \Omega$;
10          if $\Omega' = \emptyset$ then
11              $X_{day} \leftarrow (X_{day}, p_e)$;          // return to $p_e$
12              $day \leftarrow day + 1$;    // go to the next day
13              if $day = m$ then Return $X$;
14              $X_{day} \leftarrow (p_s)$;           // open a new tour
15              $queue \leftarrow queue \cup (p_e, T_s)$;
16          end
17          Calculate the priority value $pol(p)$ of each $p \in \Omega'$;
18          $p_{\text{next}} \leftarrow \arg\max_{p \in \Omega'} pol(p)$;
19          $\Omega \leftarrow \Omega \setminus p_{\text{next}}$;
20          $t_{\text{arr}} \leftarrow t + t(p^*, p_{\text{next}})$;            // arrival time
21          $t_{\text{start}} \leftarrow \max\{t_{\text{arr}}, o(p_{\text{next}})\}$; // visit start time
22          if $t_{start} > c(p_{next})$ then
23              $queue \leftarrow queue \cup (p_{\text{next}}, t_{\text{start}})$;
24          else
25              sample the actual duration $d_{\text{sample}}(p_{\text{next}})$;
26              $t \leftarrow t_{\text{start}} + d_{\text{sample}}(p_{\text{next}})$;
27              $X_{day} \leftarrow (X_{day}, p_{\text{next}})$;
28              $queue \leftarrow queue \cup (p_{\text{next}}, t)$;
29          end
30      end
31  end
```

In this paper we implemeted the well-known DBSCAN algorithm [26] to perform the clustering step in the schedule creation procedure.

The DBSCAN algorithm, described in Algorithm 3, creates clusters by iteratively selecting a POI out of the set of unvisited POIs, adding it to a cluster, and expanding the cluster by adding neighbour POIs that are within ϵ distance from the cluster, until it cannot be expanded further. This algorithm was implemented as it can cluster the POIs in the instance without needing to know how many clusters it should find, unlike other algorithms such as K-means clustering. This is important as different TOPTW problem instances could have a range in the number of sensible clusters present, and the DBSCAN algorithm suits this as it builds a cluster from a single POI based on the proximity of other POIs and repeats until no more clusters are found.

The DBSCAN algorithm allows a maximum seperation ϵ between neighbour POIs in the same cluster to be specified, meaning the clusters it finds will be continuous shapes. The weakness of DBSCAN is that it will continue to expand a cluster until it finds no more POIs within ϵ distance to add, meaning the clusters can grow larger than necessary for the TOPTW problem, as only a certain number of POIs can be visited. If the size is limited, it is still vulnerable to finding nonoptimal cluster shapes, for example, a long line of POIs could be identified as a cluster as long as each is within ϵ distance of the next.

The DBSCAN algorithms performance is dependent on the parameter ϵ, which is used to evaluate whether two POIs are close enough to be within the same cluster. In this paper, ϵ was assigned the value of the average travel time between any two POIs in the problem, divided by 50. This value means that two POIs should generally be much closer than any random two POIs if they are to be included in the same cluster, and some manual testing showed that using a value of 50 found clusters of appropriate size. Ideally, the ϵ value would be extracted from the features of a problem instance in an intelligent way, but this is a problem that would need research of its own, if the DBSCAN algorithm proves valuable.

Algorithm 3: DBSCAN Algorithm

```
   Input: A TOPTW instance $I$
1  Output: A POI $poi$ to be added to schedule, $\epsilon$
2  $clusters \leftarrow \{\}$, $list \leftarrow \mathbf{P} \setminus \{p_s, p_e\}$;
3  while $list \neq \{\}$ do
4      $startPoint \leftarrow list(0)$;
5      Remove $startPoint$ from $list$;
6      $fringe \leftarrow \{startPoint\}$;
7      $cluster \leftarrow \{startPoint\}$;
8      while $fringe \neq \{\}$ do
9          $p \leftarrow poll(fringe)$;
10         for $p' \in list$ do
11             if $t(p, p') < \epsilon$ then
12                 Add $p'$ into $fringe$ and $cluster$;
13                 Remove $p'$ from $list$;
14             end
15         end
16     end
17     Add $cluster$ into $clusters$;
18 end
19 $bestCluster \leftarrow$ the cluster with highest total score;
20 $p^* \leftarrow$ the POI with the highest score in $bestCluster$;
21 Return $p^*$;
```

C. Overall Framework

The overall GPHH evolutionary algorithm used in this paper is the same as in [1], and is shown in Algorithm 4. For each generation, the offspring population is generated using three genetic operators: crossover, mutation and reproduction. Additionally, some elites are inherited from the previous generation's population to improve convergence.

The fitness function is defined as the average total score of the solutions produced by the evaluated individual using Algorithm 2. It is calculated as:

$$\text{fit}(pol) = \frac{1}{|\mathcal{I}_{\text{train}}|} \sum_{I \in \mathcal{I}_{\text{train}}} \text{score}(\text{solution}(I, pol)). \qquad (1)$$

Algorithm 4: Pseudo code of GPHH for stochastic TOPTW

```
1  Initialise a population $pop$ of policies by ramp-half-and-half;
2  while stopping criteria is not met do
3      Sample an instance by sampling a value for all the random
         variables;
4      Evaluate individuals in $pop$;
5      Breed a new population $pop'$ by elitism and genetic
         operators;
6      $pop \leftarrow pop'$;
7  end
8  Return best individual in $pop$;
```

IV. Experimental Studies

A. Experiment Settings

The commonly used c*_100, r*_100 and rc*_100 [1], [27], c*_200, r*_200 and rc*_200 [1], [28] and pr* [1], [27], [28] are used in our experimental studies. The number of POIs ranges from about 50 to almost 300. We selected 5 instances from each dataset (e.g. c101∼c105) for the experiments. We use the same uncertainty level of 20% as in [1].

The population size is set to 1024, and the maximal number of generations is 51. The maximal tree depth is 8. The crossover/mutation/reproduction rates are 0.8/0.15/0.05. The best 10 individuals are selected for elitism. The algorithm was implemented in Java with the ECJ library [29].Each algorithm was run 30 times independently.

TABLE I
The features to be used in the terminal set of GPHH.

Notation	Description	Calculation
SCORE	score of the POI	$s(p)$
DUR	duration of the visit	$E(d(p))$
TO	time to the opening time	$o(p) - \text{currTime}$
TC	time to the closing time	$c(p) - \text{currTime}$
TA	time to arrive the POI	$t(\text{currPlace}, p)$
TR	time to return to the end point	$t(p, p_e)$
TSV	time to start the visit	$\max\{\text{TO}, \text{TA}\}$
TFV	time to finish the visit	$\text{TSV} + E(d(p))$
SL	slack	$\text{TC} - \text{TA}$
RemT	remaining time budget	$\text{remainDays} \cdot (T_e - T_s) + (T_e - \text{currTime})$
NS	neighbourhood score of POI	*see Algorithm 1*

For each Stochastic TOPTW instance, all the algorithms will be evaluated on a test set with 500 randomly sampled instances. The test performance of an algorithm is calculated as the average total score of the solutions generated for test instances, where $|\mathcal{I}_{\text{test}}| = 500$.

The features in the terminal set of GPHH are shown in Table I. The function set is comprised of $\{+, -, \times, /, \min, \max\}$. Each function takes two arguments. The "/" operator is protected, and returns 1 if the divisor is 0.

B. Results and Discussions

TABLE II
THE TEST PERFORMANCE OF THE COMPARED ALGORITHMS ON THE INSTANCES WITH $m = 3$.

Inst	BasicGP	NS-GP	DBSCAN
c101	759.6(9.6)	760.7(7.5)	623.9(44.6)(-)
c102	854.9(8.6)	857.7(8.7)	778.7(20.5)(-)
c103	908.8(6.7)	912.3(2.0)(+)	810.3(40.7)(-)
c104	954.2(13.5)	957.9(10.2)	932.8(7.0)(-)
c105	805.4(9.5)	812.6(2.7)(+)	639.6(57.3)(-)
r101	434.5(11.8)	432.3(14.2)	325.4(8.9)(-)
r102	603.1(15.5)	608.3(11.4)	495.2(12.7)(-)
r103	656.2(11.4)	655.8(12.2)	544.4(20.9)(-)
r104	711.4(15.7)	713.2(7.8)	613.5(20.3)(-)
r105	546.6(16.8)	547.7(16.6)	445.8(10.4)(-)
rc101	562.8(12.5)	562.2(14.3)	430.6(26.8)(-)
rc102	620.7(13.5)	621.5(12.7)	501.9(26.0)(-)
rc103	667.1(20.1)	669.0(14.2)	550.3(29.5)(-)
rc104	721.0(14.3)	723.2(15.0)	725.9(14.4)(+)
rc105	611.3(13.2)	616.3(11.5)	471.1(38.1)(-)
c201	1765.9(5.0)	1770.0(3.1)(+)	1584.8(18.2)(-)
c202	1774.2(5.7)	1772.9(7.6)	1741.0(33.3)(-)
c203	1767.3(5.8)	1767.1(13.1)	1737.3(21.8)(-)
c204	1779.5(7.5)	1779.6(5.2)	1756.1(47.5)(-)
c205	1783.0(3.3)	1784.0(4.0)(+)	1682.3(50.2)(-)
r201	1357.8(21.0)	1352.6(18.8)	1129.0(16.3)(-)
r202	1415.5(8.6)	1414.4(8.0)	1202.1(82.2)(-)
r203	1438.3(1.8)	1438.2(3.0)	1405.6(20.3)(-)
r204	1440.2(3.6)	1440.7(1.1)	1430.8(7.4)(-)
r205	1437.1(4.5)	1438.6(2.9)	1297.6(77.6)(-)
rc201	1597.3(17.2)	1596.7(16.6)	1361.4(70.6)(-)
rc202	1656.7(20.0)	1653.6(19.7)	1392.1(117.9)(-)
rc203	1712.1(4.9)	1711.7(NA)	1635.7(67.9)(-)
rc204	1720.2(2.5)	1720.9(0.3)	1682.9(206.4)(-)
rc205	1624.2(22.9)	1629.2(24.5)	1387.5(84.7)(-)
pr01	559.8(12.3)	563.6(13.8)	478.8(17.5)(-)
pr02	864.1(11.0)	860.9(8.8)	729.3(41.1)(-)
pr03	770.3(19.9)	772.8(14.9)	605.2(44.4)(-)
pr04	855.0(4.8)	851.9(11.0)	614.5(50.0)(-)
pr05	908.2(21.4)	915.4(15.5)	560.3(27.4)(-)
pr11	621.1(8.7)	619.3(11.7)	560.0(14.8)(-)
pr12	914.3(10.3)	913.7(11.1)	717.2(32.4)(-)
pr13	889.1(17.0)	885.7(15.9)	681.9(17.7)(-)
pr14	920.7(21.2)	916.2(26.6)	805.8(12.3)(-)
pr15	970.1(23.6)	974.7(17.3)	865.8(84.4)(-)

Table II shows the test fitness of the basic GPHH (BasicGP), GPHH using NS (NS-GP), and GPHH using DBSCAN clustering (DBSCAN) on the instances with $m = 3$. The proposed algorithms are compared with BasicGP using Wilcoxon rank sum test with significance level $\alpha = 0.05$. If a proposed algorithm (NS-GP or DBSCAN) is significantly better, then the corresponding entry is marked with "(+)". If a proposed algorithm is significantly worse, it is marked with "(-)".

From Table II, it can be seen that NS-GP performs similarly to the BasicGP on most instances, and statistically significantly better on 4 instances. This demonstrates there can be an advantage gained by including some type of clustering information into a GPHH framework for the stochastic TOPTW. This result was expected as it provides additional information not provided by BasicGP that can be used by the evolved policy, allowing it to make favourable decisions between a high score isolated POI and a cluster of lower score POIs.

The DBSCAN showed significantly worse results than BasicGP across all test instances, with the exception of rc104 which it is significantly better. The DBSCAN algorithm finds the highest value cluster of POIs in the problem and then schedules a visit to a POI within this cluster as the first visit of a tour. The expectation was that this would allow the schedule to begin in the most valuable geographical region and visit a group of surrounding POIs with minimal travel time between them. The results suggest that there are some factors not accounted for with this logic. It could be the case that in a number of instances, the highest value cluster was quite distant from the starting point, and that it would be better to either visit a closer cluster, or to visit POIs that are between the starting point and the cluster before reaching it. Another scenario could be that the highest value cluster is formed of a large number of low or medium satisfaction score POIs, which cannot all be visited in the time budget, meaning the value of the cluster isn't attainable and it would be better to visit a lower value cluster with higher satisfaction score POIs.

Figure 1 shows the convergence curves of BasicGP, NS-GP and DBSCAN on 8 representative instances (c104, c105, rc105, c203, c204, r202, pr03 and pr14). The plots show little difference in the test fitness convergence of BasicGP and NS-GP, and both appear to converge after around 30 generations, with no overfitting evident. The DBSCAN algorithm has much more variability in test performance that BasicGP and NS-GP, but also appears to converge after about 30 generations.

C. Further Analysis

Figure 2 shows the terminal frequencies of the final policies obtained by 30 runs of NS-GP for c103, c105 and c201, where NS-GP outperformed BasicGP. The figure shows that the NS feature is not one of the features that dominates the policy, however it shows up with a reasonable frequency in comparison to most other terminals. This is expected, as the policy should generally value terminals such as SCORE, DUR and TFV to evaluate the priority of a POI as these immediately affect the solution score and remaining time budget following a decision. The NS feature should provide an incentive to visit a POI with better options for subsequent visits, and act in a way similar to a tie breaker for POIs that have similar values for the dominant terminals. To examine the way the NS feature is being used in evolved policies, we analyse a simplified final policy on instance c205, where NS-GP outperformed BasicGP.

$$\max\{\mathrm{TC} * \min\{\mathrm{TA}, \mathrm{RemT}\} * \max\{\mathrm{NS}, \mathrm{TC}, \mathrm{TFV}\},$$
$$\mathrm{SCORE}, \mathrm{TSV}, \{\mathrm{TC} - \{\mathrm{TSV} + \mathrm{NS}\}\}$$

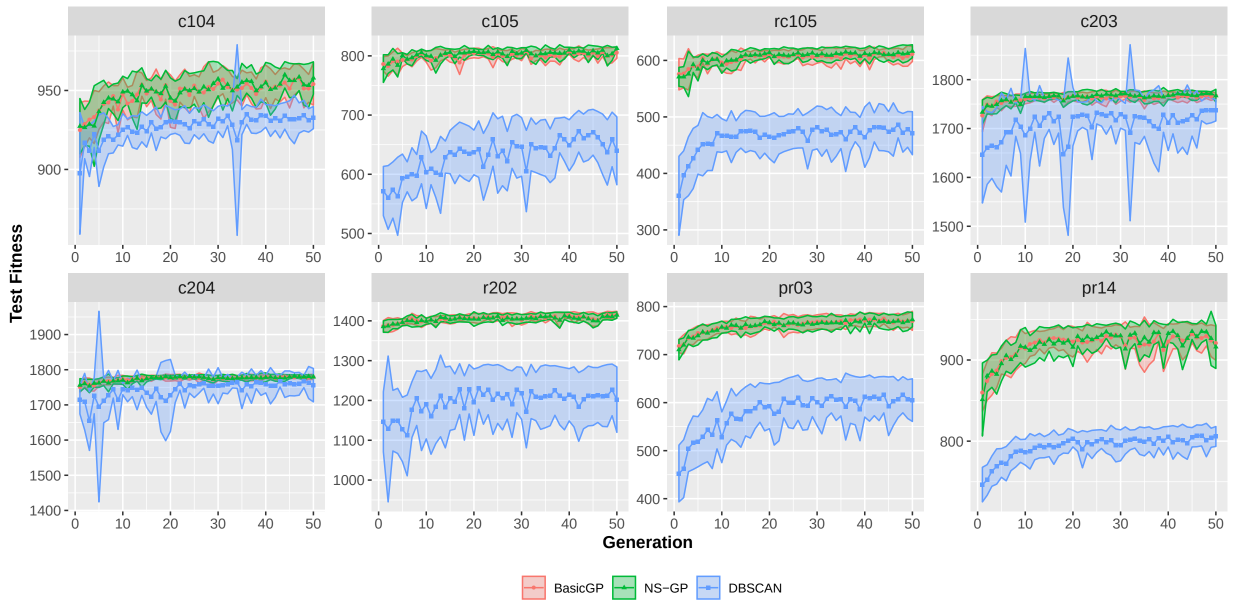

Fig. 1. The convergence curves of test performance of BasicGP, NS-GP and DBSCAN on c104, c105, rc105, c203, c204, r202, pr03 and pr 14, with $m = 3$

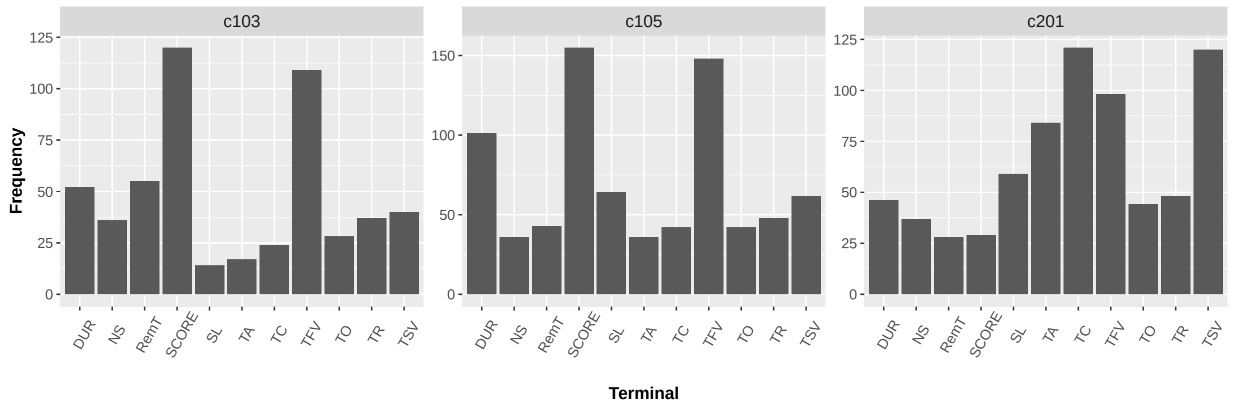

Fig. 2. The terminal frequencies of the final policies obtained by 30 runs of GPHH using the NS feature on c103, c105 and c201, with $m = 3$.

The policy shows that the NS feature is operated on with the TC, TFV and TSV features in a couple of different ways. It is combined with them numerically using addition and subtraction, as well as compared using the *max* operator. TC, TFV and TSV are all features that give a time corresponding to a visit (closing time, finishing time and starting time). This pattern could suggest that the NS feature is being used to prioritise visits to POIs that might end later, if they have a greater NS value. This indicates the feature is acting as intended as in some cases it is smarter to take longer to visit a POI with a greater NS than quickly finish a visit to another POI with fewer subsequent visit options.

To better understand the reason for the underperformance of DBSCAN, we examine the DBSCAN output on pr04 and rc105. The size of a POI in the figure corresponds to its score, where a larger circle indicates a larger score. As seen in Figure 3, the clusters found seem to be the most dense concentration of POIs available. On instance rc105, the cluster is distant from the starting point, meaning that the tour must begin with a long journey to a POI in this cluster which consumes a large amount of the time budget. The cluster found on instance pr04 is relatively close to the start point, but contains many lower-score POIs grouped together, many of which might not be able to be visited in one tour. For this reason, this cluster could be overvalued by the DBSCAN algorithm, as it could be more valuable to visit the group of fewer higher value POIs in the opposite direction. These two instances show the areas which the clustering could be improved, specifically, cluster size should be capped in some way to stop a cluster being overvalued, the duration of a visit should also be considered to avoid situations where a few low score POIs represent a more valuable cluster than a higher score POI despite taking much longer to visit in total, and lastly, the distance of a cluster from the start point should be considered, and ideally POIs on

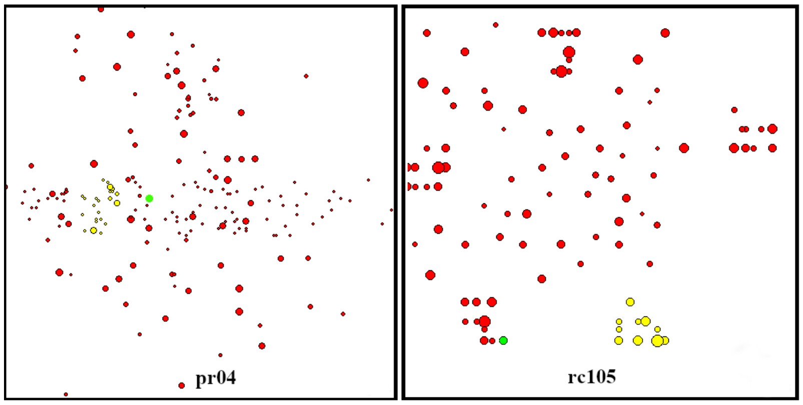

Fig. 3. DBSCAN cluster output on instances pr04 and rc105 (Red - POI, Yellow - Cluster POI, Green - Start point).

the route to the cluster could be visited if they don't require large detours.

There is potential to develop a refined clustering algorithm that does not rely on manually configured parameters, and analyses the information of a problem instance so that clusters of near-optimal size and shape can be found. The schedule creation procedure should also be updated to take possibly advantage of POI visits on route to the best cluster found, rather than ignoring these to visit the cluster first.

DBSCAN outperformed BasicGP and NS-GP on instance rc104. We investigate the solutions generated on this instance by one of the best evolved policies of each of the three algorithms, with $m = 3$. Figure 4 shows that the first cluster found by DBSCAN (yellow) is adjacent to the starting point. This is the ideal scenario for this implementation of DBSCAN to work in, as the dense region of POIs can be reached with very little initial time budget expenditure. The DBSCAN solution visits a number of the medium or high score POIs in this cluster, until these options are exhausted and it then visits medium and high score POIs surrounding the cluster to finish the tour. The second cluster visited (blue) shows fewer POIs. By day three, there are no clusters as valuable as those of day one and two remaining, so the third cluster found (grey) contains only low value POIs and the schedule leaves this cluster immediately after the first visit. This shows the weakness of the DBSCAN implementation, as when no strong cluster exists, it would be better to use the policy than to force a visit to the cluster, as this would avoid the long travel times seen in day three of the DBSCAN solution. The BasicGP and NS-GP solutions appear similar, and both involve visits in the DBSCAN cluster. These solutions both show a slight preference of making a longer trip to visit a slightly higher score POI when given the option, which is shown by them visiting fewer nodes in the prominent cluster than the DBSCAN solution. This could be a result of the evolutionary process, which would require policies favouring SCORE in order for the solutions to schedule in the direction of the main cluster and visit the group of high score POIs which appears to be central to any near-optimal route on this instance. As these algorithms policies must potentially favour SCORE in order to generate solutions that reach the cluster, it is possible that once the schedule reaches the cluster, score is favoured too much by the policy, causing some of the lower score POIs in the cluster to be ignored. The policies evolved in the DBSCAN algorithm are only used for decisions where the current visit is already in the cluster. This could allow for an evolutionary process that develops a rule that prioritises POIs more efficiently and isn't swayed by SCORE as much. This example highlights the possibility for a GPHH framework that evolves multiple rules that are employ by the schedule creation procedure based on an analysis of the real-time context. For example, if the schedule creation procedure was aware the current visit was in a cluster, it could employ a rule that might evolve to value short trips to neighbours, and otherwise, a rule that could evolve to prioritise high scoring POIs.

V. Conclusions and Future Work

In this paper, two method were proposed for incorporating cluster information into the GPHH solution for the Stochastic TOPTW presented in [1]. Specifically, one constructed feature to assign a score to POIs based on the value of their geographical location, and the DBSCAN clustering algorithm implemented into the schedule creation procedure, to direct the tour to high value clusters. The results showed that there is promise in incorporating cluster information into a GPHH solution, as the constructed feature showed marginally improved results over the standard GPHH solution. The DBSCAN clustering algorithm implemented showed worse results than the standard GPHH solution, showing that manually overriding the evolved policy to direct a tour to a specific cluster isn't effective, at least in the way that it was implemented.

In future work, the idea of implementing a clustering algorithm into the schedule creation procedure could be improved on by refining the algorithm. Additionally, new cluster based constructed features for GPHH could be researched.

References

[1] Y. Mei and M. Zhang, "Genetic Programming Hyper-Heuristic for Stochastic Team Orienteering Problem with Time Windows," in *IEEE Congress on Evolutionary Computation*, pp. 1–8, IEEE, 2018.

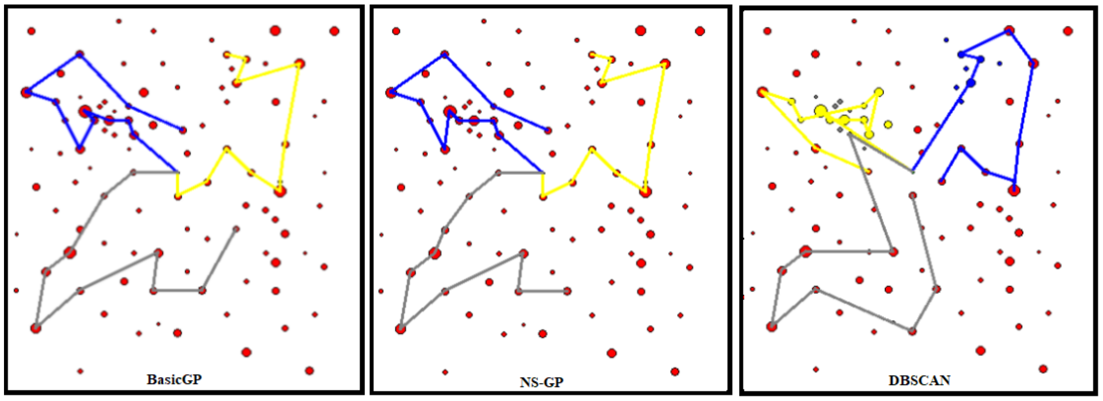

Fig. 4. BasicGP, DBSCAN and NS-GP solutions on instance rc104 (Day 1 - Yellow, Day 2 - Blue, Day 3 - Grey).

[2] P. Vansteenwegen and D. Van Oudheusden, "The mobile tourist guide: an or opportunity," *OR Insight*, vol. 20, no. 3, pp. 21–27, 2007.

[3] H. Tang and E. Miller-Hooks, "A TABU search heuristic for the team orienteering problem," *Computers & Operations Research*, vol. 32, no. 6, pp. 1379–1407, 2005.

[4] P. Vansteenwegen, W. Souffriau, G. Berghe, and D. Van Oudheusden, "Iterated local search for the team orienteering problem with time windows," *Computers & Operations Research*, vol. 36, no. 12, pp. 3281–3290, 2009.

[5] D. Karunakaran, Y. Mei, and M. Zhang, "Multitasking Genetic Programming for Stochastic Team Orienteering Problem with Time Windows," in *IEEE Symposium Series in Computational Intelligence*, 2019.

[6] V. F. Yu, A. P. Redi, P. Jewpanya, and A. Gunawan, "Selective discrete particle swarm optimization for the team orienteering problem with time windows and partial scores," *Computers & Industrial Engineering*, vol. 138, 2019.

[7] R. El-Hajj, D.-C. Dang, and A. Moukrim, "Solving the team orienteering problem with cutting planes," *Computers & Operations Research*, vol. 74, 2016.

[8] V. F. Yu, P. Jewpanya, S.-W. Lin, and A. P. Redi, "Team orienteering problem with time windows and time-dependent scores," *Computers & Industrial Engineering*, vol. 127, pp. 213–224, 2019.

[9] A. Gunawan, H. C. Lau, and K. Lu, "The latest best known solutions for the teamorienteering problem with time windows (toptw) benchmark instances," 2015.

[10] D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou, "Efficient Cluster-Based Heuristics for the Team Orienteering Problem with Time Windows," *Asia-Pacific Journal of Operational Research*, vol. 36, no. 01, 2019.

[11] J. Branke, S. Nguyen, C. Pickardt, and M. Zhang, "Automated design of production scheduling heuristics: A review," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 110–124, 2016.

[12] S. Nguyen, Y. Mei, and M. Zhang, "Genetic programming for production scheduling: a survey with a unified framework," *Complex & Intelligent Systems*, vol. 3, no. 1, pp. 41–66, 2017.

[13] 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*, pp. 290–297, ACM, 2017.

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

[15] K. Sylejmani, J. Dorn, and N. Musliu, "A tabu search approach for multi constrained team orienteering problem and its application in touristic trip planning," in *International Conference on Hybrid Intelligent Systems*, pp. 300–305, IEEE, 2012.

[16] S.-W. Lin and F. Y. Vincent, "A simulated annealing heuristic for the team orienteering problem with time windows," *European Journal of Operational Research*, vol. 217, no. 1, pp. 94–107, 2012.

[17] T. Cura, "An artificial bee colony algorithm approach for the team orienteering problem with time windows," *Computers & Industrial Engineering*, vol. 74, pp. 270–290, 2014.

[18] R. Montemanni, D. Weyland, and L. Gambardella, "An enhanced ant colony system for the team orienteering problem with time windows," in *2011 International Symposium on Computer Science and Society*, pp. 381–384, IEEE, 2011.

[19] Y. Liang and A. Smith, "An ant colony approach to the orienteering problem," *Journal of the Chinese Institute of Industrial Engineers*, vol. 23, no. 5, pp. 403–414, 2006.

[20] V. Papapanagiotou, D. Weyland, R. Montemanni, and L. Gambardella, "A sampling-based approximation of the objective function of the orienteering problem with stochastic travel and service times," in *5th International Conference on Applied Operational Research*, pp. 143–152, 2013.

[21] V. Papapanagiotou, R. Montemanni, and L. Gambardella, "Objective function evaluation methods for the orienteering problem with stochastic travel and service times," *Journal of applied Operational research*, vol. 6, no. 1, pp. 16–29, 2014.

[22] T. Ilhan, S. M. Iravani, and M. S. Daskin, "The orienteering problem with stochastic profits," *Iie Transactions*, vol. 40, no. 4, pp. 406–421, 2008.

[23] A. Gupta, R. Krishnaswamy, V. Nagarajan, and R. Ravi, "Running errands in time: Approximation algorithms for stochastic orienteering," *Mathematics of Operations Research*, vol. 40, no. 1, pp. 56–79, 2014.

[24] J. MacLachlan, Y. Mei, J. Branke, and M. Zhang, "Genetic Programming Hyper-Heuristics with Vehicle Collaboration for Uncertain Capacitated Arc Routing Problems," *Evolutionary Computation*, 2020, in press.

[25] S. Wang, Y. Mei, J. Park, and M. Zhang, "Evolving Ensembles of Routing Policies using Genetic Programming for Uncertain Capacitated Arc Routing Problem," in *IEEE Symposium Series in Computational Intelligence*, 2019.

[26] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, pp. 226–231, 1996.

[27] G. Righini and M. Salani, "Dynamic programming for the orienteering problem with time windows," tech. rep., Technical Report 91, Università degli Studi di Milano-Polo Didattico e di Ricerca di Crema, 2006.

[28] R. Montemanni and L. Gambardella, "An ant colony system for team orienteering problems with time windows," *Foundations of computing and Decision Sciences*, vol. 34, pp. 287–306, 2009.

[29] S. Luke *et al.*, "A java-based evolutionary computation research system." https://cs.gmu.edu/~eclab/projects/ecj/.