Memory-based Multi-Population Genetic Learning for Dynamic Shortest Path Problems

1st Yiya Diao
School of Automation
China University of Geosciences
Wuhan, 430074, China
diaoyiyacug@163.com

2nd Changhe Li*
School of Automation
China University of Geosciences
Hubei Key Laboratory of Advanced Control and Intelligent Automation for Complex Systems
Wuhan, 430074, China
changhe.lw@gmail.com

3rd Sanyou Zeng
School of Electrical and Mechanical
China University of Geosciences
Wuhan, 430074, China
sanyouzeng@gmail.com

4th Michalis Mavrovouniotis
KIOS Research and Innovation Center of Excellence and Department of Electrical and Computer Engineering
University of Cyprus, Nicosia, 2109, Cyprus.
mavrovouniotis.michalis@ucy.ac.cy

5th Shengxiang Yang
School of Computer Science and Informatics
De Montfort University
Leicester, LE1 9BH, United Kingdom
syang@dmu.ac.uk

***Abstract*—This paper proposes a general algorithm framework for solving dynamic sequence optimization problems (DSOPs). The framework adapts a novel genetic learning (GL) algorithm to dynamic environments via a clustering-based multi-population strategy with a memory scheme, namely, multi-population GL (MPGL). The framework is instantiated for a 3D dynamic shortest path problem, which is developed in this paper. Experimental comparison studies show that MPGL is able to quickly adapt to new environments and it outperforms several ant colony optimization variants.**

***Index Terms*—dynamic shortest path, dynamic sequence optimization, genetic learning, ant colony optimization, clustering-based multi-population,**

I. Introduction

The dynamic sequence optimization problem (DSOP) has a wide range of applications in practice, e.g., the vehicle navigation problem and the workshop scheduling problem. The SOP, a typical combinatorial optimization problem, is to find the best order for a number of given items [1]. Examples of classical SOPs are the travel salesman problem (TSP), the vehicle routing problem (VRP), and the shortest path problem (SPP), etc.

The dynamic TSP (DTSP) is a typical benchmark for DSOPs, where typical types of changes are changing the sequence number of nodes in the graph [2] or changing the distance between nodes [3]. The first type of changes can keep the global optimum unchanged, which is helpful to analyse the performance of an algorithm in dynamic environments but hardly seen in real-world applications. The second type of change is more practical but the global optimum is unknown. It makes hard to analyse the performance of an algorithm. To alleviate this issue, we propose a 3D dynamic shortest path problem (DSPP) as a benchmark problem for DSOPs. This problem is simple and can show some real-world characteristics which are challenging to solve. The optimal solution can be easily obtained by Dijkstra's algorithm. These two features make it suitable to test an algorithm's performance.

The ant colony optimization algorithm (ACO), which stimulates the ants' foraging behaviors, is the mostly used algorithm for DSOPs [4]. To make ACO effective in dynamic environments, several issues should be addressed: (1) The pheromone trail contains previous information which may mislead the search and make the algorithm hard to escape from the old optimum when the environment changes; (2) The heuristic information is needed but it is not always available or hard to obtain in practice; (3) The parameters of ACO are hard to tune. In this paper, a genetic learning (GL) [5] algoirthm is used to solve DSOPs. The GL was initially proposed in [5], updated in [6] for the static TSP and was later applied to the static SPP in [7]. GL calculates the probability only based on the current population, which enables the population to adapt to dynamic environments quickly and does not require any heuristic information.

When an algorithm converges, it is hard to find the new optimum. Dynamism handling strategies are needed to help an algorithm to improve the diversity loss issue to find new optima. This paper proposes a multi-population strategy, where each sub-population is responsible for searching for one near-optimum. To improve the search ability of each sub-population, we use a memory scheme in each sub-population

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61673355, in part by the Fundamental Research Funds for the Central Universities, China University of Geosciences (Wuhan) under Grant 201705, in part by the Hubei Provincial Natural Science Foundation of China under Grant 2015CFA010, and in part by the 111 project under Grant B17040.

*The corresponding author.

to make use of the information from all the individuals. Also, this algorithm does not depend on the detection of the change and can quick track the changing global optimum.

The rest of this paper is organized as follows. Section II describes the dynamic handling strategies that are mainly applied to ACO. Section III defines the DSPP proposed in this paper. Section IV gives the details of the proposed algorithm. Section V presents the experimental studies. Finally, conclusions and future work are presented in Section VI.

II. Related work

The basic idea of ACO is a probabilistic technique based on sampling. It was inspired by the food foraging behaviors of ants. Ants will release trail pheromones when they look for foods and the pheromone trail will guide them to find a shorter path to the food source. ACO was first published in [8]. Ant colony system (ACS) [9] and MAX-MIN ant system (MMAS) [10] are two widely used variants. Angus and Hendtlass [11] showed that the conventional ACO algorithm is able to solve a small DTSP instance. To extend ACO to dynamic environments, the loss of diversity issue should be addressed.

The goal for dynamic optimization is not only to find the global optimum but also to track the changing optimum over time. When a traditional evolutionary algorithm has converged, it hardly finds new good solutions due to the loss of diversity. Dynamism handling strategies are needed to improve diversity. These strategies can be divided into five classes: adaptive, diversity, prediction, memory and multi-population strategies [4], [12].

Adaptive strategies adapt the parameters or operators to the changing environments. For example, the pheromone evaporation rate ρ for ACO can be adaptive tuned. In [13], ρ increases when the algorithm stagnates to increase the global search ability of the algorithm. Later self-adaptive ρ was proposed in [14], where ρ is optimized during the evolutionary process. The two parameters α and β, which control the weight of pheromone trails and heuristic information, can also be adaptively tuned in ACO [15].

Diversity strategies aim to increase diversity after a change is detected or to maintain diversity during the whole run. Eyckelhof and Snoek [16] introduced a shaking method to smoothen the pheromone trails when an environmental change is detected. Immigrants strategy [3] in ACO is a popular diversity strategy, which introduces immigrant ants, either using random solutions (random immigrant ACO, RIACO) or the best solutions from the previous environment (elitism-based immigrants ACO, EIACO), to deposit pheromone trails. Guo [17] used the crowding distance to select the global best or the local best particle in particle swarm optimization (PSO) to solve dynamic multi-objective vehicle routing problems.

Predictive strategies predict the changes and take action in advance. In [18], the Kalman filters were used to predict the movement of the global optimum. However, predictive strategies work only when changes exhibit some predictable patterns.

Memory strategies store and reuse useful information, which is quite useful for periodical or recurrent changes. Population-based ACO (PACO) [19] uses a memory of limited size to record the best solution from each generation with a first-in-first-out policy. Memory-based immigrants [3] combine the memory strategy and immigrants with a memory of the population, which includes the best solutions and immigrant solutions. In [20], the memory was organized as a tree structure to reduce the computational cost and increase its capacity.

Multi-population strategies use each sub-population to track different local optimum so that when the global optimum change between different local areas, the algorithm can quickly re-locate it. One of the key issue for the multi-population is how to decide the number of sub-populations. A bottom-up clustering method was used in [21] to determine the original population size and then adapt it to the changing environment during the evolutionary process. In [22], each sub-population uses a separate pheromone table, which is helpful to improve to the diversity of ACO. Bu [23] combined the multi-population strategies particle and PSO to solve dynamic constrained optimization problems.

Recently, there are some strategies that hybridize two or more strategies introduced above. Luo [24] combined the memory strategies and the multi-population strategies. In [24], the retrieved memory individuals were classified according to their fitness values and their distances to the closest population and different operations were applied to them.

III. 3D dynamic shortest path problem

A 3D DSPP is proposed in this section. It has two advantages to be a benchmark: (1) The problem is able to simulate the progressive change of the practical problems; (2) The global optimum is always available during changes.

A. Design of the DSPP

The design of the static 3D SPP was proposed in [7], where a map is divided into $W \times H$ grids (W and H are the map width and height). Each grid can be seen as an item in the DSOP. The problem is defined by:

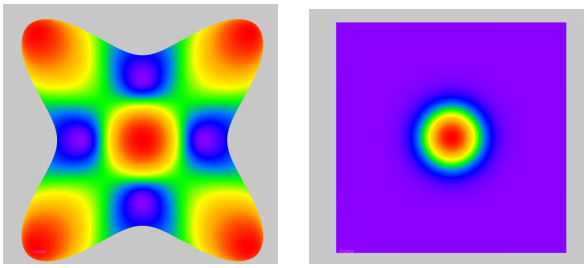
$$\begin{array}{ll} \min & f(\vec{\mathbf{x}}(t)) = \sum_{i=0}^{k-1} d(\mathbf{x}_i, \mathbf{x}_{i+1}), \\ s.t. & \mathbf{x}_{i+1} \in N_{\mathbf{x}_i},\ \mathbf{x}_0 = \mathbf{s},\ \mathbf{x}_k = \mathbf{e} \end{array} \quad (1)$$

where $\vec{\mathbf{x}} = \{\mathbf{x}_0, \mathbf{x}_1 ...., \mathbf{x}_k\}$ is a solution ($\mathbf{x}_i = (x_i, y_i, z_i(t))$, x_i, y_i, $z_i(t)$ are the abscissa value, the ordinate value and the height for $\mathbf{x_i}$, respectively), $\mathbf{s}$ and $\mathbf{e}$ are the starting grid and the ending grid, respectively, $d(\mathbf{x}_{i-1}, \mathbf{x}_i)$ is the Euclidean distance between the two grids $\mathbf{x}_{i-1}$ and $\mathbf{x}_i$. The next move from grid $\mathbf{x}_i$ must be in the neighborhood of $\mathbf{x}_i$ (written as $N_{\mathbf{x}_i}$), which is defined by:

$$N_{\mathbf{x}_i} = \{\mathbf{x}_j | \|\mathbf{x}_j - \mathbf{x}_i\|_1 = |x_j - x_i| + |y_j - y_i| \leq 1\} \quad (2)$$

The generation function for the dynamic map at time t is defined by:

$$z(x, y, t) = \sum_{i=1}^{5} \phi_i(x, y)\, g_i(t) + \sum_{j=1}^{4} \varphi_j(x, y)\, g_j(t), \quad (3)$$



ϕ φ

Fig. 1. Two basic functions for generating changes, where ϕ is for global change and φ is for local change.

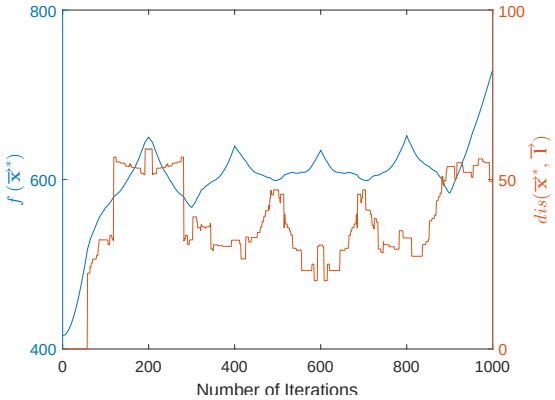


Fig. 2. The figure shows how the dynamic map changes over 1,000 generations, where the blue line shows the change of length of global optimum and the red line shows the distance between the global optimum and the direct path from $\mathbf{s}$ to $\mathbf{e}$.

where $z(x, y, t)$ is the height of the grid (x, y) of the map at the time t, ϕ and φ are two basic functions for generating two different types of changes (introduced later), g is used to change the relative influence of ϕ and φ at time t.

The change in this paper has two types: global change and local change. The global change means that the whole environment changes while the local change means that only a part of the environment changes. Fig. 1 shows that when ϕ is added to a flat map, the height of almost all the grids on the map will be changed and thus ϕ is used to generate the global change. With φ, only the height of the grid near the center will be changed, thus φ is used to generate the local change.

The formulas of the two functions are:

$$\phi(x, y) = h * \cos(x + c)\cos(y + c), \tag{4}$$

$$\varphi(x, y) = h * \exp\left((x + c)^2 - (y + c)^2\right), \tag{5}$$

where h determines the relative height of the grids on the map, (c, c) is a random center grid of the functions ϕ and φ on the map, c is a uniformly distributed random number $c \in [0, \min(W, H)]$.

To make the map changing smoothly, g is introduced to gradually change the influence of ϕ and φ, which is defined by:

$$g(t) = \begin{cases} 0 & t < t_1 \\ \frac{t - t_1}{2 * (t_2 - t_1)} & t_1 \le t < \frac{t_2 - t_1}{2} \\ \frac{t_2 - t}{2 * (t_2 - t_1)} & \frac{t_2 - t_1}{2} < t \le t_2 \\ 0 & t_2 < t \end{cases} \tag{6}$$

where t_1 and t_2 are two uniformly distributed random numbers. From the time t_1 to $\frac{t_2 - t_1}{2}$, the influence of the function changes from zero to one, and from the time $\frac{t_2 - t_1}{2}$ to t_2, the influence changes from one to zero.

B. Characteristic of the DSPP

Fig. 2 shows how the dynamic map changes over 1,000 generations. The blue line shows the length change of the global optimum and the red line shows the distance between the global optimum and the path directly from $\mathbf{s}$ to $\mathbf{e}$. Generally, the larger the length of the global optimum is, the more difficult the problem is. From the blue line, we can see that the problem is changing continuously and smoothly.

The red line is the distance between the global optimal solution (written as $\vec{\mathbf{x}}^*$) and the path direct from $\mathbf{s}$ to $\mathbf{e}$ (written as $\vec{\mathbf{l}}$). The distance between two solutions $\vec{\mathbf{x}}$ and $\vec{\mathbf{y}}$ is defined by:

$$dis(\vec{\mathbf{x}}, \vec{\mathbf{y}}) = \max_{i=1}^{k_1} \left(\min_{j=1}^{k_2} dis(\mathbf{x}_i, \mathbf{y}_j) \right), \tag{7}$$

where $\vec{\mathbf{x}} = \{\mathbf{x}_0, \mathbf{x}_1 ...., \mathbf{x}_{k_1}\}$ and $\vec{\mathbf{y}} = \{\mathbf{y}_0, \mathbf{y}_1 ...., \mathbf{y}_{k_2}\}$ are two solutions, both of which contain a path from $\mathbf{s}$ to $\mathbf{e}$. This distance is the maximum vertical distance between two solutions on the map.

If the distance is around a large value, $\vec{\mathbf{x}}^*$ would be far away from the relatively simple solution $\vec{\mathbf{l}}$ and thus the global optimum is more difficult to find. This means that the test problem remains a certain difficulty during the whole change. We can see that although the problem is continuously changing, the change of the global optimum may be abrupt, indicating that the global optimum may switch to another area.

IV. Multi-population Genetic Learning

A general framework for DSOPs is proposed in this section. It consists of three parts: (1) the GL algorithm; (2) a clustering-based multi-population strategy; (3) a local memory strategy. Based on statistical learning, this algorithm uses individuals from the current population to calculate the probability matrix $\mathbf{P}$ used in GL so that the population can quickly adapt to the changing environment. To avoid the situation that the algorithm falls into a local optimum when the environment changes, we use a clustering-based multi-population strategy to improve the diversity of the algorithm. Also, to remember some useful information used to guide the local search, a local memory is used for each sub-population.

A. Framework of MPGL

Algorithm 1 is the framework of MPGL. To maintain diversity during the search, we divide the whole population into sub-populations and the probability matrix $\mathbf{P}$ is calculated only based on the best individuals from each sub-population.

Algorithm 1 Multi-population genetic learning

```
Randomly initialize a population X;
while termination criteria not satisfied do
    Divide the population X into sub-populations by Algorithm 3;
    Update the local memory for each sub-population using Eqs. (12)-(14);
    Set the best individual and the local memory for each sub-population as B;
    Calculate the probability matrix P based on B using Eqs. (8)-(11);
    for all individuals in each sub-population do
        Create a new solution with P and its local memory according to Algorithm 2;
    end for
    Put all individuals in X;
end while
```

Algorithm 2 Genetic learning

Input:
Individual $\vec{\mathbf{x}_i}$, the probability matrix $\mathbf{P}$ and the local memory;
Output:
A new solution for $\vec{\mathbf{x}_i}$;

```
while the construction of x_i is not completed do
    if rand() ≤ ρ then
        Select the next feasible item based on P by roulette selection;
    else
        Select the next feasible item based on the local memory;
    end if
end while
```

To help a sub-population search for the near optimum nearby, a local memory scheme is used to record the information from its individuals. In MPGL, the probability matrix works to maintain diversity for global search and the local memory scheme focus on local search. With the two strategies, GL can balance global search and local search during the evolutionary process.

1) Genetic learning: Algorithm 2 shows the construction of a solution in GL [6]. GL is based on solution construction strategies. It uses the parameter ρ to balance local search and global search (whether to learn from $\mathbf{P}$ or the local memory);

The gene recorded in GL can be regarded as an ordered pair of two items π_{ij} in DSOPs. Its main idea is to use optimized information to estimate the probability for each gene and use this probability to guide the evolution. The probability matrix $\mathbf{P}$ is used to record the weighted frequency of π_{ij} visited by the historical best solution found by each individual.

The probability of π_{ij} is calculated as follows:

$$p_{ij} = \frac{w_{ij}}{\sum w_{il}}, \forall l \in N_j, \tag{8}$$

where N_j is the set of neighbor items of item j, p_{ij} means the probability of each individual moves from the item i to the item j, w_{ij} is the weight of π_{ij}.

The objective value for all individuals in the population is normalized by:

$$\dot{f}_i = \frac{f_{max} - f_i + 1}{f_{max} - f_{min} + 1}, \tag{9}$$

where $\dot{f}_i$ is the normalized objective value for the individual i of objective value f_i, f_{max} and f_{min} are the maximum and the minimum objective values of all the individuals in the MPGL population, respectively.

A weight ω_i is assigned to the i^{th} individual based on its normalized objective value $\dot{f}_i$ by:

$$\omega_i = \frac{1}{1 + \mathrm{e}^{-\dot{f}_i}}, \tag{10}$$

The weighted frequency of π_{ij} is calculated by:

$$w_{ij} = \sum_{k=1}^{S} \omega_k * o_{ij}, o_{ij} = \begin{cases} 1, \pi_{ij} \in \vec{\mathbf{x}}_k \\ 0, else \end{cases} \tag{11}$$

where S is the population size, $\vec{\mathbf{x}}_k$ is the solution created by the individual k. Note that, we only use the current population to update $\mathbf{P}$ to enable GL to adapt to a new environment quickly.

2) Multi-population strategy: Although GL can quickly adapt its probability matrix $\mathbf{P}$ to the changing environment, it still suffers from the issue that when it converges, it can hardly escape from the old optimum. Dynamic handling strategies are needed to help GL to escape from old optimum. The multi-population strategy inspired by [21] is designed in this paper.

In the multi-population strategy, how to generate multiple populations is crucial, because the number of populations should be different at the different time. For example, in the proposed benchmark, current peaks may be gone and new peaks may be formed as changes occur on the map. As a result, local optima may disappear or appear as time goes on.

This paper uses a density-based top-down clustering idea [25] to self-adaptively cluster individuals into multiple sub-populations. There are two key ideas in [25]: (1) The points with high density are more likely to be the center of a cluster; (2) The center of each cluster should be far away from each other. Similarly, in a population, we can use the two rules to locate the near-optimum. A solution with a high fitness is more likely to be a near-optimum and such solutions should be far away from each other.

Algorithm 3 shows the framework of the clustering process. Individuals are sorted based on fitness values to make sure that individuals with high fitness are more likely to be chosen as near-optima. We merge an individual with its nearest individual of higher fitness value. By this way, each near-optimum is far away from each other. α is a parameter to separate two sub-populations and can be regarded as the minimum distance between two solutions in two different sub-populations.

Algorithm 3 Multi-population strategy

Input:
The whole population $\mathbf{X}$;
Output:
The sub-populations;
Sort the whole population based on the fitness value from the largest to the smallest;
for each solution $\vec{\mathbf{x}}$ in the whole population **do**
  for each solution $\vec{\mathbf{y}}$ whose fitness value is smaller than $\vec{\mathbf{x}}$ **do**
    if $dis(\vec{\mathbf{x}}, \vec{\mathbf{y}})$ ¡ α **then**
      $\vec{\mathbf{x}}$ and $\vec{\mathbf{y}}$ belong to the same sub-population;
      break;
    end if
  end for
  if $\vec{\mathbf{x}}$ does not belong to any sub-populations **then**
    $\vec{\mathbf{x}}$ is recorded as the center of a new sub-population;
  end if
end for

3) Local memory: In the GL, the local memory is originally used for local search. To make use of the information from all the individuals, a local memory is proposed for each sub-population to record the information from all the individuals in the sub-population and each local memory works for local search.

For π_{ij} in DSOPs, we remember 'the best performance' as a local memory for each sub-population. For each π_{ij} there must be the best solution that containing it. With this knowledge, we can regard the objective value of the best solution associated with each π_{ij} as a local memory for each sub-population and store them in a matrix ($\mathbf{M}$). Each element m_{ij}^k records the objective value associated with π_{ij} for the sub-population k.

When a new solution $\vec{\mathbf{x}}$ with objective value $f_{\vec{\mathbf{x}}}$ enters the sub-population k, the memory matrix $\mathbf{M}$ is updated according to the following equation:

$$m_{ij}^k = \min(m_{ij}^k, f(\vec{\mathbf{x}})), \forall \pi_{ij} \in \vec{\mathbf{x}} \tag{12}$$

When two sub-populations a and b merge into one sub-population c, their associated local memory $\mathbf{M}^a$ and $\mathbf{M}^b$ will also merge into one local memory $\mathbf{M}^c$ by:

$$m_{ij}^c = \min(m_{ij}^a, m_{ij}^b) \tag{13}$$

As this is a dynamic problem, the associated best objective value for each π_{ij} would not stay the same as the problem changes and old $\mathbf{M}$ may transfer the wrong information to the next generation. To address this issue, inspired by the evaporation mechanism in ACO, this paper increases the objective value in $\mathbf{M}$ as follows:

$$m_{ij} = m_{ij} * (1.0 + \beta), \tag{14}$$

where $0 < \beta < 1$ is a const parameter to determine the uncertainty of the information from the last generation. The information is more uncertain as β increases and the effect of old knowledge will become weaker as time goes. All the memory matrix $\mathbf{M}$ is updated at the end of each generation.

Note that, there are some differences between the pheromone trail in ACO and the local memory in MPGL. The pheromone trail in ACO needs to balance global search and local search, and the evaporation speed is very important to the search. The local memory in GL just focus on local search for a local optimum. With Eqs. (12) - (14), when the algorithm finds a newly better solution, $\mathbf{M}$ can be quickly updated with the objective value.

Algorithm 4 Local update $\mathbf{L}$

Input a solution $\vec{\mathbf{x}} = \{\mathbf{x}_0, \mathbf{x}_1 ...., \mathbf{x}_n\}$ and a distance matrix ;
d = 0;
// d is the shortest distance from the current grid to $\mathbf{e}$
for $i = n - 1$; $i \geq 1$; $i = i - 1$ **do**
  // To visit $\vec{\mathbf{x}}$ from $\mathbf{e}$ to $\mathbf{s}$;
  $d = d + d(\mathbf{x}_i, \mathbf{x}_{i+1})$;
  if $d < l_{\mathbf{x}_i}$ **then**
    $l_{\mathbf{x}_i} = d$
  else
    $d = l_{\mathbf{x}_i}$
  end if
end for

B. Implementation on DSSP

With the domain knowledge on the DSSP, a distance vector $\mathbf{L}$ is proposed to replace the local memory $\mathbf{M}$. The vector $\mathbf{L}$ records the historical shortest distance to $\mathbf{e}$ for each visited grid on the map, i.e., each elements l_i^k records the shortest distance for the grid i to $\mathbf{e}$ for sub-population k.

Algorithm 4 works as a local search operator in MPGL and tries to improve the local search ability for MPGL. The distance from each grid to e would be changing, and the distance vector $\mathbf{L}$ should be updated with a new solution as Eq. (12).

In the multi-population strategy, when the best individuals from different sub-populations are merged to the same sub-population, their associated local memory, that is $\mathbf{L}$ also should be merged by Eq. (13). $\mathbf{L}$ is updated as in Eq. (14) at each generation. In MPGL, when an individual learns from the local memory, it always selects the next feasible item with the smallest l_i.

V. Experimental studies

We carry out two sets of experimental studies. The first shows the effect of different components of MPGL on DSPP. The second compares the MPGL with other ACO variants.

A. Experimental setup

For the DSPP, the map size $W = H = 50$ was used. The population size $S = 60$. The starting point $\mathbf{s} = (0, 0)$ is at the left bottom grid and the ending point $\mathbf{e} = (W - 1, H - 1)$ is at the left top grid on the map. A Change occurs after every

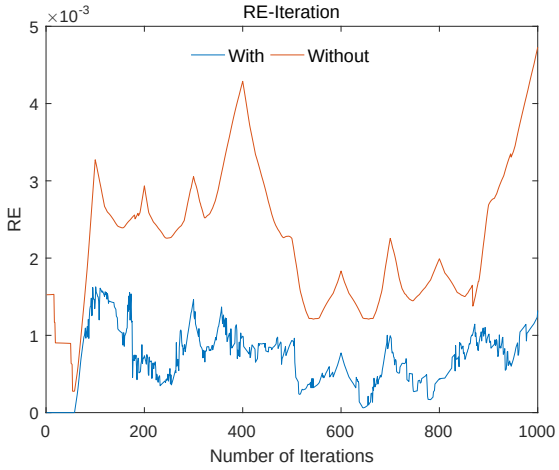


Fig. 3. Comparison of the performance between GL with and without the local memory over a single run.

NC evaluations. We assume that the change in this problem is progressive and we set the maximum change to one unit. The maximum change means the maximum difference of the height at each point in the map over two successive change.

In MPGL, there are three parameters:(1) The ratio of global search ρ was recommended in [7]; (2) The minimum distance α for population clustering was set to the average distance between all individuals and the best individual of the sub-population it belongs to. (3) The uncertainty factor β was set to 0.1 for achieving the best overall performance.

The relative error (RE) is used to evaluate the performance.

$$RE = (f(\vec{\mathbf{x}}^b) - f(\vec{\mathbf{x}}^*))/f(\vec{\mathbf{x}}^*), \tag{15}$$

where $\vec{\mathbf{x}}^b$ and $\vec{\mathbf{x}}^*$ are the best solution found by the algorithm and the global optimum, respectively.

B. Component analysis

To analyse the relationship between the local memory or the multi-population strategy with the environmental changes, the map changes at every iteration of the MPGL.

1) Effect of the local memory on MPGL for DSPP: In Fig. 3, the blue line and the red line show the change in the RE of MPGL with and without local memory over a single run of 1,000 generations. From the figure, we can see that L can obviously help optimize the path and improve the accuracy of the algorithm. In the process of path construction, L will be updated gradually and thus help improve the solution. During the path construction, the algorithm always selects a relatively close grid to make a relative good solution. This can avoid the potentially bad selection, and thus the constructed path has a smaller relative error.

2) Effect of the multi-population strategy for DSPP: In the top figure from Fig. 4, the blue line and the red lines show the change in the RE of MPGL with and without the multi-population strategy, respectively, over 1000 generations. MPGL can adapt to the change more quickly. The multi-population strategy makes the algorithm more stable. For

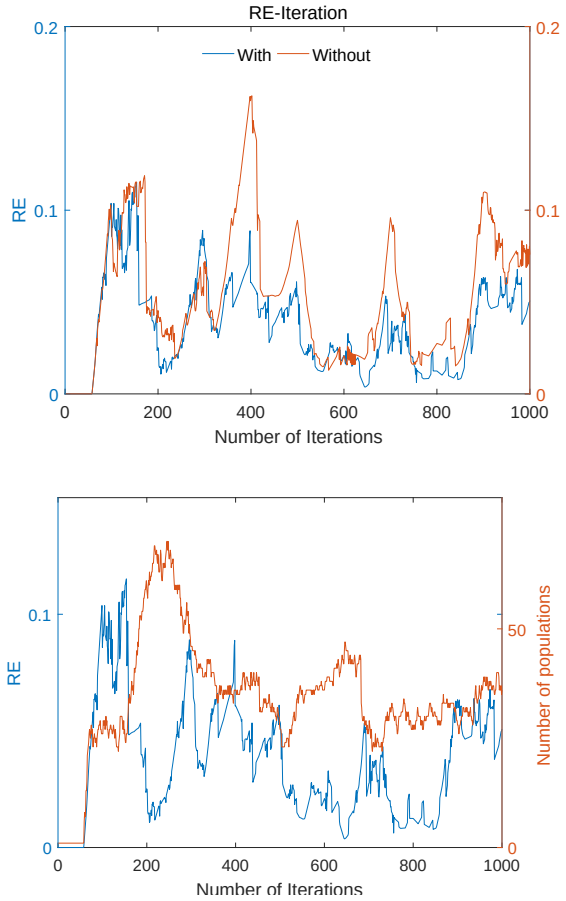


Fig. 4. Comparison of the performance between MPGL with and without the multi-population strategy over a single run.

example, around the 400^{th} generation, there is a huge rise in the RE of MPGL without the multi-population strategy while the rise of MPGL is relatively smaller. It indicates that the algorithm can effectively capture the local optimal when the global optimum switches to other areas.

In the bottom figure from Fig. 4, the red line shows the change in the number of populations over 1000 generations. It can be seen that the number of sub-populations changes as the environment changes. For example, around the 400^{th} generation, the RE gradually increases. However, before the RE increases, the number of populations already increases. It indicates that the multi-population can track environmental changes. Before the global optimal moves to the other area, the environment already changes and the number of populations increases to help the algorithm capture the near-optima. When the global optimal move between different local optimal, the algorithm can quickly capture it.

C. Comparison with other algorithms

For a fair comparison, the local memory without problem domain knowledge $\mathbf{M}$ is used in MPGL and we do not use any local search based on the problem domain knowledge because different ACO variants need different local search operators, which have a significant impact on the performance

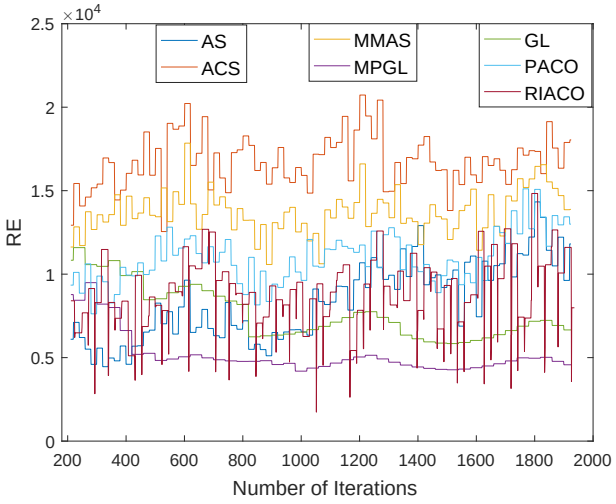


Fig. 5. Comparison between MPGL, GL and ACO variants, where $W = H = 50, NC = 10$ over 30 independent runs.

TABLE I
AVERAGE RE AND STANDARD DEVIATION OF RE OVER 1000 EVALUATIONS FOR ALL ALGORITHM OVER 30 INDEPENDENT TESTS, WHERE W AND H ARE THE WIDTH AND THE HEIGHT OF THE MAP, RESPECTIVELY.

Algorithm	W=50,H=50		
	NC=2	NC=10	NC=20
MPGL	10.45 ± 1.70	$\mathbf{4.18 \pm 0.78}$	$\mathbf{2.98 \pm 0.37}$
GL	$\mathbf{7.10 \pm 0.70}$	4.65 ± 0.50	3.65 ± 0.41
PACO	28.14 ± 1.31	26.67 ± 2.83	24.70 ± 1.57
RIACO	14.63 ± 2.22	14.93 ± 3.08	14.57 ± 3.17
MMAS	25.54 ± 2.79	23.19 ± 1.87	21.38 ± 2.12
AS	13.27 ± 1.53	15.24 ± 2.08	16.90 ± 2.11
ACS	29.12 ± 2.52	29.36 ± 2.18	26.70 ± 2.68
Algorithm	W=100,H=100		
	NC=2	NC=10	NC=20
MPGL	20.64 ± 0.75	$\mathbf{7.71 \pm 0.50}$	$\mathbf{6.72 \pm 0.57}$
GL	$\mathbf{15.94 \pm 1.38}$	9.45 ± 0.45	7.66 ± 0.46
PACO	36.68 ± 1.24	40.35 ± 3.78	37.49 ± 1.82
RIACO	22.84 ± 4.18	25.80 ± 4.44	25.26 ± 4.45
MMAS	47.75 ± 2.02	43.25 ± 2.20	38.62 ± 2.68
AS	20.4 ± 3.49	20.03 ± 3.38	22.41 ± 3.45
ACS	54.33 ± 3.01	46.74 ± 2.60	46.56 ± 2.52

of an algorithm. For example, MMAS with a proper local search operator even outperforms the state-of-art ACO variants for DTSP [4]. Also, we would like to see the potential for MPGL to solve other DSOPs, such as DTSP, DVRP through a simple comparison with ACO variants. All the parameters for ACO variants were recommended by their authors and the parameters for MPGL and GL is the same in the DSPP.

From Fig. 5, we can see that GL and MPGL outperform the ACO variants on the DSPP. Without the local memory $\mathbf{M}$ and multi-population strategy, GL still works better than RIACO. This is because they calculate the probability matrix $\mathbf{P}$ based on the current population so that they can quickly adapt to the changing environment. RIACO sometimes may find the best solution because it has certain probability of finding new good solutions, but its performance is quite unstable because of the randomness.

From Table I, we can see that when the frequency of the environmental changes is fast, GL is better than MPGL, maybe the local memory $\mathbf{M}$ in MPGL may mislead the individuals. However, when the speed of the environmental changes becomes slow, MPGL performs much better than GL. The average performances of GL and MPGL are much better than other ACO variants.

VI. CONCLUSION

This paper proposes a multi-population genetic learning framework, namely, MPGL, for DSOPs. This framework combines the GL and a clustering-based multi-population strategy with a memory scheme. The multi-population strategy works to maintain the diversity and the memory scheme helps to improve the local search ability.

From the experimental studies, we can see both MPGL and GL outperform the ACO variants. It is mainly because that the GL uses individuals from the current population to calculates the probability so that it can adapt to the changing environment quickly. With the multi-population strategy, MPGL can maintain the diversity during the whole search. Compared with RIACO, we also find that with some randomness, algorithms may have a better performance.

The relationship between the number of populations and the environmental changes is not clear and we will take a further study on it. We will also apply it to the other types of DSOPs, e.g., DTSP and DVRP, with some appropriate local search operators.

REFERENCES

[1] S. Raggl, A. Beham, V. A. Hauder, S. Wagner, and M. Affenzeller, "Discrete real-world problems in a black-box optimization benchmark," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2018, Kyoto, Japan, July 15-19, 2018*, 2018, pp. 1745–1752.

[2] L. Kang, A. Zhou, R. I. McKay, Y. Li, and Z. Kang, "Benchmarking algorithms for dynamic travelling salesman problems," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2004, 19-23 June 2004, Portland, OR, USA*, 2004, pp. 1286–1292.

[3] M. Mavrovouniotis and S. Yang, "Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors," *Appl. Soft Comput.*, vol. 13, no. 10, pp. 4023–4037, 2013.

[4] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: Algorithms and applications," *Swarm and Evolutionary Computation*, vol. 33, pp. 1–17, 2017.

[5] Y. Xia, C. Li, and S. Zeng, "Three new heuristic strategies for solving travelling salesman problem," in *Advances in Swarm Intelligence - 5th International Conference, ICSI 2014, Hefei, China, October 17-20, 2014, Proceedings, Part I*, 2014, pp. 181–188.

[6] Y. Xia and C. Li, "Memory-based statistical learning for the travelling salesman problem," in *IEEE Congress on Evolutionary Computation, CEC 2016, Vancouver, BC, Canada, July 24-29, 2016*, 2016, pp. 2935–2941.

[7] Y. Diao, C. Li, Y. Ma, J. Wang, and X. Zhou, "A probabilistic learning algorithm for the shortest path problem," in *Simulated Evolution and Learning - 11th International Conference, SEAL 2017, Shenzhen, China, November 10-13, 2017, Proceedings*, 2017, pp. 631–643.

[8] M. Dorigo, "Optimization, learning and natural algorithms," *Thesis Politecnico Di Milano Italy*, 1992.

[9] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[10] T. Stützle and H. Hoos, "Improvements on the ant-system: Introducing the max-min ant system," in *Artificial Neural Nets and Genetic Algorithms*. Vienna: Springer Vienna, 1998, pp. 245–249.

[11] D. Angus and T. Hendtlass, "Ant colony optimisation applied to a dynamically changing problem," in *Developments in Applied Artificial Intelligence, 15th International Conference on Industrial and Engineering, Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2002, Cairns, Australia, June 17-20, 2002, Proceedings*, 2002, pp. 618–627.

[12] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 6, pp. 1–24, 2012.

[13] M. Mavrovouniotis and S. Yang, "Adapting the pheromone evaporation rate in dynamic routing problems," in *Applications of Evolutionary Computation - 16th European Conference, EvoApplications 2013, Vienna, Austria, April 3-5, 2013. Proceedings*, 2013, pp. 606–615.

[14] ——, "Ant colony optimization with self-adaptive evaporation rate in dynamic environments," in *2014 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments, CIDUE 2014, Orlando, FL, USA, December 9-12, 2014*, 2014, pp. 47–54.

[15] M. Ankerl and A. Hämmerle, "Applying ant colony optimisation to dynamic pickup and delivery," in *Computer Aided Systems Theory - EUROCAST 2009, 12th International Conference, Las Palmas de Gran Canaria, Spain, February 15-20, 2009, Revised Selected Papers*, 2009, pp. 721–728.

[16] C. J. Eyckelhof and M. Snoek, "Ant systems for a dynamic TSP," in *Ant Algorithms, Third International Workshop, ANTS 2002, Brussels, Belgium, September 12-14, 2002, Proceedings*, 2002, pp. 88–99.

[17] Y. Guo, J. Cheng, S. Luo, D. Gong, and Y. Xue, "Robust dynamic multi-objective vehicle routing optimization method," *IEEE/ACM Trans. Comput. Biology Bioinform.*, vol. 15, no. 6, pp. 1891–1903, 2018.

[18] C. Rossi, M. Abderrahim, and J. C. Díaz, "Tracking moving optima using kalman-based predictions," *Evolutionary Computation*, vol. 16, no. 1, pp. 1–30, 2008.

[19] M. Guntsch and M. Middendorf, "Applying population based ACO to dynamic optimization problems," in *Ant Algorithms, Third International Workshop, ANTS 2002, Brussels, Belgium, September 12-14, 2002, Proceedings*, 2002, pp. 111–122.

[20] T. Zhu, W. Luo, and L. Yue, "Dynamic optimization facilitated by the memory tree," *Soft Comput.*, vol. 19, no. 3, pp. 547–566, 2015.

[21] C. Li, T. T. Nguyen, M. Yang, M. Mavrovouniotis, and S. Yang, "An adaptive multipopulation framework for locating and tracking multiple optima," *IEEE Trans. Evolutionary Computation*, vol. 20, no. 4, pp. 590–605, 2016.

[22] M. Mavrovouniotis, S. Yang, and X. Yao, "Multi-colony ant algorithms for the dynamic travelling salesman problem," in *2014 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments, CIDUE 2014, Orlando, FL, USA, December 9-12, 2014*, 2014, pp. 9–16.

[23] C. Bu, W. Luo, and L. Yue, "Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies," *IEEE Trans. Evolutionary Computation*, vol. 21, no. 1, pp. 14–33, 2017.

[24] W. Luo, J. Sun, C. Bu, and H. Liang, "Species-based particle swarm optimizer enhanced by memory for dynamic optimization," *Appl. Soft Comput.*, vol. 47, pp. 130–140, 2016.

[25] R. Alex and L. Alessandro, "Machine learning. clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, p. 1492, 2014.