Niching Evolutionary Computation With a Priori Estimate for Solving Multi-Solution Traveling Salesman Problem

Ting Huang[1], Yue-Jiao Gong[1,*], Xiaoyan Li[2], Xiao-Min Hu[3], and Jun Zhang[4]
[1] School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
[2] College of Computer and Information Engineering, Henan Normal University, Xinxiang, China
[3] School of Computers, Guangdong University of Technology, Guangzhou, China
[4] Hanyang University, Seoul, South Korea.
[*] Corresponding author: gongyuejiao@gmail.com

***Abstract*—Multi-solution traveling salesman problem has diverse optimal routes. To obtain the different optimal solutions, current researches incorporate evolutionary algorithms with niching techniques. However, without knowing the problem characteristics in advance, the algorithms suffer from difficulties in setting the niching parameters. To address this issue, we utilize a graph neural network to predict a prior knowledge about the optimal tour length. Then, with the prior estimate, the niche radius can be adjusted for a specific problem. We develop a niching evolutionary algorithm that utilizes the calculated niche radius to identify diverse niches. Besides, a selective local search strategy is embedded into the algorithm to enhance the search capability. The experimental results show that the proposed algorithm has a competitive performance over the comparison algorithms on the benchmark suite.**

***Index Terms*—Graph neural network, multi-solution traveling salesman problem, niching evolutionary algorithm, prior estimate knowledge.**

I. Introduction

Multi-solution traveling salesman problem (MSTSP) has multiple solutions with the same minimum tour cost. It means that a traveler can traverse all the cities with several shortest tour programs. Fig. 1 shows a problem instance of the MSTSP benchmark suite in [1]. The classical TSPs have been widely applied in engineering and academic issues, such as vehicle routing problems [2]–[4], the coding order optimization problem [5], and the in-port ship routing and scheduling problem [6]. In the literature, most researches focus on algorithm design for a global optimum. In practice, there are generally more than one promising solutions in TSP (or more exactly, MSTSP). Diverse alternatives can provide flexible choices for a decision maker, quick response in the uncertain environment, and traffic balance between different routes.

This work was supported in part by the Key Project of Science and Technology Innovation 2030, Ministry of Science and Technology of China, under Grant 2018AAA0101300, and in part by the National Natural Science Foundation of China under Grants. 61873095 and U1701267, in part by the Guangdong Natural Science Foundation Research Team Project under Grant 2018B030312003, and in part by the Guangdong-Hong Kong Joint Innovation Platform Project under Grant 2018B050502006.

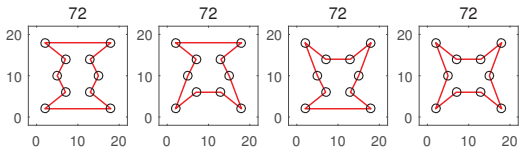

Fig. 1. Four optimal solutions of MSTSP9.

Considering the advantages of offering multiple optimal solutions, it is highly desired to design an effective optimization algorithm for MSTSP. However, so far, only a few publications are related to MSTSP [1], [7]–[10]. One challenge of tackling an MSTSP lies in the unknown of the cost and number of the optimal routes, which are significant to the algorithm design. The existing MSTSP optimization algorithms are the evolutionary algorithms (EAs) equipped with niching techniques. The EAs search for promising solutions [11], [12], while the niching technique enables parallel location of latent optima by dividing the population into multiple sub-populations. The niching strategy commonly involves a sensitive niching parameter that determines how to identify the promising region and divide the population. The population-based ant colony optimization (P-ACO) algorithms introduced in [8] incorporated two niching strategies, i.e., fitness sharing and crowding, both with fixed parameter settings. The niche-based ant colony system (NACS) [9] remained or created a niche according to a niche radius and an acceptable threshold. The neighborhood-based genetic algorithm (NGA) [1] adopted a less sensitive niching parameter, the neighbourhood size. In the niching memetic algorithm (NMA) [10], the neighborhood size is calculated in an adaptive manner. However, so far, the niching parameter is problem-dependent and difficult to appropriately set without the prior knowledge of the specific instance.

In order to design a more reliable and effective niching approach, more information about an MSTSP instance could be valuable. Recently, a graph neural network model is proposed by Prates *et al.* [13] to solve a TSP variant, decision TSP

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

(dTSP). The model is trained to yield "yes" or "no" answer whether the route with cost C exists in the instance. Utilizing the proposed model, named GNN-TSP, we can predict the optimal cost of TSP/MSTSP. Based on the prior knowledge, it is possible to design an effective niching algorithm with problem-specific information.

In this paper, we propose a niching evolutionary algorithm with a prior estimate (NEA-PE) to search for optimal solutions of MSTSP. We first predict the minimum tour cost of an MSTSP with the model GNN-TSP. Subsequently, the prior estimate is fed into a niching EA for assisting identifying the niches. Then the population are enhanced by using selective local search and undergone with reproduction. Finally, the elites of the population are distinguished and provided.

The rest of the paper is organized as follows. Section II introduces the existing MSTSP optimization algorithms and GNN. Section III describes the proposed algorithm in details. Experimental results are demonstrated and analysed in Section IV. Finally, we reach conclusions in Section V.

II. Background

A. Multi-Solution Optimization Algorithms for MSTSP

MSTSP is an NP-hard problem as TSP but is more difficult to be solved. MSTSP requires the solvers to obtain overall optimal tour programs. Consequently, the optimizers for MSTSP should not only possess a high search efficiency for satisfactory solutions but also maintain a good population diversity for different solutions. With limited search resources, it is difficult to achieve both of the two targets.

The existing multi-solution optimization algorithms for dealing with MSTSP include the multi-chromosomal cramping based genetic algorithm (MCC-GA) [8], the niche-based ant colony system (NACS) [9], the neighborhood-based genetic algorithm (NGA) [1], and the niching memetic algorithm (NMA) [10]. In MCC-GA [8], a chromosome is encoded with a set of solutions. With the number of included solutions, the problem space increases exponentially. Besides, without the prior knowledge of the optima size, it is difficult to find out the whole optimal solution set. Differently, NACS [9] utilizes several pheromone matrixes to locate different promising candidate solutions. When ants detect a good and distinct solution, a new pheromone matrix is created based on the solution. The ants on the different pheromone matrixes are favored to locate different optima. NACS confronts the problem of setting a suitable niching parameter to identify the solution for a new niche. Similarly, NGA [1] aggregates neighbors into one group and performs the basic reproduction operations of GA. The evolution and selection are restricted within groups, so as the population locate diverse optima. However, NGA encounters the problem of slow convergence and difficulty in setting an appropriate neighborhood size. NMA [10] develops a niching memetic algorithm with adaptive neighborhood strategy. The adaptive neighborhood strategy formulates the neighborhood size according to the population state. However, the formulation is built with the relative information of the evolving population. Therefore, how to appropriately set the niching parameter is still an open issue.

Fig. 2. The pipeline of the problem solving: the problem instance MSTSP, the prediction model GNN-TSP, and the problem solver NEA-PE.

B. Graph Neural Network Model for a TSP Variant

Recently, searchers have noticed that the studies of neural network (NN) are limited in regular Euclidean data like images and text, but NN cannot be applied for non-Euclidean data, such as graph [14]. The graph neural network (GNN), first proposed by Scarselli *et al.* [15], becomes the mainstream of studying on the non-Euclidean data. TSP is a classical graph problem with non-Euclidean structure. The problem can be modeled with vertex (cities) and edges (roads). In 2019, Prates *et al.* [13] proposed a GNN-TSP model to tackle a TSP variant, the decision TSP (dTSP). dTSP requires a binary solution to indicate whether the problem has a route with a specific cost C. In the proposed GNN-TSP, both of the weights on the edges and the predefined cost C are fed to a multilayer perceptron to obtain an edge embedding. Each vertex embedding updates itself with a recurrent neural network and communicates with adjacent vertexes along the edges. After the iterations of massage exchange and information update, the logit probability is yielded to give the answer.

III. The Proposed Algorithm

In order to obtain more information of MSTSP, we first utilize the GNN-TSP model to predict the optimal cost of the problem instance. Note that the model can be trained beforehand. Subsequently, the prior estimate cost L_{est} is fed into the niching EA, named NEA-PE. The pipeline of the problem solving is illustrated in Fig. 2. In the following parts, the GNN model and the proposed NEA-PE will be introduced in details.

A. The Prediction Model: GNN-TSP

GNN-TSP in [13] is essentially a message-passing algorithm on graph. The vertices communicate with adjacency vertices along the edges. Moreover, the weights on edges are fed into the model for learning both relational and numerical information. In this way, we can train a model on TSP/MSTSP.

In the experiment, the authors observed that the answer for the question with a optimal tour cost C^* can reach at around 50% prediction value. Based on this assumption, a binary search with GNN-TSP was developed to predict the cost of TSP's optimum. The essence is that GNN-TSP will give a more positive answer when the trial value C is larger than the optimal cost C^*, and vice versa. Based on the rule, we can approach to C^* using a binary search. To be specific, we first initialize the lower/upper boundary of an MSTSP, i.e., L_{lb} or L_{ub}, with the sum of the first city size smallest/largest weight (or length) values. Next, we set the estimate cost L_{est}

Algorithm 1 Binary search with GNN-TSP

Input: An MSTSP model G
Output: Estimate cost L_{est}

L_{lb}, $L_{\text{ub}} \leftarrow$ initialBound(G)
$L_{\text{est}} \leftarrow (L_{\text{lb}} + L_{\text{ub}})/2$
while $L_{\text{lb}} < L_{\text{est}} \times (1-\delta)$ or $L_{\text{est}} \times (1+\delta) < L_{\text{ub}}$ **do**
  if GNN-TSP(G, L_{est}) $< 50\%$ **then**
    $L_{\text{lb}} \leftarrow L_{\text{est}}$
  else
    $L_{\text{ub}} \leftarrow L_{\text{est}}$
  end if
  $L_{\text{est}} = (L_{\text{lb}} + L_{\text{ub}})/2$
end while

Algorithm 2 NEA-PE

Input: An MSTSP instance, and the prior estimate cost L_{est}
Output: A representative solution set *S*

Parent $\leftarrow$ Initialize(*NP*)
Evaluate(*Parent*)
Archive $\leftarrow \{\}$
while the termination condition is not reached **do**
  Niches $\leftarrow$ NichingWithPior(*Parent*, L_{est}) // **Algorithm 3**
  ENiches $\leftarrow$ SelectiveLS(*Niches*, *Archive*) // **Algorithm 4**
  Offspring $\leftarrow$ Reproduction(*ENiches*)
  Evaluate(*Offspring*)
  Parent $\leftarrow$ Selection(*Offspring*)
end while
S $\leftarrow$ EliteIdentif (*Parent*, *Archive*)

with the average of L_{lb} and L_{ub}. Then, we recurrently reset the L_{lb} or L_{ub} to L_{est} according to the prediction value given by GNN-TSP. Besides, L_{est} is updated with the average of the new boundaries. Until the termination condition is met, we obtain the final estimate L_{est}. The procedure is concluded in **Algorithm 1**.

B. The Problem Solver: NEA-PE

For NEA-PE, we adopt the genetic algorithm (GA) [16] as the baseline. The algorithm first initializes the population with *NP* chromosomes and then evaluates the initial population. Each individual in the population is a candidate solution for the problem. Next, the evolution iteration is called. The L_{est} predicted by GNN-TSP is adopted as a prior knowledge to calculate the niche radius. According to the calculated niching parameter, the population is partitioned into several niches (subpopulations). Then, a selective local search strategy is performed. Within each niche, we adopt the reproduction approaches and breed the offspring. The offspring are evaluated and adopted the environmental selection to provide the population in the next generation. The above process is iterated until the termination condition is satisfied. At the end, we identify and preserve the elite solutions from the final population for output. The process is outlined in **Algorithm 2**. In the following, we describe the subcomponents of NEA-PE in details.

1) Niching Strategy With Piror Estimate Cost: The niching strategy can maintain multiple promising candidates at simultaneous in the evolution. However, in the niching approach, the niche radius is a problem-dependent parameter, which affects the algorithm performance largely. By using the GNN-TSP, we can bring in the prior estimate information to calculate the niche radius.

Algorithm 3 NichingWithPior(Parent, L_{est})

L_{best}, $L_{\text{avg}} \leftarrow$ UpdateLength(*Parent*, L_{best})
while *Parent* $\neq \varnothing$ **do**
  Niches $\leftarrow \{\}$
  Seed $\leftarrow$ FindBest(*Parent*)
  $r \leftarrow$ CalNicheRadius(*Parent*, *Seed*, L_{est}) // using Eq. (1)
  sParent $\leftarrow$ SortByDist(*Parent*, *Seed*, "asc") // using Eq. (2)
  for each p in *sParent* **do**
    if $D(p, Seed) \leq r$ or $|Niches| < m$ **then**
      Niches $\leftarrow$ *Niches* $+ p$
    end if
  end for
  Parent $\leftarrow$ *Parent* $-$ *Niches*
  Niches $\leftarrow$ *Niches* $+$ *Niches*
end while
Parent $\leftarrow$ *Niches*

The method first updates the best-so-far length L_{best} and the average length L_{avg} in the population. Next, the niches are identified and grouped iteratively. To be specific, the best solution in the unprocessed population is regarded as the seed. Next, the niche radius r is calculated with the prior estimate length L_{est}. Then, the available individuals are sorted by the distance with the seed (using Eq. (2)), in an ascending order. In the sequence, the individuals whose distance falls in the niche radius r belong to the same niche of the seed. If the grouped member size is smaller than m, the top unsettled individuals will be supplemented to the niche. Subsequently, the niche is added to the mating set and the corresponding individuals are marked as "processed" in the current population. The value of m is set to 4 to ensure sufficient individuals to conduct the reproduction. The above procedure is concluded in **Algorithm 3**

Here, we introduce the two equations mentioned above. The niche radius measures the size of a "peak". The words "niche radius" and "peak" are brought from the concept of the continuous fitness landscape. There is a similar observation in the combinatorial optimization filed. Distinguishing different peaks facilitates the detection for distributed optima. However, the niche radius is difficult to set, as we are blind to the solution distribution of the problem. Bringing the prior knowledge, we can calculate the niche radius with

$$r = \frac{L_{\text{seed}} - L_{\text{est}}}{L_{\text{avg}} - L_{\text{best}}} \tag{1}$$

where L_{seed} is the length of a seed, L_{est} is the prior estimate length, L_{avg} is the average length of the population, and L_{best} is the best-so-far length. The numerator measures the difference for the seed towards the optimum and the denominator scales the difference. Note that the value of r is truncated in [0, 1]. Generally, the better of an individual is, the more likely it locates at top of a peak, and therefore the corresponding niche radius is smaller. For example, when an individual is an optimum, the value of r is 0 in theory. That is to say, the individual is at the peak and do not need any search

Algorithm 4 *ENiches* ← SelectiveLS(*Niches*, *Archive*)

```
count ← 0
ENiches ← {}
for each niche in Niches do
    p ← FindBest(niche − Archive)
    if count ≤ N/10 then
        Archive ← Archive + p
        Localsearch(p)
        count++
    end if
    randomshuffle(niche)
    ENiches ← ENiches + niche
end for
```

resource. In another case, when an individual is worse or equal to L_{avg}, the value of r may be 1. It may locates at a valley between hills. The rest unprocessed individuals will be grouped together to explore for more promising locations.

In the process of the niche identification, the distance between two solutions should be measured. Note that the solution of MSTSP is encoded as the permutation of cities. The distance between two solutions π_i and π_j is calculated as follows

$$D(\pi_i, \pi_j) = 1 - \frac{|\Phi(\pi_i) \cap \Phi(\pi_j)|}{N} \tag{2}$$

where the $\Phi(\pi_i)$ and $\Phi(\pi_j)$ denote the edge set of the solutions π_i and π_j, N is the city size, and $|\Phi(\pi_i) \cap \Phi(\pi_j)|$ is the number of the common edges of the solutions $\Phi(\pi_i)$ and $\Phi(\pi_j)$.

2) Selective Local Search Strategy for Niches: After the niches are identified and settled, we perform selective local search strategy to improve the search capability of niches. A 2-opt local search [17] is carried out. The 2-opt exchanges two edges of a solution. As the local search is effective for local optimum search but is a fitness-evaluation-consumptive operation, we selectively adopt the local search for the members of niches. Besides, we maintain an archive and a counter to record. The archive preserves the individuals that are to take local search to avoid the redundant implementation. The counter is adopted to limit the executions of local search operations. To be specific, for each niche, we find the best solution that does not exist in the archive, and then perform the local search. In each generation, the number of performing local search is limited within $N/10$ times, where N denotes the city size. Besides, the members of each niche are randomly shuffled. Note that we count one fitness evaluation when the 2-opt local search takes $N/4$ times, as one complete evaluation consumes N operations and one local search requires 4 operations. The selective local search is presented in **Algorithm 4**.

3) Reproduction Techniques: The reproduction is implemented within niches. GA contains two reproduction techniques: crossover and mutation. The crossover mates two chromosomes to breed two children. We adopt partially-mapped crossover [18]. Specifically, the genes between the exchange points are inherited from its direct parental chromosome directly; other slots take genes from another parental chromosome; genes are derived from the mapping relationship when the genes to be placed in the slot have already included in the current chromosome.

The mutation perturbs the chromosome to avoid the population being trapped into the local optimum. We randomly exchange two genes in the chromosome to perform the mutation.

4) Selection Approch: The parent from the same niche can breed similar children at a high probability, and the reproduction limits the search around the niche. The environmental selection is also limited within niches. Through this method, we can maintain multiple promising candidates in different subregions. For each child, we choose the closest parental chromosome in the niche to compare and survive the best one.

5) Elite Idetification Approch: When the evolution iteration stops, we obtain the final population set in niches. Note that the archive of the selective local search strategy is identified as a niche and is appended to the set. The obtained candidates are somewhere between the in-process and final solutions. The in-process solutions are inferior and redundancy to provide for a decision maker. Therefore, we refine the solutions with an elite identification approach. We distinguish the elite solutions on two occasions: (a) The length of the candidate solution is exactly L_{best}. (b) The candidate solution has a length shorter than L_{thr} and a minimum distance from other selected solutions larger than a threshold ϵ. The value of L_{thr} is set to $1.01 \times L_{\text{best}}$ and ϵ is set to 0.2. Through this way, we can preserve the elites and eliminate the inferiors from the final solution set.

TABLE I
MAXFES APPLIED FOR 2 RANGES OF TEST INSTANCES

Two ranges of test instances	MaxFEs
MSTSP1 - MSTSP12	6.00E+04
MSTSP13 - MSTSP25	1.20E+06

IV. EXPERIMENTS

A. Experimental Setup

The proposed NEA-PE is compared with state-of-the-art MSTSP algorithms, i.e., the niche-based ant colony system (NACS) [9], the neighborhood-based genetic algorithm (NGA) [1], and the niching memetic algorithm (NMA) [10], and two discrete multimodal optimization algorithms, i.e., crowding GA (CGA) [19] and sharing GA (ShGA) [19]. The algorithms are tested on the 25 MSTSP benchmark suite in [1]. All the algorithms set the population size as 150 and terminate when the maximum fitness evaluations (MaxFEs) are exhausted. The MaxFEs of MSTSPs are listed in Table I. The parameters of the comparison algorithms are set according to their publications. The algorithms are independently executed 50 times. To measure the algorithm performance, we adopt two evaluation indicators, i.e., F_β and the diversity indicator (DI). For the obtained solution set of the algorithm, F_β score evaluates the quality and DI measures the diversity. The details of these two indicators can be referred to [1], [10].

TABLE II

F_β SCORE OF THE ALGORITHMS ON 25 MSTSPS OVER 50 RUNS. EACH RESULT IS ASSOCIATED WITH A SIGNIFICANT NOTATION IN THE PARENTHESIS. THE AVERAGE F_β VALUE AND THE OVERALL SIGNIFICANT RESULTS ARE SUMMARIZED IN THE LAST TWO ROWS.

F_β	NACS	NGA	CGA	ShGA	NMA	NEA-PE
MSTSP1	0.684(−)	0.973(−)	0.024(−)	0.026(−)	**1.000**(≈)	**1.000**
MSTSP2	0.804(−)	0.959(−)	0.030(−)	0.034(−)	**1.000**(≈)	**1.000**
MSTSP3	0.497(−)	0.935(−)	0.078(−)	0.110(−)	**1.000**(≈)	**1.000**
MSTSP4	0.724(−)	0.932(−)	0.034(−)	0.034(−)	**1.000**(≈)	**1.000**
MSTSP5	0.989(≈)	0.846(−)	0.017(−)	0.017(−)	**1.000**(≈)	**1.000**
MSTSP6	0.643(−)	0.877(−)	0.034(−)	0.034(−)	**1.000**(≈)	0.999
MSTSP7	0.125(−)	0.769(−)	0.261(−)	0.435(−)	0.923(−)	**0.992**
MSTSP8	0.137(−)	0.578(−)	0.337(−)	0.700(−)	0.772(−)	**0.861**
MSTSP9	0.768(−)	0.974(−)	0.034(−)	0.034(−)	**1.000**(≈)	**1.000**
MSTSP10	0.813(−)	0.969(−)	0.034(−)	0.034(−)	**1.000**(≈)	**1.000**
MSTSP11	0.459(−)	0.949(−)	0.071(−)	0.118(−)	**1.000**(≈)	**1.000**
MSTSP12	0.090(−)	0.331(+)	0.275(+)	**0.919**(+)	0.535(+)	0.247
MSTSP13	0.025(−)	0.096(−)	0.255(−)	0.003(−)	0.611(−)	**0.943**
MSTSP14	0.087(−)	0.172(−)	0.090(−)	0.000(−)	**0.883**(≈)	0.871
MSTSP15	0.004(−)	0.416(−)	0.219(−)	0.003(−)	0.732(−)	**0.904**
MSTSP16	0.000(−)	0.054(−)	0.211(−)	0.000(−)	0.554(−)	**0.973**
MSTSP17	0.000(−)	0.044(−)	0.044(−)	0.000(−)	0.605(−)	**0.910**
MSTSP18	0.000(−)	0.031(≈)	0.062(−)	0.000(−)	**0.571**(+)	0.113
MSTSP19	0.000(≈)	0.007(≈)	0.040(+)	0.000(≈)	**0.168**(+)	0.007
MSTSP20	0.000(−)	0.000(−)	0.015(−)	0.000(−)	0.165(≈)	**0.208**
MSTSP21	0.012(−)	0.000(−)	0.001(≈)	0.000(−)	0.023(≈)	**0.082**
MSTSP22	0.000(≈)	0.000(≈)	0.001(≈)	0.000(≈)	**0.013**(≈)	0.011
MSTSP23	0.000(≈)	0.000(≈)	0.000(≈)	0.000(≈)	0.016(≈)	**0.026**
MSTSP24	0.000(−)	0.000(−)	0.002(≈)	0.000(−)	0.010(≈)	**0.024**
MSTSP25	0.000(≈)	0.000(≈)	0.000(≈)	0.000(≈)	0.002(≈)	**0.005**
Avg.	0.274	0.437	0.087	0.100	0.623	**0.647**
−/≈/+	0/5/20	1/5/19	2/5/18	1/4/20	3/16/6	

* The figures in the format of (−/≈/+) indicate that the number of the comparison algorithm are significantly worse than, similar to, or significantly better than the proposed NEA-PE, respectively.

B. Comparisons With MSTSP Optimization Algorithms

1) F_β score: The algorithm with a larger F_β value indicates high solution quality. We carry out the comparison expeirment on the F_β score. Furthermore, a Wilcoxon rank-sum test with significant level 0.05 is conducted to validate the algorithm performance in terms of F_β. Table II summerizes the F_β values of NEA-PE and the comparison algorithms and their significant results. The best results are marked in bold. From the table, we can see that NEA-PE performs the best 19 out of 25. NEA-PE has the average F_β value at 0.647, followed by NMA with 0.623. The significant results indicate that NEA-PE significantly outperforms NACS, NGA, CGA, shGA, and NMA on 20, 19, 18, 20, and 6 out of 25. Overall, the experimental results in terms of F_β validate the compititive performance of NEA-PE.

2) DI: The algorithm with a higher DI values implys that it has more diverse solution set. The pseudo-color image in terms of DI is illustrated in Fig. 3. The figure suggests that CGA, NMA, and NEA-PE have an overall good diveristy. Particularly, NEA-PE achives the best DI on the MSTSP13-MSTSP17. NEA-PE is outstanding among the most comparison algorithms in terms of DI.

C. Investigation of the Prior Estimate Knowledge

The niche radius is calculated with the prior estimate cost L_{est} with the premise: the value of L_{est} predicted by GNN-TSP is equal to the minimum tour cost L^* of the MSTSP. However, GNN-TSP is a probabilistic model trained with data.

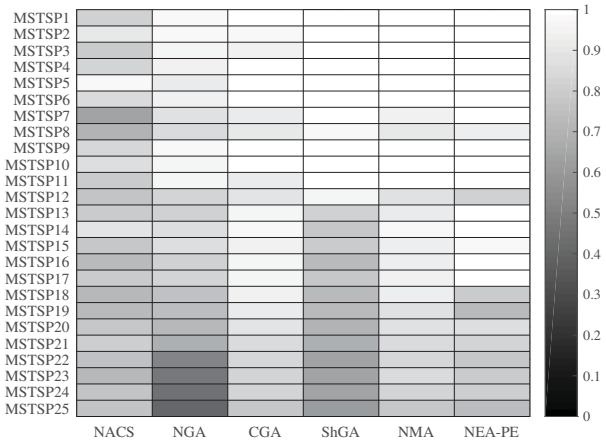

Fig. 3. The pseudo-color plot of 25 MSTSPs with the comparison algorithms and NEA-PE.

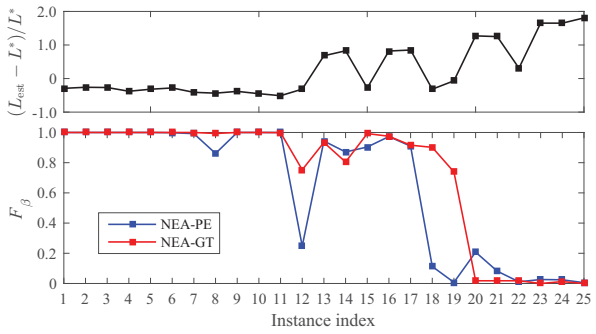

Fig. 4. Line chart of the deviation ratio (upper subgraph) and average F_β (lower subgraph) of NEA-PE and NEA-GT on each problem instance.

There is deviation on the prediction to the reality. In this part, we present the deviation of the prior estimate and investigate its influence. To conduct the comparison and analyze the influence of the deviation, we feed the ground-truth cost L^* to the proposed algorithm, which results in NEA-GT. In Fig. 4, we plot the line chart of the deviation ratio (calculated with $(L_{\text{est}} - L^*)/L^*$) on each MSTSP instance, and correspondingly illustrate the average F_β obtained by NEA-PE and NEA-GT. From the figure, we can see that GNN-TSP always undervalues the simple and geometry problems (MSTP1–MSTSP12) and overvalues the composite problems (MSTSP13-MSTSP25). In an overall perspective of the lower subgraph, NEA-GT outperforms or ties with NEA-PE on most MSTSP instances.

To make a further analysis, we study the specific problem instances. For example, for the MSTSP12, NEA-GT achieves a higher F_β than NEA-PE does. GNN-TSP predicts the route cost with -3.15 absolute deviation from the ground truth. The underestimation makes NEA-PE to increase the niche size, which results in a limited number of niches. Consequently, NEA-PE cannot maintain sufficient number of optima, which leads to a worse F_β value. As to another instance, MSTSP20, NEA-PE performs better than NEA-GT. GNN-TSP predicts the route cost with 1.27 absolute deviation ratio. MSTSP20

with 45 cities owns a relatively large problem space. The overestimation makes NEA-PE think that it finds optima wrongly and hence set a smaller niche size. The shrink of the search space improves the convergence ability, as it exploits in a smaller problem space. But generally, as can be observed in Fig. 4, a more accurate estimator can enhance the search capability of NEA-PE in most cases.

V. CONCLUSIONS

This paper develops a niching evolutionary algorithm incorporating a prior estimate knowledge to deal with MSTSPs. First, a GNN-TSP model predicts the prior optimal route cost for the problem. Then, the predicted information is fed into the niching evolutionary algorithm for the calculation of the niche size. According to the niche radius, we can identify the niches from population. Based on the niches, we further strengthen the search capability and distinguish the elites for the output.

In order to investigate the effectiveness of the proposed algorithm, NEA-PE is compared with the existing MSTSP optimization algorithms. The experimental results validate that NEA-PE outperforms the comparison algorithms in terms of solution quality and solution diversity. Furthermore, we study the influence of the prior estimate knowledge on the solution quality. The summarized results indicate that the algorithm favors a more accurate optimal length.

REFERENCES

[1] T. Huang, Y.-J. Gong, and J. Zhang, "Seeking multiple solutions of combinatorial optimization problems: A proof of principle study," in *Proc. IEEE Symposium Series Comput. Intell.* ACM, 2018, pp. 87–88.

[2] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 1, pp. 70–85, Jan 2017.

[3] J.-H. Wang, Y. Zhou, Y. Wang, J. Zhang, C. L. P. Chen, and Z.-B. Zheng, "Multiobjective vehicle routing problems with simultaneous delivery and pickup and time windows: formulation, instances, and algorithms," *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 582–594, Mar. 2016.

[4] R. Roberti and M. Wen, "The electric traveling salesman problem with time windows," *Transp. Res. Part E: Logistics Transp. Rev.*, vol. 89, pp. 32–52, 2016.

[5] A. Zheng, Y. Yuan, J.-T. Zhou, Y.-F. Guo, H.-T. Yang, and O. C. Au, "Adaptive block coding order for intra prediction in HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 11, pp. 2152–2158, Nov 2016.

[6] M. J. Arnesen, M. Gjestvang, X. Wang, K. Fagerholt, K. Thun, and J. G. Rakke, "A traveling salesman problem with pickups and deliveries, time windows and draft limits: Case study from chemical shipping," *Comput. & Operations Res.*, vol. 77, pp. 20 – 31, 2017.

[7] S. Ronald, "Finding multiple solutions with an evolutionary algorithm," in *Proc. IEEE Int. Conf. Evol. Comput.*, vol. 2, Nov. 1995, pp. 641–646.

[8] D. Angus, "Niching for population-based ant colony optimization," in *IEEE Int. Conf. E-Science Grid Comput.*, Dec 2006, pp. 115–115.

[9] X.-C. Han, H.-W. Ke, Y.-J. Gong, Y. Lin, W.-L. Liu, and J. Zhang, "Multimodal optimization of traveling salesman problem: A niching ant colony system," in *Proc. Genet. Evol. Comput. Conf. Companion*. ACM, 2018, pp. 87–88.

[10] T. Huang, Y.-J. Gong, S. Kwong, H. Wang, and J. Zhang, "A niching memetic algorithm for multi-solution traveling salesman problem," *IEEE Trans. Evol. Comput.*, pp. 1–1, 2019.

[11] Z. Liu and Y. Wang, "Handling constrained multiobjective optimization problems with constraints in both the decision and objective spaces," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 870–884, 2019.

[12] Z. Liu, Y. Wang, S. Yang, and K. Tang, "An adaptive framework to tune the coordinate systems in nature-inspired optimization algorithms," *IEEE Trans. Cybern.*, vol. 49, no. 4, pp. 1403–1416, 2019.

[13] M. Prates, P. H. Avelar, H. Lemos, L. C. Lamb, and M. Y. Vardi, "Learning to solve NP-complete problems: A graph neural network for decision TSP," in *Proc. AAAI Conf. Artificial Intell.*, vol. 33, 2019, pp. 4731–4738.

[14] J. Zhou, G.-Q. Cui, Z.-Y. Zhang, C. Yang, Z.-Y. Liu, L.-F. Wang, C.-C. Li, and M.-S. Sun, "Graph neural networks: A review of methods and applications," 2018.

[15] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan 2009.

[16] T. Huang, Y. Gong, Y. Zhang, Z. Zhan, and J. Zhang, "Automatic planning of multiple itineraries: A niching genetic evolution approach," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–16, 2019.

[17] S. Lin, "Computer solutions of the traveling salesman problem," *Bell System Technical Journal*, vol. 44, no. 10, pp. 2245–2269, 1965.

[18] D. E. Goldberg, R. Lingle *et al.*, "Alleles, loci, and the traveling salesman problem," in *Proc. Int. Conf. Genet. Algorithms Their Appl.*, vol. 154. Lawrence Erlbaum, Hillsdale, NJ, 1985, pp. 154–159.

[19] R. Thomsen, "Multimodal optimization using crowding-based differential evolution," in *IEEE Congr. Evol. Comput.*, vol. 2, 2004, pp. 1382–1389.