Multitasking Genetic Programming for Stochastic Team Orienteering Problem with Time Windows

Deepak Karunakaran, Yi Mei, Mengjie Zhang
Victoria University of Wellington
Wellington, New Zealand
Email: {deepak.karunakaran, yi.mei, mengjie.zhang}@ecs.vuw.ac.nz

***Abstract*—The tourism industry is witnessing high growth in recent years leading to a large number of options for a tourist. Personalised tourist trip design is faced with many places of interests, different tourist preferences and uncertainty in visit duration. In this paper, we study the stochastic Team Orienteering Problem with Time Windows (TOPTW) that well models the personalised tourist trip design. Under an uncertain environment, determining a robust solution in advance is not very effective due to frequent changes in the trip. Reactive decision-making policies have shown to be effective alternatives. Genetic programming-based hyper-heuristic (GPHH) approaches have been explored to automatically design policies. However, GPHH is computationally intensive. Considering a large number of trip design scenarios (e.g. cities), evolving a policy for each of these scenarios individually is difficult and time consuming. In this work, we propose a multitasking GPHH approach based on island model to evolve a set of policies which are effective across multiple trip design scenarios. The experimental studies show that our multitasking approach which needs only a single run to evolve policies for all problem instances is both efficient and effective when compared with the standard GPHH approach which requires a separate population for each TOPTW instance and runs sequentially.**

I. Introduction

Tourist itinerary planning is an important activity for a tourist who is travelling to an unknown city and wants to maximize the positive experience of the trip. This activity becomes quite challenging due to a large number of places of interest (POIs) with numerous constraints and the tourist showing varying levels of preferences for various POIs [1]. In the real world, trip planning is further affected by uncertainties like traffic and weather requiring frequent modifications or even partial cancellation of trips. The personalized tourist trip design problem (TTDP) is to design a trip comprising of sequence of visits under budgetary constraints while maximizing the tourist's satisfaction.

TTDP has been formulated as a Team Orienteering Problem with Time Windows (TOPTW) [2] which designs a multi-day trip such that the visit to each POI must fall within a time window. This models realistic scenarios because a tourist generally stays within a city for multiple days and need to plan his visits to POIs while considering their opening and closing hours. In the real world, actual visit duration is usually uncertain due to factors like traffic and weather. Taking uncertainty into consideration, Mei et al. [3] presented a stochastic TOPTW problem which can take into account events like extended stay at some POIs, cancellation of a subset of visits, etc., while incorporating modifications to the original trip.

Stochastic TOPTW problems model the trip design problem under more realistic settings by incorporating the uncertainties which arise during a tour. Consequently, solving stochastic TOPTW problems effectively will require solution strategies which are reactive in nature. To this end, using decision making policies has been shown to be effective viz., shortest travel time (STT) policy, nearest neighbor policy (NN), etc. But manually designing these policies is difficult and time consuming. For other optimization problems like dynamic scheduling and routing, which present similar challenges, genetic programming-based hyper-heuristic approach has been shown to be very useful in automatically designing reactive decision making policies (e.g. dispatching rules for dynamic job shop scheduling problems [4] and routing policies for uncertain arc routing [5]). In light of exciting results from such works, Mei et al. [3] have developed a GPHH approach to evolve effective decision making policies for stochastic TOPTW problems. They design features which could be used by GPHH to evolve effective policies and develop an decision-making heuristic framework (so-called *meta-algorithm*) toward generating feasible solution for stochastic TOPTW problems using the evolved policies.

Even though stochastic TOPTW problems are an important step toward considering realistic scenarios and design of solution methodologies like GPHH for automatically designing reactive decision making policies have shown promise, there are many important challenges which needs addressing. Considering the exponentially growing tourism industry across the world, the number of destination cities with a large number of POIs is growing at a very fast rate, making the trip design problem even more computationally challenging. Due to various geographical, economic and historical factors, different cities present POIs with vastly different characteristics. For example, in some cities POIs are clustered around historically significant locations. Some POIs like natural formations are located at the outskirts of cities while others like museums and galleries are within the city. Similarly, tourists have different preferences; an adventure seeking tourist will look for POIs related to water sports or bungee jumping where as an art lover will look for galleries and museums.

GPHH is a computationally intensive approach [3] and designing policies for each of the cities while considering

the numerous possible scenarios arising due to the tourist preferences, the constraints and the characteristics of POIs is very difficult and challenging. To emulate the success of GPHH in automatically designing reactive policies for other dynamic optimization problems like scheduling in stochastic TOPTW problems, it is of vital importance to develop algorithms which could evolve effective policies for different cities and trip design scenarios under reasonable computational budget. Moreover, it is also equally important to consider the stochastic TOPTW problems for designing policies which reflect the aforementioned trip scenarios.

Recently, Evolutionary Multi-Tasking Optimization (EMTO) has been proposed [6] which conducts evolutionary search on multiple search space concurrently where the search spaces correspond to different tasks or optimization problems. EMTO exploits the latent synergies [7] between the different tasks through knowledge sharing across the similar tasks or optimization problems. Unlike traditional evolutionary algorithms (EAs) which solve a single optimization problem in each run, the EMTO algorithms are capable of solving multiple optimization problems in a single run [8]. Considering the challenge faced by the current GPHH approach in evolving effective policies for each and every city and scenario, it is evident that the EMT approaches have the potential to address this issue.

Furthermore, one of the advantages of hyper-heuristic approaches is that while effectively automating the design of heuristics (policies in our case) it also allows researchers to utilize machine learning concepts toward developing algorithms for solving difficult optimization problems [9]. Moreover, GPHH approaches present a lot of opportunities to address a variety of challenges in optimization due to its characteristics like flexible representation. For example, Park et al. [10] explore ensemble learning methods, which are quite popular with supervised machine learning, toward evolving effective dispatching rules represented as genetic programs (GPs) for dynamic job shop scheduling problems.

Motivated by the potential of these technologies, in this work we aim to efficiently evolve effective policies for TOPTW problems under uncertain conditions reflecting realistic trip design scenarios. Specifically, we aim to develop a new evolutionary multitasking approach towards efficiently designing a set of scenario-specific policies for stochastic TOPTW problems using GPHH. Instead of evolving a separate policy using an evolutionary run for each scenario, our goal is to develop a multitasking approach which can evolve policies for multiple scenarios in a single run. We expect the proposed algorithm to achieve better performance than the existing algorithms within comparable time budget. The specific research objectives are as follows.

- Develop a feature construction and clustering method to be used by the multitasking algorithm to group the scenarios.
- Develop a new EMTO GP algorithm leveraging an island model for evolution to enable knowledge sharing across search spaces.
- Improve the effectiveness of proposed EMTO GP algorithm by developing a Thompson sampling approach towards giving more emphasis to problem instances with specific characteristics.

The rest of the paper is organized as follows. In Section II, we present the background where we present the mathematical model of the problem and discuss the related works from literature. In Section III, we present our proposed multitasking approach based on island model. The experiment design is discussed in Section IV and the experimental results are presented in Section V. Finally, Section VI provides the conclusions and future work.

II. Background

A. Problem Description

We describe the stochastic TOPTW below. Given a set of n POIs $\mathcal{P} = \{p_1, \ldots, p_2\}$. Each POI p_i is associated with a score $s(p_i)$, visit duration $d(p_i)$ and time window $[o(p_i), c(p_i)]$. The visit duration $d(p_i)$ is a random variable, while the score and time window are deterministic, and exactly known in advance. There are starting and ending POIs $\{p_s, p_e\} \in \mathcal{P}$. For each pair of POIs, the travel time between them is denoted as $t(p_i, p_j)$. For the sake of simplicity, the travel time is deterministic and static. The time window of a daily trip is $\{T_s, T_e\}$.

Then, the stochastic TOPTW is to design a m-day trip plan, which is a set of sequences of POIs, so that the expected total score of the visited POIs is maximised. Meanwhile, the following constraints have to be satisfied: (a) There are m sequences in total (one for each day). (b) Each sequence starts at p_s at T_s and ends at p_e no later than T_e. (c) Each POI is visited at most once. (d) The visit of a POI p_i can start only within its time window. That is, if the tourist arrives at p_i earlier than $o(p_i)$, they have to wait until $o(p_i)$ to start the visit.

In this paper, we represent a solution as $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$ where $\mathcal{X}_k = \{x_{k1}, x_{k2}, \ldots, x_{kL_k}\}$ is a sequence (daily trip) for kth day. The problem [3] can be formulated as follows.

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

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

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

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

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

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

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

where $v(x_{kl})$ is the visit starting time for x_{kl}. Eq. (1) is the objective function, which is the total score of the visited POIs to be maximized. Eqs. (2) and (3) indicate that each daily trip starts and ends at the proper places, and is within the time budget. Eq. (4) means that for each visited POI, the visit cannot start until the tourist arrives at the POI. Eq. (5) indicates that the visit has to be within the time window of the POI.

Eq. (6) implies that each POI is visited no more than once. Finally, eq. (7) defines the domain of the variables.

B. Related Work

The orienteering problem (OP) originally proposed as an NP-hard problem in [11] has many variants [12]. OP is basically a routing problem consisting of a subset of nodes which must be visited under time constraint with the goal of maximizing the total collected score. Essentially, it is the combination of the Knapsack problem and the Travelling salesman problem [2]. There are a number of variants of OP, viz. Team OP, Multi-agent OP, Multi-Period OP, etc. The OP which constrains the service at a node to a predefined time window are called OP with time windows (OPTW). When the OP requires multiple paths as solution, it is called a Team OP and consequently for the case of predefined time windows, its called Team OPTW (TOPTW). For more practical scenarios, when travel times between nodes cannot be determined in a deterministic way, OP with stochastic aspects are defined [3]. Some of the more recent and prominent techniques from the literature toward solving TOPTW problems are [13], [14] which are based in iterated local search. Gavalas et al. present cluster based algorithms to solve tourist trip design problems as an application of TOPTW problems. [15]–[17] are some of the recent works which present approaches to solve stochastic TOPTW problems.

Evolutionary Multi-Tasking Optimization (EMTO) is a relatively new research direction led by prominent works like [6], [18]. EMTO exploits the implicit parallelism of population-based search algorithms to simultaneously tackle multiple distinct optimization tasks. EMTO approaches try to map the optimization tasks to a unified search space, which through continuous genetic transfer enables the evolving population of candidate solutions to harness the hidden relationships between them. Though the mode of genetic transfer is usually crossover, newer approaches which use viz. explicit autoencoding [19] have been proposed. As an example of application of EMTO for combinatorial optimization problems, [20] is a recent work which develops multitasking methods for dynamic resource allocation problems by adaptively allocating computational resources based on complexities of the tasks. Park et al. [10] investigate an EMTO approach using niched genetic programming for dynamic job shop scheduling problems under machine breakdowns.

Island models are one of the important classes of parallel evolutionary algorithms which involve communication between subpopulations (across different machines) [21]. The communication is defined as exchange of individuals which is based on the topology of the island model. The topology is essentially a directed graph with the islands as its nodes. By design, the island models incorporate the dynamics of exploration and exploitation. Through diversity maintenance across the different islands the exploration is promoted while focusing on a specific region of search space, the islands promote exploitation [22]. The dynamics of exploration and exploitation have been empirically demonstrated further in [23].

Island models for evolutionary computation have shown considerable interest in recent years [24]. For example, [25] leverage the implicit ability of island models to control exploration and exploitation in the search space toward developing effective GPHH methods toward evolving dispatching rules for multi-objective scheduling problems. EMTO approaches aim to use a single or lower number of populations to evolve solutions for multiple tasks. One of the primary characteristics of EMTO approaches is knowledge transfer [6] across multiple tasks (or problems). As mentioned above, this is achieved through genetic transfer which has been explored through different techniques. Since island models incorporate migration of individuals, they are a candidate for achieving genetic transfer for EMTO. More recently, [26] demonstrated that knowledge transfer using the migration policies of a relatively simple island model-based multitasking approach performs as good as the more complicated EMTO approaches in literature.

In recent years, GPHH has been very successful in automatically designing policies for a number of combinatorial optimization problems such as scheduling [4], vehicle routing [27], etc. GPHH has also been demonstrated to be effective for problems under uncertainty [25]. Mei et al. [3] have successfully employed GPHH for stochastic TOPTW problems and shown that GPHH is a very promising approach for this type of problem.

III. Proposed Method

In this section, we present our proposed EMTO GP approach to evolve a set of policies. To this end, first we develop a feature extraction and clustering method for the problem scenarios. The purpose of this step is to group together similar TOPTW problems. Then we present our island model-based EMTO approach which uses the outcome of the clustering method. We also describe the migration strategies of the island model. Furthermore, to improve the search efficiency, a Thompson sampling approach is developed, which can exploit the relationship between the clusters of problem scenarios and the performance of the GP-evolved policies, thus can push the evolutionary search toward evolving policies which are able to cover contrasting trip design scenarios effectively.

A. Feature Extraction and Problem Scenario Clustering

As described earlier in Section 1, a large number of cities, the characteristics of POIs and varying interests of tourists associated with the POIs results in a large number of scenarios for trip design. We develop a feature extraction method so that the extracted features can be used to group similar TOPTW instances together.

The following motivations play a role in the design of our features. Firstly, the distribution of the geological locations of POIs with respect to each other is an important characteristic of a TOPTW problem. For example, a city with two prominent clusters of POIs separated geographically presents a problem different to a large city with POIs spread across the city. Secondly, the distribution of the preferences (score value of POI) of a tourist is very important. A tourist who is relaxed

and wishes to spend equal time across all POIs is different from a tourist who is interested in specific attractions. Thirdly, the duration of visit is again dependent on the preference of a tourist and it could vary across POIs for tourists.

Based on the above considerations, we describe our feature extraction procedure. To capture the geographical distribution, we calculate the distances between each pair of POIs, say

$$\mathcal{D} = \{dist(p_1, p_2), \ldots, dist(p_{n-1}, p_n)\}.$$

Then, we calculate the four quartile values for $\mathcal{D}$,

$$\{q_1^{\mathcal{D}}, q_2^{\mathcal{D}}, q_3^{\mathcal{D}}, q_4^{\mathcal{D}}\}.$$

Similarly, we calculate the four quartiles for duration and score values as $\{q_1^{\mathcal{T}}, q_2^{\mathcal{T}}, q_3^{\mathcal{T}}, q_4^{\mathcal{T}}\}$ and $\{q_1^{\mathcal{S}}, q_2^{\mathcal{S}}, q_3^{\mathcal{S}}, q_4^{\mathcal{S}}\}$ respectively, where $\mathcal{T} = \{t(p_1, p_2), \ldots, t(p_{n-1}, p_n)\}$ and $\mathcal{S} = \{s(p_1), \ldots, s(p_n)\}$. These three sets of values together form the feature vector $\mathcal{F}$.

$$\mathcal{F} = \{q_1^{\mathcal{D}}, q_2^{\mathcal{D}}, q_3^{\mathcal{D}}, q_4^{\mathcal{D}}, q_1^{\mathcal{T}}, q_2^{\mathcal{T}}, q_3^{\mathcal{T}}, q_4^{\mathcal{T}}, q_1^{\mathcal{S}}, q_2^{\mathcal{S}}, q_3^{\mathcal{S}}, q_4^{\mathcal{S}}\}$$

Given a set of n stochastic TOPTW problem instances, we generate a hierarchical grouping of the problem instances by recursively applying the K-means clustering method. Specifically, we first extract features for all these problems and cluster them into two groups $\mathcal{C}_1$ and $\mathcal{C}_2$ using K-means clustering ($k = 2$). Then, we apply K-means clustering ($k = 2$) again on each of these clusters to yield $\{\mathcal{C}_{11}, \mathcal{C}_{12}\}$ and $\{\mathcal{C}_{21}, \mathcal{C}_{22}\}$ respectively. This process can be repeated to obtain more sub-clusters $\{\{\mathcal{C}_{111}, \mathcal{C}_{112}\}, \{\mathcal{C}_{121}, \mathcal{C}_{122}\}\}$ and $\{\{\mathcal{C}_{211}, \mathcal{C}_{212}\}, \{\mathcal{C}_{221}, \mathcal{C}_{222}\}\}$ and so on. We employ this kind of hierarchical clustering technique because we will leverage the proximity between a pair of clusters in our EMTO GP approach.

B. Multitasking Approach using Island Model

Due to the simplicity of island models and their utility in hyper-heuristic search, we choose to develop a multitasking approach to our problem using island models. The notation used is described in Table I.

TABLE I
THE NOTATION USED IN THE ISLAND MODEL-BASED APPROACH.

Notation	Description
C_x	cluster used in island x in current generation
g	current generation
N_G	total number of generations
G, A, B	names of islands whose clusters are fixed throughout evolution.
$X1, X2, Y1, Y2$	names of islands whose clusters vary every n_p generations.
ω_x	policy evolved from island x.
n_p	number of generations before new clusters are selected
$\mathcal{M}_{\overrightarrow{I_1, I_2}}$	migration strategy from island I_1 to I_2.
$\mathcal{I}$	a stochastic TOPTW instance
$s(\omega, \mathcal{I})$	total score calculated using policy ω on instance $\mathcal{I}$

The overall framework of the proposed EMTO GP algorithm with island model is described in Algorithm 1.

Algorithm 1: The overall framework.

```
Extra features for each training instance;
Cluster the training instances into C_1 and C_2;
Further cluster C_1 and C_2 using hierarchical clustering;
Create 7 islands G, A, B, X1, X2, Y1, Y2;
Associate the entire training set to G, C_1 to A, C_2 to B;
Randomly initialise population for each island;
while not stopping do
    Run evolution for each island;
    Migrate individuals between islands;
end
return the best individual for each cluster;
```

1) Evolution for Island Models: We explain the proposed island model using Fig 1. Our island model consists of 7 islands where each island is associated with a cluster of TOPTW problem instances. The islands G, A and B evolve policies using training instances from clusters which remain the same throughout the evolutionary process. Island G always uses the entire training set. Islands A and B use the first-level two scenario clusters $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. Islands X1, X2, Y1 and Y2, on the other hand, may change their training instances after each n_p generations. Algorithms 2 and 3 present the evolutionary processes in these islands.

For islands G, A and B, the evolutionary process is the almost same as standard GPHH. The only difference is that after each generation, the island sends and receives some individuals to and from other islands using the migration strategies, which will be introduced next. For islands X1, X2, Y1 and Y2, the evolutionary process is different from that of the islands G, A and B in that after every n_p generations, a new cluster of training instances is re-sampled by Thompson sampling (Algorithm 4). Note that the clusters of the islands X1 and X2 come from the clusters of the island A, while the clusters of the islands Y1 and Y2 come from the clusters of the island B. This way, the island pairs $\{$X1, X2$\}$ and $\{$Y1, Y2$\}$ associate with dissimilar problem clusters.

Algorithm 2: Evolution for Islands G, A, B

```
Input: C_x, x ∈ {G, A, B}
Output: the best individual for the island
for g ← 1 : N_G do
    Sample a stochastic TOPTW instance I ∈ C_x;
    Run the g^th generation of GPHH on I;
    Receive/Send individuals using migration strategies;
end
return the best individual;
```

2) Migration Strategies: Migration strategies play a major role in the performance of the island models [23]. A migration strategy defines the individuals to be sent to another destination island, the generation at which it should start the migration, and the frequency of migration. The purpose of the migration strategies in the context of this work is to enable knowledge

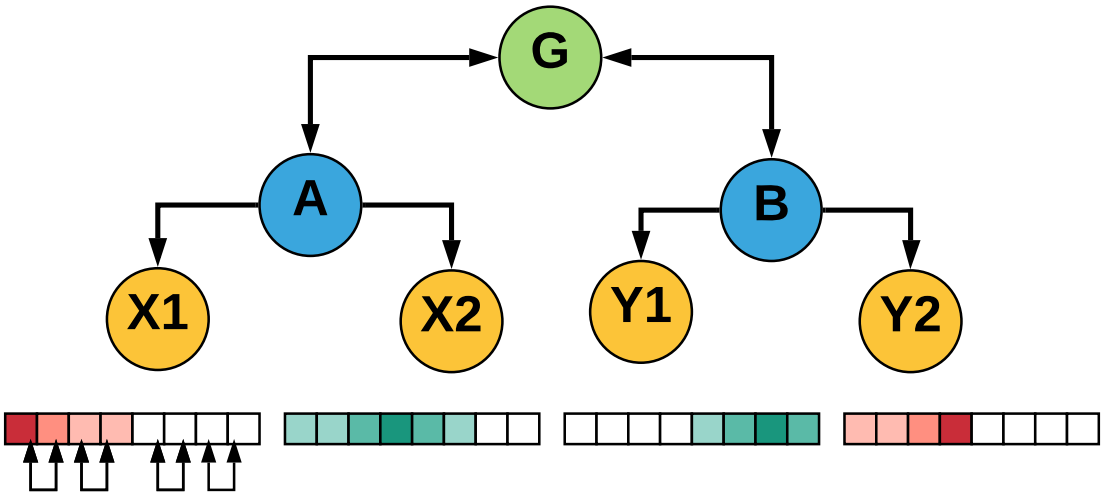

Fig. 1. Illustration of the island model structure.

Algorithm 3: Evolution for islands X1, X2, Y1 and Y2

Input: $\mathcal{C}_x,\ x \in \{\texttt{X1}, \texttt{X2}, \texttt{Y1}, \texttt{Y2}\}$
Output: the best individual for the island
for $g \leftarrow 1 : N_G$ **do**
  if $g \% n_p = 0$ **then**
    $\mathcal{C}_x \leftarrow$ Algorithm 4 (Thompson Sampling);
  Sample a stochastic TOPTW instance $\mathcal{I} \in \mathcal{C}_x$;
  Run g^{th} generation of GPHH using $\mathcal{I}$;
  Receive/Send individuals using migration strategies;
end
return the best individual;

Algorithm 4: Thompson Sampling

Input: $\{\omega_{z1}, \omega_{z2}, \omega_{\texttt{G}}\},\ \texttt{z} \in \{\texttt{X}, \texttt{Y}\}$
Output: The sampled cluster $\mathcal{C}_t^z$
if $g \leq n_p$ **then**
  return a randomly sampled cluster $\mathcal{C}_t^z$;
end
Let $\mathcal{C}_z^1$ be the currently used cluster in island Z1;
Let $\mathcal{C}_z^2$ be the currently used cluster in island Z2;
Randomly sample a set of p instances $\mathcal{I}_{z1}$ from $\mathcal{C}_z^1$;
Randomly sample a set of p instances $\mathcal{I}_{z2}$ from $\mathcal{C}_z^2$;
$val \leftarrow \frac{s(\omega_{z1},\mathcal{I}_{z1}) - s(\omega_{z2},\mathcal{I}_{z1})}{s(\omega_{\texttt{G}},\mathcal{I}_{z1})} + \frac{s(\omega_{z2},\mathcal{I}_{z2}) - s(\omega_{z1},\mathcal{I}_{z2})}{s(\omega_{\texttt{G}},\mathcal{I}_{z2})}$;
Update β corresponding to $\mathcal{C}_z^k$ with val;
Update β corresponding to pair of $\mathcal{C}_z^k$ with $val/2$;
Sample from all posterior β distributions and choose the cluster $\mathcal{C}_t^z$ corresponding to maximum value;
return $\mathcal{C}_t^z$;

sharing between the islands toward the goal of effective multi-tasking.

The direction of migration is shown in Fig. 1. Since island G evolves policies using the entire training set, it sends individuals to both islands A and B, which evolve policies using instances from the first-level clusters. Since we want the islands X1, X2, Y1 and Y2 to evolve policies which are specific to scenarios, the migration strategy of these islands ensure that the individuals from these islands are not shared among each other.

More formally, we define a policy $\mathcal{M}_{\overrightarrow{I_1,I_2}}$ from island I_1 to I_2 to be a triplet $\langle$start generation, frequency, #individuals to send$\rangle$. With this definition $\mathcal{M}_{\overrightarrow{I_1,I_2}}$ is different from $\mathcal{M}_{\overrightarrow{I_2,I_1}}$. The individuals to be migrated are selected based on elitism, i.e, a proportion of fittest individual(s) are chosen from the population for migration. In this work, apart from the direction of migration the values of each of the elements of a triplet in a migration strategy is the same for all islands.

3) Thompson Sampling: Now we explain our method to select appropriate clusters of instances for the islands X1, X2, Y1 and Y2. In Fig. 1, just below each of the four islands a set of rectangular boxes are shown. Each of these boxes corresponds to a cluster which is obtained by iteratively applying K-means for more levels (Section III-A). Moreover, each pair of these clusters are obtained from the same parent cluster resulting in pairs; the pairing is illustrated using the set of boxes under island X1.

The input to the Algorithm 4 is a triplet of policies corresponding to either $\{\texttt{X1}, \texttt{X2}, \texttt{G}\}$ or $\{\texttt{Y1}, \texttt{Y2}, \texttt{G}\}$. These are the best individuals from the island at that stage of evolution. During the initial stages of evolutionary process, a random cluster $\mathcal{C}_t^z$ is chosen when $g \leq n_p$.

Without loss of generality, assuming that the current island for which sampling is being performed is X1, and its corresponding pair is X2. The current clusters associated with X1 and X2 are C_z^1 and C_z^2 respectively. A set of instances is randomly sampled from each cluster. Then, the val is calculated based on the performance of the best individuals of X1, X2 and G on different problem clusters (line 8). The mathematical formula calculates the relative performance of the individuals if migrated to the paired island. In the formula,

$s(\omega_{\mathtt{z1}}, \mathcal{I}_{\mathtt{z1}})$ is the score values obtained using policy $\omega_{\mathtt{z1}}$ applied to the instances in $\mathcal{I}_{\mathtt{z1}}$. A higher val indicates that the evolved policies are more specific to certain problem scenarios. The score values corresponding to $\omega_{\mathtt{G}}$ are meant for normalization.

Lines 9–11 are adapted from a β-Thompson sampling algorithm [28]. Thompson sampling is a heuristic from the multi-armed bandit framework for choosing actions while facing the exploration versus exploitation dilemma. A β-Thompson sampling associates a prior beta distribution with each of the agents. From each of these beta distributions a sample is obtained and the agent whose corresponding sample value is maximum is chosen to perform the action. Depending upon the reward from the action, the beta distribution parameters are updated. For the next round, the updated beta distributions are considered. More detailes can be found in [28].

In line with the description of Thompson sampling above, each of the candidate clusters of instances is associated with a beta distribution. When a cluster is chosen, a new pair of policies are obtained after a few generations of evolution which corresponds to the "action". The reward is calculated using the expression in line 8. This is used to update the parameters of beta distribution in lines 9–10. Since we expect a pair of cluster from the same parent to have similarities the beta parameters are also modified, though slightly, for the other cluster in the pair (line 10). This is illustrated in Fig. 1 with different shades in the rectangular boxes corresponding to possible beta distribution parameters. In line 11, the samples are obtained from the posterior distributions and the cluster associated with the maximum value is selected as output of Algorithm 4.

C. Testing

At the end of the evolutionary process, we obtain a set of policies, $\Omega = \{\omega_{\mathtt{G}}, \omega_{\mathtt{A}}, \omega_{\mathtt{B}}, \omega_{\mathtt{X1}}, \omega_{\mathtt{X2}}, \omega_{\mathtt{Y1}}, \omega_{\mathtt{Y2}}\}$ with corresponding centroids of the clusters associated with each island. For island G, we can determine the centroid by averaging the other centroids.

When a stochastic TOPTW test instance is to be evaluated, the first step is to extract the feature vector of the instance, as explained in Section III-A. The policy with a centroid vector which is closest to this feature vector, based on the Euclidean distance, is selected to solve the test instance.

IV. Experiment Design

In order to create a data set which reflects the real world, we took the existing data sets [29] and made some modifications into them. We replaced the location coordinates, durations and score values from the dataset with new values. We considered 3 different normal distributions each with following parameters for the location coordinates, duration and score as follows $\{\{24, 10\}, \{44, 10\}, \{70, 10\}\}$, $\{\{24, 10\}, \{44, 10\}, \{70, 10\}\}$ and $\{\{24, 10\}, \{44, 10\}, \{70, 10\}\}$. Each pair of numbers correspond to the mean and standard deviation parameters. We generated our training set using combinations of these distributions. The uncertainty in duration is set with $\sigma = 0.2$ as in [3].

For the hierarchical K-means clustering approach, the number of levels chosen were such that the total number of clusters in the final level is 32. This means that each of the islands $\{\mathtt{X1}, \mathtt{X2}, \mathtt{Y1}, \mathtt{Y2}\}$ is associated with 8 clusters in the Thompson sampling routine of Algorithm 4. For Thompson sampling the parameters for all the prior beta distributions are $\{2, 3\}$, which were obtained by observing the values of the rewards obtained by evaluating the expression in line 8 of Algorithm 4.

The GPHH algorithm and the meta-algorithm used to solve the TOPTW problem are same as in [3]. The parameters and terminal set are presented in Tables II and III. The function set of GP is $\{+, -, \times, /, \min, \max\}$, where the division operator returns 1 if divided by 0. Through the selection of these parameters, we have ensured that the computational budget for every single run of BasicGP for a TOPTW instance is approximately equal to a single run of our proposed method, namely Island-EMT, which evolves policies for all TOPTW instances. In other words, if we have N training instances, then the number of runs required by the BasicGP is N, but for multitasking a single run produces a set of policies for all instances. Therefore, the total computational cost associated with BasicGP to evolve policies is considerably higher. The parameters for the migration policies for the islands is $\langle 5, 5, 25\rangle$, that is, the migration starts at generation 5, and then occurs after every 5 generations. During the migration, the top 25 fittest individuals are migrated.

Since the expected score value is the objective value for a stochastic TOPTW problem, for testing we generate 500 samples from each test instances and calculate the average score value [3].

TABLE II
Parameter Settings

Parameter	BasicGP	Island-EMT
Pop. size	1024	600/island
Elitism	10 best ind.	6 best ind.
Tourn. sel. size	7	7
Maximal depth	8	8
Crossover rate	80%	80%
Mutation rate	15%	15%
Reproduction rate	5%	5%
# of generations	250	60

V. Results and Discussions

Table IV shows the mean and standard deviation of the total score obtained by the compared GPHH algorithms on the test sets of 30 randomly selected TOPTW. For each instance, each compared algorithm was run 30 times independently and the results are compared statistically. The total score obtained by using manually designed policies: Score-Over-Slack (SOS) and Score-Over-Time (SOT) are also provided

TABLE III
TERMINAL SET OF GP.

Notation	Description
SCORE	score of POI
DUR	visit duration
TO	time to open
TC	time to close
TA	time to arrive
TR	time to return to end point
TSV	time to start the visit of POI
TFV	time to finish the visit
SL	slack
RemT	remaining time budget

TABLE IV
THE SCORE OF THE COMPARED GPHH METHODS ON THE TEST SET OF THE 30 STOCHASTIC TOPTW INSTANCES. THE LAST COLUMN SHOWS A '+' IF THE PROPOSED METHOD OUTPERFORMS BASICGP. '-' AND '=' INDICATE WORSE AND SAME PERFORMANCE RESPECTIVELY. THESE RESULTS ARE BASED ON WILCOXON'S RANK SUM TEST WITH SIGNIFICANCE LEVEL $\alpha = 0.05$.

#Inst.	SOS	SOT	BasicGP		Island-EMT		Wil.
			mean	std	mean	std	test
1	574.70	896.73	1496.31	102.95	**1569.24**	85.79	+
2	793.39	1003.69	1914.57	102.93	1931.73	96.27	=
3	914.25	926.89	1492.93	76.16	**1668.22**	124.86	+
4	3237.27	2986.79	**7218.20**	166.96	6977.19	310.65	−
5	2985.79	3872.82	7232.02	280.30	7221.90	245.74	=
6	3810.60	4331.86	7856.48	189.76	7832.24	242.25	=
7	2137.52	3026.68	6338.71	155.98	6287.37	275.31	=
8	3272.59	3030.28	7616.86	418.76	7750.26	211.60	=
9	2684.72	3269.41	5732.50	201.84	5627.20	316.31	=
10	1619.34	3334.14	5085.29	248.99	5132.47	166.20	=
11	3790.69	4914.20	7711.39	417.21	**8243.34**	232.89	+
12	3236.11	5818.99	8313.17	538.26	8465.07	193.01	=
13	4206.68	6102.90	8955.82	235.81	8856.88	325.81	=
14	462.37	1259.15	1771.86	38.36	**1822.60**	99.12	+
15	963.52	1249.47	2123.35	135.02	**2384.58**	132.87	+
16	2992.65	4292.83	6674.96	481.88	6653.18	368.63	=
17	2090.23	5513.75	7540.82	160.57	7508.22	259.95	=
18	2276.89	5349.10	7533.25	225.56	7500.23	313.88	=
19	2894.11	5614.68	7941.65	299.62	8043.24	314.73	=
20	791.94	860.32	**2331.81**	83.59	2147.73	118.63	−
21	779.33	891.01	1603.84	77.86	1580.59	86.13	=
22	404.31	1038.16	1513.59	73.29	**1561.95**	68.99	+
23	995.54	1246.68	**1839.45**	109.96	1727.29	103.36	−
24	484.59	1187.50	1343.81	72.93	**1418.42**	98.19	+
25	2074.39	3457.87	6552.24	200.95	6488.89	214.35	=
26	1797.63	3354.11	6560.33	248.01	6498.08	211.17	=
27	3025.04	4315.53	6998.16	171.69	6865.27	411.96	=
28	3060.19	3505.94	7061.87	173.57	7029.32	243.71	=
29	644.15	887.31	1655.91	77.89	**1738.04**	66.97	+
30	825.14	1481.88	2439.19	140.32	2449.03	125.96	=

for comparison. As expected, both the GPHH approaches outperform the manually designed policies.

For both the algorithms, BasicGP and Island-EMT, the mean and standard deviation metrics are shown. Out of 30 test instances, Island-EMT outperforms BasicGP in 8 instances. In 3 of these instances BasicGP does better. On the remaining instances the performance of both the algorithms is similar. To summarize, as $[\mathbf{win} - draw - loss]$:

$$[\mathbf{8} - 19 - 3]$$

We can observe that the performance of the proposed approach is better than the BasicGP. In light of the fact that the computational cost of BasicGP for evolving a policy for a single instance is the same as that for evolving a set of policies for all scenarios by the proposed multitasking approach this improvement in the result is very significant.

Essentially, through the migration mechanism of the island model, we have achieved knowledge sharing across the independent tasks of the island. Moreover, the choice of instances for the training in respective islands was effectively handled using hierarchical K-means clustering and Thompson sampling. The hierarchy of clusters enabled the island model to automatically design policies which span from more general to more specialized. The Thompson sampling method ensured that the specialized policies are suitable for different trip design scenarios.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, a multitasking approach to automatically evolve decision making policies for stochastic TOPTW problem using GPHH is developed. Since the proposed approach is based on island models, the knowledge sharing mechanism of multitasking is simpler than the existing works in literature. The experimental results have shown that the proposed algorithm can achieve no worse (and sometimes even better) solutions than the baseline GP approach without multitasking. Furthermore, the proposed algorithm can evolve the policies for a wide range of problem scenarios (30 in our experiment) simultaneously, while the baseline GP has to evolve for each of them one-by-one. In this sense, the proposed algorithm has a dramatically better efficiency than the baseline GP. This clearly demonstrates the effectiveness of using multitasking techniques and island models to solve a range of related problem domains.

In future, we would consider more realistic problems for trip design with time dependent score values and stochastic travel times and explore better island model topologies for such scenarios. With the success of this approach, We would also like to explore the island model toward developing transfer learning approaches to evolve effective policies for new scenarios with minimal additional computational cost.

REFERENCES

[1] K. H. Lim, J. Chan, C. Leckie, and S. Karunasekera, "Personalized tour recommendation based on user interests and points of interest visit durations," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[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] Y. Mei and M. Zhang, "Genetic programming hyper-heuristic for stochastic team orienteering problem with time windows," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–8.

[4] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 193–208, 2014.

[5] 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.

[6] A. Gupta, Y.-S. Ong, and L. Feng, "Multifactorial evolution: toward evolutionary multitasking," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 343–357, 2016.

[7] B. Da, Y.-S. Ong, L. Feng, A. Qin, A. Gupta, Z. Zhu, C.-K. Ting, K. Tang, and X. Yao, "Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metric, and baseline results," *arXiv preprint arXiv:1706.03470*, 2017.

[8] J. Ding, C. Yang, Y. Jin, and T. Chai, "Generalized multi-tasking for evolutionary optimization of expensive problems," *IEEE Transactions on Evolutionary Computation*, 2017.

[9] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013.

[10] J. Park, Y. Mei, S. Nguyen, G. Chen, and M. Zhang, "An investigation of ensemble combination schemes for genetic programming based hyper-heuristic approaches to dynamic job shop scheduling," *Applied Soft Computing*, vol. 63, pp. 72–86, 2018.

[11] B. L. Golden, L. Levy, and R. Vohra, "The orienteering problem," *Naval Research Logistics (NRL)*, vol. 34, no. 3, pp. 307–318, 1987.

[12] 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.

[13] A. Gunawan, H. C. Lau, and K. Lu, "An iterated local search algorithm for solving the orienteering problem with time windows," in *European conference on evolutionary computation in combinatorial optimization*. Springer, 2015, pp. 61–73.

[14] ——, "Sails: hybrid algorithm for the team orienteering problem with time windows," 2015.

[15] 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.

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

[17] 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.

[18] Y.-S. Ong and A. Gupta, "Evolutionary multitasking: a computer science view of cognitive multitasking," *Cognitive Computation*, vol. 8, no. 2, pp. 125–142, 2016.

[19] L. Feng, L. Zhou, J. Zhong, A. Gupta, Y.-S. Ong, K.-C. Tan, and A. Qin, "Evolutionary multitasking via explicit autoencoding," *IEEE transactions on cybernetics*, vol. 49, no. 9, pp. 3457–3470, 2018.

[20] M. Gong, Z. Tang, H. Li, and J. Zhang, "Evolutionary multitasking with dynamic resource allocating strategy," *IEEE Transactions on Evolutionary Computation*, 2019.

[21] J. Kacprzyk and W. Pedrycz, *Springer handbook of computational intelligence*. Springer, 2015.

[22] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: a survey," *ACM Computing Surveys (CSUR)*, vol. 45, no. 3, p. 35, 2013.

[23] K. Nowak, D. Izzo, and D. Hennes, "Injection, saturation and feedback in meta-heuristic interactions," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2015, pp. 1167–1174.

[24] A. R. Bertels and D. R. Tauritz, "Why asynchronous parallel evolution is the future of hyper-heuristics: A cdcl sat solver case study," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*. ACM, 2016, pp. 1359–1365.

[25] D. Karunakaran, Y. Mei, G. Chen, and M. Zhang, "Sampling heuristics for multi-objective dynamic job shop scheduling using island based parallel genetic programming," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2018, pp. 347–359.

[26] R. Hashimoto, H. Ishibuchi, N. Masuyama, and Y. Nojima, "Analysis of evolutionary multi-tasking as an island model," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 2018, pp. 1894–1897.

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

[28] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, Z. Wen *et al.*, "A tutorial on thompson sampling," *Foundations and Trends® in Machine Learning*, vol. 11, no. 1, pp. 1–96, 2018.

[29] G. Righini and M. Salani, "Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming," *Computers & operations research*, vol. 36, no. 4, pp. 1191–1203, 2009.