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

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

Mengjie Zhang
School of Engineering and Computer Science
Victoria University of Wellington
Kelburn 6011, New Zealand
Email: mengjie.zhang@ecs.vuw.ac.nz

***Abstract*—This paper investigates the stochastic team orienteering problem with time windows, which is a well known problem to model personalised tourist trip design. Specifically, we consider the stochastic visit duration, which may make preplanned trip infeasible. Existing studies focus on optimising robust solutions in advance, which is not effective in adjusting the subsequent trip in real time. Decision making policies, on the other hand, are effective heuristics to this end. However, it is very challenging to manually design effective policies. In this paper, we investigate automatically evolving policies for the stochastic team orienteering problem with time windows by genetic programming hyper-heuristics. We designed novel problem-specific features for the terminal set, and a meta-algorithm for fitness evaluation. Furthermore, we developed two look-ahead features that can provide more fruitful information than the basic features for real-time decision making. The experimental studies showed that the proposed genetic programming hyper-heuristic can evolve policies that are much better than the manually designed policies. In addition, it seems that the look-ahead features are not so effective when directly included in the terminals. This suggests the requirement of more intelligent ways of incorporating look-ahead information.**

I. Introduction

Personalised Tourist Trip Design Problem (TTDP) [1] is an important and challenging problem to solve. It is to design a (multi-day) trip, which is a sequence of visits to the Places-of-Interests (POIs), for a tourist under a given budget, so that the tourist's satisfaction of the trip is maximised. A typical objective is to maximise the total score (reflecting the satisfaction of the tourist) of the POIs visited during the trip.

TTDP can be formulated as an orienteering problem [2], which contains a variety of models. In this paper, we focus on the Team Orienteering Problem with Time Windows (TOPTW) [3] due to its closeness to the reality. TOPTW aims to design a multi-day ("teamed") trip where the visit of each POI must start within its time window. First, a tourist usually stays in a city/region for multiple days, and thus it is more reasonable to design a multi-day trip instead of a single-day trip. Second, many POIs (e.g. museums, parks, special events) have opening hours, and it is impossible to visit them outside their opening hours. TOPTW exactly considers these two real-world factors.

In this paper, we consider another important factor in addition to the above two factors, i.e. the stochastic visit durations. In real world, it is common that the actual visit durations are different from the expected values. The tourist may find a POI very attractive, and stay there longer than expected, while dislike another POI and leave much earlier than expected. In this case, the planned trip may become infeasible, as a longer stay of a POI may result in arriving some subsequent POI later than its time window. Therefore, in the *stochastic* TOPTW, it is necessary to adjust/plan the subsequent trip in real time.

There have been a number of studies for TOPTW (e.g. [3], [4], [5], [6], [7], [8]). However, none of them considered the stochastic visit duration, and the preplanned trip designed by existing studies may become infeasible in real time. As a result, the existing studies are not directly applicable to solving stochastic TOPTW.

Reactive decision making policies such as dispatching rules for job shop scheduling have shown to be very effective to make real-time response in a dynamic/stochastic environment. However, manually designing effective such policies is time-consuming and demands domain expertise. To address this issue, Genetic Programming Hyper-Heuristic (GPHH) has been successfully applied to automatically design (evolve) effective decision making policies for dynamic scheduling [9], [10] and routing [11], [12].

To the best of our knowledge, our work is the first time to consider evolving decision making policies for stochastic orienteering problems. This paper aims to propose a GPHH algorithm to evolve effective policies that make real-time decisions for stochastic TOPTW. More specifically, it contains the following research objectives.

- To design the *meta-algorithm*, i.e. the algorithm to generate a feasible solution for stochastic TOPTW given an instance and a policy.
- Design the features to be used by the GPHH to evolve effective policies.
- Analyse the structure of the evolved policies and understand the importance of the designed features in effective policies.

The rest of the paper is organised as follows. Section II gives the background introduction, including the problem description and related work. Then, the proposed GPHH is described in III. Then, the experimental studies and analysis

are carried out in Section IV. Finally, the conclusions and future work are given in Section V.

II. Background

A. Problem Description

In TOPTW, a set of n POIs $\mathbf{P} = \{1, \ldots, n\}$ are given. Each POI $p \in \mathbf{P}$ has a score $s(p)$, a visit duration $d(p)$ and a time window $[o(p), c(p)]$. A starting point $p_s \in \mathbf{P}$ and an ending point $p_e \in \mathbf{P}$ is also given. For each pair of POIs (p_i, p_j), the travel time $t(p_i, p_j)$ to travel from p_i to p_j is known. The time budget is given by m days, where each day has a starting time T_s and an ending time T_e. Then, the problem aims to design a set of sequences of POIs so that the total score of the visited POIs is maximised, and the following constraints are satisfied.

1) There are m sequences of POIs, each for a day.
2) Each sequence departs from p_s at time T_s, and returns to p_e no later than T_e.
3) Each POI is visited at most once.
4) The visit of p_i starts no earlier than o_i and no later than c_i. That is, if the tourist arrives p_i earlier than o_i, s/he has to wait until o_i to start the visit.

Assuming that a solution X is represented by $X = \{X_1, \ldots, X_m\}$, where $X_k = (x_{k1}, x_{k2}, \ldots, x_{kL_k})$, then the problem can be modelled as follows.

$$\max \ f(X) = \sum_{k=1}^{m} \sum_{l=1}^{L_k} s(x_{kl}), \tag{1}$$

$$s.t. \ x_{k1} = p_s, \ x_{kL_k} = p_e, \quad k = 1, \ldots, m \tag{2}$$

$$v(x_{k1}) = T_s, \ v(x_{kL_k}) \leq T_e, \quad k = 1, \ldots, m \tag{3}$$

$$v(x_{k(l+1)}) \geq v(x_{kl}) + d(x_{kl}) + t(x_{kl}, x_{k(l+1)}), \tag{4}$$

$$o(x_{kl}) \leq v(x_{kl}) \leq c(x_{kl}), \quad l = 2, \ldots, L_k - 1, \tag{5}$$

$$x_{k_1 l_1} \neq x_{k_2 l_2}, \quad k_1 \neq k_2 \text{ or } l_1 \neq l_2, \tag{6}$$

$$x_{kl} \in \mathbf{P}, \tag{7}$$

where $v(x_{kl})$ stands for the visit starting time of x_{kl}. Eq. (1) aims to maximise the total score of the visited POIs. Eq. (2) means that all the daily trips depart at p_s and return to p_m. Eq. (3) specifies that each daily trip starts at T_s and finishes no later than T_e. Eqs. (4) and (5) indicate that the visit of each POI cannot start before its arrival, and has to be within its time window. Eq. (6) means that each POI is visited no more than once. Finally, Eq. (7) is the domain constraint.

In Stochastic TOPTW, the visit duration $d(p)$ is a random variable instead of a deterministic value. In this case, the objective becomes maximising the *expected* total score.

B. Related Work

Orienteering Problem has been studied for many years [13], [2]. So far, researchers have considered many different variants such as multi-objective orienteering [14], time-dependent orienteering [15], team orienteering [16], orienteering with time windows [3] and mixtures of them [6], and have proposed effective solution methods such as mathematical programming [17], individual-based search (e.g. iterated local search [3], simulated annealing [7] and tabu search [18]) and population-based search (e.g. genetic algorithms [15], [8] and ant colony optimisation [19], [20]). However, these existing studies mainly focus on deterministic problems, where the information is fully known in advance, and will not change during the execution of the solution (i.e. during the process of visiting the POIs based on the planned trip).

There are a few works investigating stochastic orienteering problems. In [21], [22], a sampling-based method was proposed to approximate the expected objective value under stochastic travel and service times. A bi-objective genetic algorithm was proposed for orienteering problem with stochastic profit [23]. Efficient approximation algorithms were proposed for orienteering problem with stochastic service times [24], [25]. A variable neighbour search [26] was proposed for orienteering problem with stochastic travel and service times. However, all the existing studies focus on finding a robust preplanned *solution* that is expected to work well in all possible realised situations. Our work is the first attempt to find a *policy* that effectively reacts to real-time environment changes.

GPHH for automated design of policies has been successfully applied to a number of challenging combinatorial optimisation problems such as scheduling and vehicle routing, and many works have been done to investigate representation [27], [28], feature selection [29], [30] and generalisation [31]. These studies demonstrate the effectiveness of GPHH to evolve policies for dynamic/uncertain combinatorial optimisation problems. Therefore, in this paper, we will investigate the possibility of evolving policies for stochastic TOPTW by GPHH.

III. Genetic Programming Hyper-Heuristic for Stochastic Team Orienteering Problem with Time Windows

This section describes the GPHH algorithm proposed for evolving policies for stochastic TOPTW. The framework is given in Algorithm 1, which is a standard evolutionary algorithm framework. However, the individual representation (especially the terminal set of GP) and fitness evaluation are problem-specific, and will be described next.

Algorithm 1: The framework of GPHH

```
Randomly initialise a population of policies;
while Stopping criteria not met do
    Evaluate the policies in the current population;
    Generate an offspring population by applying
      crossover/mutation/reproduction operators;
    Select the policies from the current and offspring
      populations to form the new population;
end
```

A. Individual Representation

The proposed GPHH uses the traditional tree-based representation, and an individual (policy) is represented as a tree. It is essentially a priority function of the features of a POI.

Given any state during the decision making process, the policy calculates the priority of each candidate POI, and selects the one with the highest priority value to visit next. Figure. 1 shows an example of the policy "Score/TravelTime". In other words, the policy tends to select the POI with the highest score and shortest travel time from the current place to visit next.

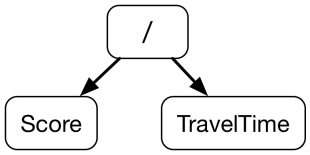

Fig. 1. An example of the policy Score/TravelTime.

The terminal and function sets are critical to the performance of GP [9]. In the proposed GP, the function set is set to $\{+, -, \times, /, \min, \max\}$, where all the operators take two arguments. Note that the "/" operator is protected, i.e. it returns 1 if divided by zero.

The selection of the terminals is problem dependent. It has been shown that time-invariant features are more effective in evolving dispatching rules for dynamic job shop scheduling [32]. Therefore, we design time-invariant features, i.e. the features relative to the current time, for stochastic TOPTW as well. More specifically, the features included in the terminal set of GPHH are described in Table I. For calculating the features involving the random duration $d(p)$, the expected duration is used.

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	$o(p) - \texttt{currTime}$
TC	time to the closing time	$c(p) - \texttt{currTime}$
TA	time to arrive the POI	$t(\texttt{currPlace}, p)$
TR	time to return to the end point	$t(p, p_e)$
TSV	time to start the visit	$\max\{\texttt{TO}, \texttt{TA}\}$
TFV	time to finish the visit	$\texttt{TSV} + E(d(p))$
SL	slack	$\texttt{TC} - \texttt{TA}$
RemT	remaining time budget	$\texttt{remainDays} \cdot (T_e - T_s)$ $+(T_e - \texttt{currTime})$

From the table, one can see that some features seem to be redundant as they can be represented by the combination of other features (e.g. $\texttt{TFV} = \texttt{TSV} + \texttt{DUR}$). However, previous studies [9] have shown that directly keeping promising (constructed) features may be helpful to GP evolution. Therefore, we keep all the features in Table I in the terminal set.

In addition to the above features, two look-ahead features are designed as well. They are described as follows.

1) MNS: the maximal score among the *feasible* next POIs.
2) ANS: the average score of the *feasible* next POIs.

For a candidate POI p, a feasible next POI p' must satisfy the following constraint:

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

Obviously, the calculations of MNS and ANS are much more time consuming than the features in Table I, as they require enumeration of all the unvisited POIs. However, they provide look-ahead information which can form more effective policies. In the experimental studies, we will investigate the effectiveness of the look-ahead features by comparing the GPHH with and without MNS and ANS.

B. Fitness Evaluation and Meta-Algorithm

The fitness evaluation of a GP individual is different from the traditional evaluation of a solution, since a GP individual is a policy rather than a solution. To evaluate a policy, one needs a set of sampled TOPTW instances (i.e. *training instances* $\mathcal{I}_{\text{train}}$) and a *meta-algorithm* (solution()) that generates a feasible solution given any TOPTW instance and a policy. Then, one can approximate the fitness of a policy as the average total score of the solutions generated for training instances. That is, for a policy pol,

$$\texttt{fit}(pol) = \frac{1}{|\mathcal{I}_{\text{train}}|} \sum_{I \in \mathcal{I}_{\text{train}}} \texttt{score}(\texttt{solution}(I, pol)). \quad (8)$$

The meta-algorithm is described in Algorithm 2. It is a event-driven decision process, where the events are stored in a priority queue. An event is triggered whenever a visit of a POI is finished, and the policy is applied to select the next POI to visit. At first, the decision process state (i.e. current day and time, current place and tour, and unvisited POIs) and the event queue are initialised (lines 1–3). Then, whenever a POI is visited, all the feasible unvisited POIs are examined. In line 6, an unvisited POI p is feasible if it satisfies the following constraints.

- p can be arrived before its closing time, i.e. $t + t(p^*, p) \leq c(p)$.
- After visiting p, the trip can return to the end point by T_e, i.e. $\max\{t + t(p^*, p), o(p)\} + d(p) + t(p, p_e) \leq T_e$.

Note that in line 22, the duration $d_{\text{sample}}(p_{\text{next}})$ is the actual (sampled) value. However, when examining the feasibility (line 6) and calculating the priority (line 14) of each unvisited POI, the actual visit duration is unknown since the POI is not visited yet. Therefore, the expected value $E(d(p))$ is used in these two situations.

C. Overall Algorithm

In this subsection, we give a more detailed description to the GPHH for stochastic TOPTW. The pseudo code is given in Algorithm 3. At each generation, the offspring population is generated by three genetic operators, i.e. crossover, mutation and reproduction. In addition, some elites are inherited from the previous population to improve convergence.

Note that in line 3, only one training instance is used ($|\mathcal{I}_{\text{train}}| = 1$). However, in each generation, a different sampled

Algorithm 2: `solution`$(\cdot,\cdot)$

Input: A TOPTW instance I, a policy pol.
Output: A feasible solution $X = \{X_1, X_2, \ldots, X_m\}$.

```
// initialise decision process
day ← 0, t ← T_s, p_c ← p_s, X_day ← (p_s);
Ω ← P \ {p_s, p_e};                    // unvisited POIs
queue ← {(p_s, t)};                    // initial event queue
while queue ≠ ∅ do
    // trigger the next event
    (p*, t*) ← poll(queue);
    Update the feasible unvisited POIs Ω' ⊆ Ω;
    if Ω' = ∅ then
        X_day ← (X_day, p_e);          // return to p_e
        day ← day + 1;                 // go to the next day
        if day = m then Return X;
        X_day ← (p_s);                 // open a new tour
        queue ← queue ∪ (p_e, T_s);
    end
    Calculate the priority value pol(p) of each p ∈ Ω';
    p_next ← arg max_{p∈Ω'} pol(p);
    Ω ← Ω \ p_next;
    t_arr ← t + t(p*, p_next);         // arrival time
    t_start ← max{t_arr, o(p_next)};   // visit start time
    if t_start > c(p_next) then
        // violate the time window, simply
           skip the visit
        queue ← queue ∪ (p_next, t_start);
    else
        sample the actual duration d_sample(p_next);
        t ← t_start + d_sample(p_next);
        X_day ← (X_day, p_next);
        queue ← queue ∪ (p_next, t);
    end
end
```

Algorithm 3: Pseudo code of GPHH for stochastic TOPTW

```
Initialise a population pop of policies by ramp-half-and-half;
while stopping criteria is not met do
    Sample an instance by sampling a value for all the
      random variables;
    Evaluate individuals in pop;
    pop' ← elites(pop);
    while |pop'| < |pop| do
        Randomly sample r ∈ U(0, 1);
        if r < P_x then
            // Do crossover
            Select two parents pol_1 and pol_2 from pop by
              tournament selection;
            Apply tree-based crossover to pol_1 and pol_2 to
              generate two offsprings pol'_1 and pol'_2;
            pop' ← pop' ∪ {pol'_1, pol'_2};
        else if r < P_x + P_m then
            // Do mutation
            Select pol ∈ pop by tournament selection;
            Apply tree-based mutation to pol to generate an
              offspring pol';
            pop' ← pop' ∪ pol';
        else
            // Do reproduction
            Select pol ∈ pop by tournament selection;
            pop' ← pop' ∪ pol;
        end
    end
    pop ← pop';
end
Return best individual in pop;
```

instance is generated (i.e. using a different random seed) by resampling the random duration variables. Such "instance rotation" has been shown to lead to much better generalisation of GPHH in other problems (e.g. [33], [11]).

IV. Experimental Studies

A. Experiment Settings

The commonly used c*_100, r*_100 and rc*_100 [34], c*_200, r*_200 and rc*_200 [20] and pr* [34], [20] are used in our experimental studies. These test instances were extended from vehicle routing instances [35], [36], in which the number of POIs ranges from about 50 to almost 300. Due to page limit, we selected 5 instances for each dataset (e.g. c101∼c105) in the experiments.

Fo each TOPTW instance, the stochastic version is generated by replacing each deterministic $d(p)$ value with a random variable $D(p) \sim \mathcal{N}(d(p), \sigma \cdot d(p))$, where σ is called the *uncertainty level* of the random variable. Here, the uncertainty level is set to $\sigma = 0.2$ as a rule of thumb. That is, the standard deviation of each duration is 20% of its mean value. For each instance, the versions with $m = 1$ (single-day trip) and $m = 3$ (3-day trip) are tested.

Table II shows the parameter setting of the GPHH proposed in this paper. The algorithm was implemented in Java using the ECJ library [37] and run on Linux OS with Intel(R) Core(TM) i7 CPU @3.60GHz. The GPHH with and without look-ahead features (i.e. `MNS` and `ANS` described in Section III-A) are compared. The two GPHH versions are named BasicGP and LA-GP respectively. Each compared algorithm was run 30 times independently, and the results were compared statistically. In addition, two manually designed policies are included in the comparison as a reference. One is the Score-Over-Slack (SOS, with priority function `SCORE/SL`) policy, and the other is the Score-Over-Time (SOT, with priority function `SCORE/TA`) policy.

TABLE II
The parameter setting of the GPHH.

Parameter	Value
Population size (`popsize`)	1024
Elitism	10 best individuals
Tournament selection size	7
Maximal depth	8
Crossover rate	80%
Mutation rate	15%
Reproduction rate	5%
Number of generations	51

For each stochastic TOPTW instance, all the compared policies will be evaluated on a test set with 500 randomly sampled instances. That is, the test fitness of a policy is

calculated as

$$\texttt{fit}_{\text{test}}(pol) = \frac{1}{|\mathcal{I}_{\text{test}}|} \sum_{I \in \mathcal{I}_{\text{test}}} \texttt{score}(\texttt{solution}(I, pol)), \quad (9)$$

where $|\mathcal{I}_{\text{test}}| = 500$. With such a large sample size, one can achieve a precise approximation of the expected performance over all the possible samples.

B. Experimental Results

Table III shows the test fitness values of BasicGP, LA-GP and the two manually designed policies (SOS and SOT) on the test instances with $m = 1$. In addition, BasicGP is compared with LA-GP statistically using Wilcoxon's rank sum test with significance level $\alpha = 0.05$. If BasicGP is significantly better than LA-GP, then the corresponding result is marked with "(+)". If BasicGP is significantly worse than LA-GP, then it is marked with "(-)".

From Table III, it is obvious that the policies evolved by BasicGP and LA-GP are both much better than the manually designed policies. On some instances (e.g. rc101, rc102, rc105 and r203), the average total score of the GP-evolved policies are even more than twice as the total score obtained by the manually designed policies. This demonstrates that the stochastic TOPTW is so challenging that it is hardly possible to manually design effective policies. In addition, the performance of SOS and SOT can be much different on the same instance (e.g. rc105 and c204). Note that the slack includes more information than the time to arrive (i.e. $\texttt{SL} = \texttt{TC} - \texttt{TA}$). The advantage of SOT over SOS on a decent number of instances (e.g. rc101, rc105, pr03 and pr13) indicates that using more advanced features like slack in manually designed policies may not always be helpful.

When comparing between BasicGP and LA-GP, one can be found that they perform statistically similar in almost all the instances with $m = 1$ (except pr01, where BasicGP significantly outperformed LA-GP). This indicates that simply including the look-ahead features into the terminal set of GP does not seem to improve the test performance of the GP-evolved policies.

Table IV shows the corresponding results for the instances with $m = 3$, i.e. 3-day trips. One can observe similar patterns as those in $m = 1$. The GP-evolved policies are much better than the manually designed policies. In addition, BasicGP performed statistically the same as LA-GP on most instances (LA-GP significantly outperformed BasicGP on 3 instances with "(-)", while defeated by BasicGP on 4 instances with "(+)".)

C. Further Analysis

First, we investigate the generalisation of the proposed GPHH algorithm. To this end, we arbitrarily selected eight instances (c103, r101, rc101, rc105, c203, r202, pr03, and pr14 with $m = 1$) and plot the convergence curves of the test performance of BasicGP and LA-GP on these instances in Fig. 2. Other instances show similar patterns. The ribbons are the standard deviation. From the figure, one can see that for most instances there is a clear upward trend of the test performance for both algorithms. In addition, both algorithms almost converged after 30 generations, and there was no overfitting. This verifies the effectiveness of the instance rotation during the training process. Also, the convergence curves of the two algorithms are very close to each other, indicating that including the look-ahead features into the terminals did not influence the GP evolution process much.

Then, we conduct some investigation on the importance of features. To this end, we selected some representative instances (r101, c203 and pr03 with $m = 1$), and plot the frequency (total occurrences) of the terminals in the 30 final policies obtained by the 30 runs of BasicGP and LA-GP. Other instances showed similar patterns. Figs. 3 and 4 show

TABLE III
THE TEST FITNESS OF THE COMPARED GPHH METHODS ON THE INSTANCES WITH $m = 1$. IF BASICGP IS STATISTICALLY SIGNIFICANTLY BETTER THAN LA-GP UNDER WILCOXON'S RANK SUM TEST WITH $\alpha = 0.05$, THEN THE CORRESPONDING RESULT IS MARKED WITH "(+)". IF BASICGP IS STATISTICALLY SIGNIFICANTLY WORSE THAN LA-GP, THEN THE CORRESPONDING RESULT IS MARKED WITH "(-)".

Instance	SOS	SOT	BasicGP	LA-GP
c101	187.16	159.74	292.44(10.25)	289.33(13.79)
c102	205.96	183.52	338.90(5.68)	335.62(7.57)
c103	264.46	245.84	375.89(8.26)	377.39(4.26)
c104	257.64	257.40	387.61(6.43)	386.82(7.75)
c105	192.32	171.42	323.02(8.56)	324.66(7.18)
r101	98.24	74.38	189.52(4.25)	187.59(11.06)
r102	121.10	175.03	273.66(6.82)	275.10(3.10)
r103	121.10	93.10	281.19(6.18)	280.32(8.32)
r104	149.09	128.04	288.08(11.90)	285.68(11.88)
r105	100.97	88.00	224.94(6.02)	221.98(10.18)
rc101	68.07	98.00	203.83(9.05)	202.91(9.81)
rc102	66.41	102.29	233.54(7.20)	234.81(9.03)
rc103	91.52	100.56	244.39(5.31)	241.93(10.36)
rc104	129.05	129.68	253.77(8.32)	254.80(7.03)
rc105	57.47	104.37	217.59(16.48)	220.19(14.61)
c201	210.00	182.16	822.96(10.20)	824.42(13.19)
c202	268.52	183.50	867.84(12.59)	867.96(14.76)
c203	435.94	233.66	916.61(11.48)	915.46(10.31)
c204	469.00	197.60	933.20(9.23)	929.83(6.67)
c205	273.72	188.40	861.22(9.63)	861.09(8.38)
r201	211.00	144.69	720.94(18.50)	720.80(14.35)
r202	309.73	257.08	840.37(15.17)	836.37(12.62)
r203	325.51	257.08	923.57(9.65)	924.27(12.34)
r204	522.80	307.24	1004.91(16.66)	1006.66(10.45)
r205	226.92	240.40	831.81(12.11)	828.60(15.25)
rc201	182.99	117.49	729.19(20.24)	734.17(19.46)
rc202	214.20	186.40	859.00(29.34)	864.76(17.16)
rc203	235.10	188.66	926.51(10.38)	928.59(8.42)
rc204	266.33	400.14	1040.73(13.96)	1039.69(13.62)
rc205	220.91	147.50	795.34(17.91)	790.46(18.13)
pr01	127.35	88.51	287.99(3.50)**(+)**	284.20(5.64)
pr02	128.91	99.83	383.04(3.09)	380.80(8.81)
pr03	93.00	180.06	310.41(5.40)	311.30(3.33)
pr04	133.97	125.63	341.93(8.13)	341.89(8.40)
pr05	115.18	144.66	391.88(15.63)	391.23(20.75)
pr11	140.94	168.94	311.50(12.20)	305.05(17.73)
pr12	138.68	187.60	402.91(6.09)	405.88(4.66)
pr13	90.46	183.18	373.90(11.58)	369.98(21.23)
pr14	119.04	226.82	405.52(18.82)	404.36(14.74)
pr15	103.10	355.43	458.58(10.36)	461.24(11.11)

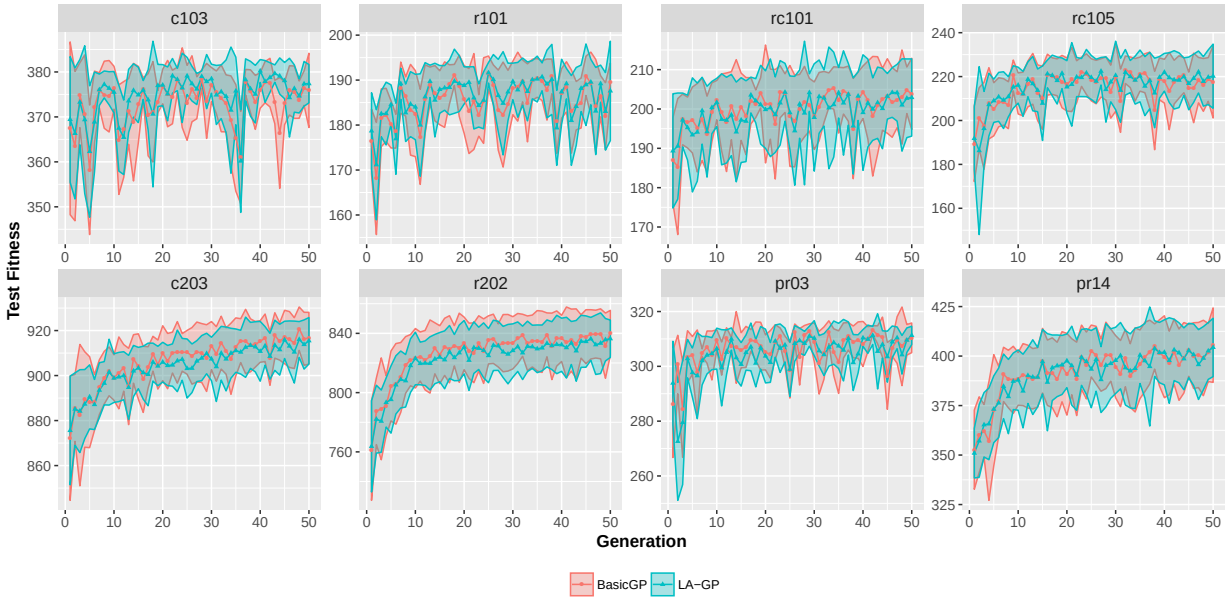

Fig. 2. The convergence curves of the test performance of BasicGP and LA-GP on c103, r101, rc101, rc105, c203, r202, pr03 and pr14.

the terminal frequencies of BasicGP and LA-GP, respectively. Note that in Fig. 3, the frequencies of ANS and MNS are zero, as these two features are not used in BasicGP.

From the figures, one can see that SCORE is always important in all the instances, which is consistent with intuition, as the objective is to maximise the total score. Then, either TSV or TFV or both appeared frequently in the final policies obtained by both algorithms in all the instances. This is also consistent with intuition, as one tends to select POIs with shortest time to start or finish the visit. On the other hand, TO, TC, TA and $\mathtt{SL} = \mathtt{TC} - \mathtt{TA}$ did not often appear. Note that $\mathtt{TSV} = \max\{\mathtt{TO}, \mathtt{TA}\}$, Figs. 3 and 4 indicate that some features are redundant, and TO, TC, TA or/and SL may be removed in the presence of TSV or TFV.

For c203 and pr03, the frequencies of the look-ahead features ANS and MNS are quite low. For r101, ANS and MNS seemed to appear a bit more frequently. More specifically, the frequencies of ANS and MNS in the final policies of the 30 runs of LA-GP is shown in 5. It can be seen that the maximal frequency is ANS appearing for 5 times in runs 9, 20 and 28. Correspondingly, the test fitness of these three runs are 194.41, 190.7 and 190.9, which are higher than the average test fitnesses of both algorithms (189.52 and 187.59). As an example, we investigated the structure of the final policy obtained by run 9. After simplification, the policy can be written as follows.

$$\begin{aligned}&\mathtt{ANS} - \max\{\mathtt{MNS}, \mathtt{BB}\} + 2 \cdot \mathtt{BB} - \mathtt{DUR} - \\ &\max\{\frac{\max\{\mathtt{DUR}, \mathtt{RemT}\} - \mathtt{TFV}}{\mathtt{TA} - \mathtt{TR}}, \\ &\min\{\max\{\mathtt{ANS}, \mathtt{SCORE}\}, \mathtt{RemT} + \mathtt{SL}\}\},\end{aligned}$$

where $\mathtt{BB} = \mathtt{SCORE} + \mathtt{ANS}$. From the policy, one can see that $\mathtt{SCORE} + \mathtt{ANS}$ can be a promising constructed feature, which stands for the sum of the score of the current POI and the expected (average) score of next POI. The policy clearly attempts to select the POI with the maximal $\mathtt{SCORE} + \mathtt{ANS}$ value. This shows that the look-ahead features can be helpful to the effectiveness of the policy. Besides, this demonstrates the ability of GP to automatically construct promising features from low-level features.

Finally, Table V shows the average training time (in seconds) of BasicGP and LA-GP over all the 40 TOPTW instances in the experiment. From the table, it is clear that LA-GP is significantly slower than BasicGP. This is mainly due to the calculation of the ANS and MNS features, which requires the enumeration of all the unvisited POIs. In other words, although the look-ahead features provide more comprehensive information for decision making, they are much more time consuming to calculate, which makes the GPHH much less efficient.

V. Conclusions and Future Work

In this paper, a GPHH method was proposed for evolving policies for stochastic TOPTW. In particular, a meta-algorithm for fitness evaluation is developed and a set of features are designed for the terminal set of the GPHH. Two versions of GPHH, one with and the other without the two look-ahead features, are compared on benchmark instances. The results showed that the current terminal set may contain redundant features, and may be improved by identifying and removing the redundant features. Also, it is demonstrated that the look-ahead features can help improve the effectiveness of the policy if used properly. However, simply including the look-ahead features into the terminal set seems not to be effective in employing the look-ahead information. In future work, we will consider feature selection and construction so that the

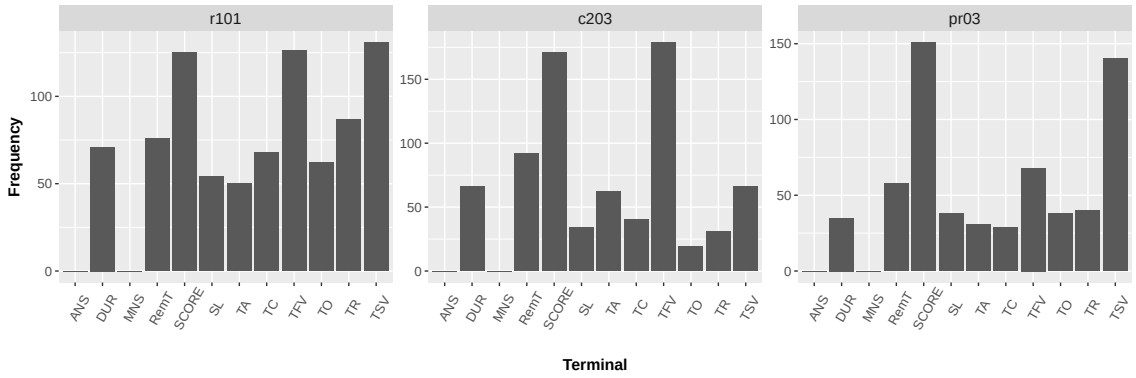

Fig. 3. The terminal frequencies of the final policies obtained by the 30 runs of BasicGP on r101, c203 and pr03 with $m = 1$.

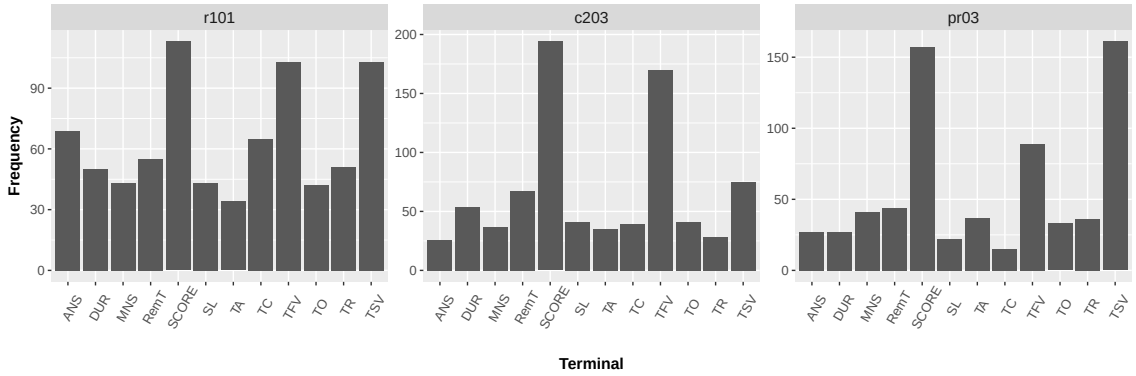

Fig. 4. The terminal frequencies of the final policies obtained by the 30 runs of LA-GP on r101, c203 and pr03 with $m = 1$.

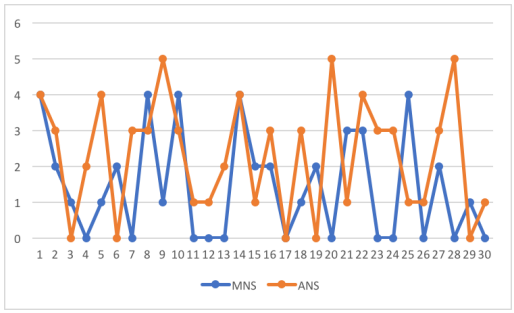

Fig. 5. The frequencies of `ANS` and `MNS` in the final policies obtained by the 30 runs of LA-GP on r101 with $m = 1$.

look-ahead information can be employed in a more intelligent way. In addition, new features with better balance between effectiveness and efficiency are to be developed.

REFERENCES

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

[2] A. Gunawan, H. C. Lau, and P. Vansteenwegen, "Orienteering problem: A survey of recent variants, solution approaches and applications," *European Journal of Operational Research*, vol. 255, no. 2, pp. 315–332, 2016.

[3] 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.

[4] W. Souffriau, P. Vansteenwegen, G. Vanden Berghe, and D. Van Oudheusden, "The multiconstraint team orienteering problem with multiple time windows," *Transportation Science*, vol. 47, no. 1, pp. 53–63, 2013.

[5] D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou, and Y. Tasoulas, "Cluster-based heuristics for the team orienteering problem with time windows," in *International Symposium on Experimental Algorithms*. Springer, 2013, pp. 390–401.

[6] D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou, and N. Vathis, "Efficient heuristics for the time dependent team orienteering problem with time windows," in *Applied Algorithms*. Springer, 2014, pp. 152–163.

[7] 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.

[8] N. Labadie, J. Melechovskỳ, and R. W. Calvo, "Hybridized evolutionary local search algorithm for the team orienteering problem with time windows," *Journal of Heuristics*, vol. 17, no. 6, pp. 729–753, 2011.

[9] 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.

[10] 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.

[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. 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)*. IEEE, 2017, pp. 1948–1955.

TABLE IV

THE TEST FITNESS OF THE COMPARED GPHH METHODS ON THE INSTANCES WITH $m = 3$. IF BASICGP IS STATISTICALLY SIGNIFICANTLY BETTER THAN LA-GP UNDER WILCOXON'S RANK SUM TEST WITH $\alpha = 0.05$, THEN THE CORRESPONDING RESULT IS MARKED WITH "(+)". IF BASICGP IS STATISTICALLY SIGNIFICANTLY WORSE THAN LA-GP, THEN THE CORRESPONDING RESULT IS MARKED WITH "(-)".

Instance	SOS	SOT	BasicGP	LA-GP
c101	488.46	339.02	759.64(9.63)	764.59(4.83)
c102	538.34	386.50	854.93(8.62)	858.90(11.82)
c103	576.90	503.08	908.82(6.69)	909.86(1.38)
c104	668.90	612.58	954.18(13.46)	953.51(7.83)
c105	527.74	345.38	805.41(9.47)	806.09(8.17)
r101	240.84	181.61	434.46(11.84)	439.18(8.95)
r102	297.38	285.34	603.07(15.47)	602.76(21.14)
r103	323.44	270.31	656.17(11.44)	653.98(16.79)
r104	306.39	447.93	711.40(15.70)	713.47(10.82)
r105	256.52	202.53	546.63(16.83)	548.51(13.70)
rc101	275.79	187.65	562.82(12.49)	563.28(16.79)
rc102	269.82	243.99	620.66(13.54)	618.41(17.62)
rc103	279.12	205.06	667.10(20.07)	665.63(15.06)
rc104	312.69	557.12	721.00(14.32)	721.39(11.40)
rc105	250.46	246.25	611.26(13.20)**(-)**	618.42(9.23)
c201	542.52	722.00	1765.92(5.04)**(+)**	1757.61(4.48)
c202	675.46	616.58	1774.19(5.69)	1773.14(10.79)
c203	898.84	622.68	1767.32(5.81)	1767.91(5.43)
c204	1008.46	948.68	1779.52(7.46)	1775.30(16.58)
c205	659.62	599.82	1782.96(3.33)**(-)**	1784.29(2.61)
r201	588.73	402.01	1357.85(20.99)	1350.16(20.94)
r202	786.46	593.48	1415.45(8.58)**(+)**	1408.21(8.22)
r203	750.60	858.52	1438.33(1.79)	1437.19(3.14)
r204	1028.85	1008.51	1440.19(3.61)	1438.27(13.71)
r205	788.44	819.42	1437.14(4.51)	1437.14(3.19)
rc201	640.44	407.00	1597.26(17.15)	1595.38(15.60)
rc202	768.40	524.57	1656.74(20.01)	1653.82(22.96)
rc203	703.26	599.21	1712.06(4.90)**(+)**	1708.82(7.67)
rc204	793.36	1299.10	1720.24(2.49)	1720.54(1.17)
rc205	622.49	555.27	1624.24(22.86)	1629.21(31.23)
pr01	246.86	294.71	559.84(12.25)**(-)**	568.47(6.11)
pr02	338.35	474.12	864.08(10.97)	853.41(21.48)
pr03	266.48	383.93	770.33(19.91)	775.33(8.95)
pr04	391.48	448.86	855.00(4.76)	846.50(41.50)
pr05	276.33	464.83	908.18(21.45)	911.95(14.53)
pr11	369.38	457.60	621.10(8.69)	619.72(14.64)
pr12	396.00	657.75	914.26(10.34)	912.31(13.59)
pr13	341.32	611.21	889.10(16.95)**(+)**	878.93(22.28)
pr14	385.58	656.05	920.71(21.19)	929.29(16.20)
pr15	401.63	739.16	970.11(23.57)	974.61(18.28)

TABLE V

THE AVERAGE TRAINING TIME (IN SECONDS) OF BASICGP AND LA-GP OVER ALL THE TOPTW INSTANCES.

	BasicGP	LA-GP
Avg. Time	966.30	26182.05

[13] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011.

[14] M. Schilde, K. Doerner, R. Hartl, and G. Kiechle, "Metaheuristics for the bi-objective orienteering problem," *Swarm Intelligence*, vol. 3, no. 3, pp. 179–201, 2009.

[15] Y. Mei, F. D. Salim, and X. Li, "Efficient meta-heuristics for the multi-objective time-dependent orienteering problem," *European Journal of Operational Research*, vol. 254, no. 2, pp. 443–457, 2016.

[16] I. Chao, B. Golden, and E. Wasil, "The team orienteering problem," *European Journal of Operational Research*, vol. 88, no. 3, pp. 464–474, 1996.

[17] A. Leifer and M. Rosenwein, "Strong linear programming relaxations for the orienteering problem," *European Journal of Operational Research*, vol. 73, no. 3, pp. 517–523, 1994.

[18] 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 *Hybrid Intelligent Systems (HIS), 2012 12th International Conference on*. IEEE, 2012, pp. 300–305.

[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] 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.

[21] 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, Proceedings, Lecture Notes in Management Science*, 2013, pp. 143–152.

[22] 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.

[23] 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.

[24] A. Gupta, R. Krishnaswamy, V. Nagarajan, and R. Ravi, "Approximation algorithms for stochastic orienteering," in *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2012, pp. 1522–1538.

[25] ——, "Running errands in time: Approximation algorithms for stochastic orienteering," *Mathematics of Operations Research*, vol. 40, no. 1, pp. 56–79, 2014.

[26] A. M. Campbell, M. Gendreau, and B. W. Thomas, "The orienteering problem with stochastic travel and service times," *Annals of Operations Research*, vol. 186, no. 1, pp. 61–81, 2011.

[27] S. Nguyen, M. Zhang, M. Johnston, and K. Tan, "A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 621–639, 2013.

[28] D. Jakobović, L. Jelenković, and L. Budin, "Genetic programming heuristics for multiple machine scheduling," in *Genetic Programming*. Springer, 2007, pp. 321–330.

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

[30] Y. Mei, S. Nguyen, B. Xue, and M. Zhang, "An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 5, pp. 339–353, 2017.

[31] Y. Mei and M. Zhang, "A comprehensive analysis on reusability of gp-evolved job shop dispatching rules," in *Proceedings of Congress on Evolutionary Computation*. IEEE, 2016.

[32] Y. Mei, S. Nguyen, and M. Zhang, "Evolving time-invariant dispatching rules in job shop scheduling with genetic programming," in *European Conference on Genetic Programming*. Springer, 2017, pp. 147–163.

[33] 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.

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

[35] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations research*, vol. 35, no. 2, pp. 254–265, 1987.

[36] J. Cordeau, M. Gendreau, and G. Laporte, "A tabu search heuristic for periodic and multi-depot vehicle routing problems," *Networks*, vol. 30, no. 2, pp. 105–119, 1997.

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