Evolving Hard and Easy Traveling Salesman Problem Instances: A Multi-objective Approach

He Jiang, Wencheng Sun, Zhilei Ren, Xiaochen Lai, Yong Piao

School of Software, Dalian University of Technology, Dalian, China
jianghe@dlut.edu.cn, wencheng.sun.dlut@gmail.com, zren@dlut.edu.cn, laixiaochen@dlut.edu.cn, piaoy@dlut.edu.cn

Abstract. It becomes a great challenge in the research area of metaheuristics to predict the hardness of combinatorial optimization problem instances for a given algorithm. In this study, we focus on the hardness of the traveling salesman problem (TSP) for 2-opt. In the existing literature, two approaches are available to measure the hardness of TSP instances for 2-opt based on the single objective: the efficiency or the effectiveness of 2-opt. However, these two objectives may conflict with each other. To address this issue, we combine both objectives to evaluate the hardness of TSP instances, and evolve instances by a multi-objective optimization algorithm. Experiments demonstrate that the multi-objective approach discovers new relationships between features and hardness of the instances. Meanwhile, this new approach facilitates us to predict the distribution of instances in the objective space.

Keywords: TSP · 2-opt · multi-objective optimization algorithm · random forest

1 Introduction

Many metaheuristics such as genetic algorithms [12], local search [1], simulated annealing [11], tabu search algorithm [7], and ant colony optimization [9] have been used to solve NP-hard combinatorial optimization problems (COPs). For a particular NP-hard problem, there exist easy instances and hard instances for distinct algorithms. Hereafter, an instance could be obtained by specifying all the problem parameters with the given problem formulations [10]. With the development of metaheuristics, it becomes a hot topic to select an appropriate algorithm to resolve a given instance of a NP-hard COP. In [18], Rice first proposed the problem of algorithm selection, which seeks to predict which algorithm is likely to perform best on one given instance.

What exactly makes an optimization problem instance hard or easy? To answer this question, Macready [14] makes it clear that the features of an instance determine its hardness for a particular algorithm. Some recent survey papers [4] [15] point out that the instance features might influence algorithm performance which is denoted as exploratory landscape analysis. Researches in [13] [17] study the problem hardness to an algorithm by analyzing the expected running time.

In this study, we focus on the hardness of the Traveling Salesman Problem (TSP), which aims at finding a shortest tour visiting each of N cities once and returning to the starting city in the end. There have been a great number of metaheuristics to solve the TSP. We choose 2-opt [8], one of the most popular local search algorithms, to analyze the hardness of TSP instances based on their feature vectors. For a large-scale TSP instance, we calculate its features to predict its hardness for 2-opt, then we can know whether it is cost-effective to select 2-opt. More precisely, if the instance is hard for 2-opt, it is considerable to choose some other metaheuristics instead. However, it is still a challenge to measure the hardness of TSP instances for 2-opt. Two different approaches have been proposed to evaluate the hardness of TSP instances. One adopts the efficiency of 2-opt obtaining a local optimum to measure the hardness of TSP instances when solving these instances [19], while the other employs the effectiveness of the solutions achieved by 2-opt to evaluate the hardness of the instances [16]. Accordingly, each of them only considers one objective, either the efficiency or the effectiveness of 2-opt respectively. However, there exist some conflicts between two objectives [16]. For example, 2-opt possesses high efficiency but may achieve poor effectiveness with bad solutions on some instances, whereas it obtains desired effectiveness with low efficiency on some other instances.

To address this challenge, we evaluate the hardness of TSP instances by combining both the effectiveness and the efficiency objectives. More precisely, for 2-opt, one instance is easier than another if 2-opt achieves higher efficiency and better effectiveness on the former instance. Based on this evaluation formulation, we evolve easy and hard instances by a multi-objective optimization algorithm following NSGA-II [2]. For the purpose of straightforward illustration and significant analysis, all the instances are mapped into a 2-dimension objective space. Results show that the easy instances and the hard instances are distributed within different areas in the objective space. To study which features mainly affect the distribution of the instances in the objective space, we get the influence coefficient of each feature by training a prediction model. New relationships are discovered by the multi-objective approach that at least six features have a major influence on the hardness of TSP instances. The distribution of random TSP instances and TSPLIB instances in the objective space can be well predicted based on these relationships.

The remainder of this paper is organized as follows. Section 2 analyzes the relationships between two existing evaluation approaches to the hardness of TSP instances. Section 3 generates instances based on the multi-objective approach. Section 4 investigates the relationships between features and hardness. We evaluate the relationships on random TSP instances and sampled TSPLIB instances in Section 5. We conclude this paper in Section 6.

2 Traditional evaluations of hardness of TSP instances

In this section, we demonstrate the conflicts between two existing approaches for evaluating the hardness of TSP instances. There are two approaches to evaluate

the hardness of TSP instances in the literature based on single objective: efficiency or effectiveness. Smith-Miles et al. [19] measure the hardness of a given TSP instance by the efficiency of 2-opt on this instance, which is calculated by the number of 2-opt swaps to reach a local optimum. They consider that 2-opt has high efficiency on easy instances and low efficiency on hard instances. Meanwhile, Mersemann et al. [16] evaluate the hardness of a given instance for 2-opt by the effectiveness of 2-opt on this instance, which is presented by the quality of the solution achieved by 2-opt. To measure the quality of a solution obtained by 2-opt, they compare the solution against the global optimal solution achieved by the concorde solver [3]. Both researches use an evolutionary algorithm to evolve hard and easy TSP instances, and analyze the relationships between the features and the hardness. We adopt the genetic algorithm with the same crossover and mutation operators used in [16] to evolve instances based on the efficiency objective or the effectiveness objective. Moreover, we denote the corresponding collections of instances as "swaps_instances" and "quality_instances", respectively. A TSP instance is represented by a list of N (x, y) city coordinates on a 1×1 grid. To validate our finding on the instances provided on TSPLIB, we rescale the city coordinates of TSPLIB instances to a 1×1 grid as well.

2.1 Evolving TSP instances by traditional evaluations of hardness

We generate swaps_instances and quality_instances with fixed sizes of 25, respectively [16]. The size of an instance means the number of cities in the instance. We choose the 2-opt in [8] whose main idea is that making an initial solution randomly and obtaining a local optimum after a few of 2-opt swaps. Accordingly, we adopt 2-opt on each TSP instance and take the number of 2-opt swaps to reach a local optimum as the fitness of the instance for the genetic algorithm when generating swaps_instances. It is obvious that the fitness of each instance depends on the random initial solution, which makes the fitness of instances uncertain. To make the fitness of instances more reasonable, we use 2-opt to solve each instance 500 times, and take the average of the number of 2-opt swaps to reach a local optimum as the fitness of the instance.

We generate TSP instances randomly for the initial population. When evolving an easy swaps_instance, the instance which takes less 2-opt swaps for 2-opt to reach a local optimum has higher fitness. We select the instances with higher fitness from the current generation for the next generation, and the instance with the highest fitness in the last generation will be choosed as an easy swaps_instance. We repeat this process until we get the expected number of easy swaps_instances. In contrast, the instance taking more 2-opt swaps to reach a local optimum has higher fitness when evolving hard swaps_instances, and we choose the instance with the highest fitness after generations of optimization as a hard swaps_instance. Repeat this process until the desired number of hard swaps_instances are evolved.

In addition, we evolve quality_instances based on the effectiveness objective which is measured by the approximation ratio of path length that 2-opt achieves for a given TSP instance to the length of global optimal path achieved by the

concorde solver. The approximation ratio equals to 1 means that 2-opt has the same effectiveness as the concorde solver when solving an instance. Therefore, the closer the approximation ratio of a given instance is to 1, the easier the instance is for 2-opt. Similar to the process of evolving swaps_instances, we generate the quality_instances by taking the approximation ratio instead of the 2-opt swaps as the fitness of instances for the genetic algorithm.

100 swaps_instances and 100 quality_instances of either easy or hard with fixed sizes of 25 are evolved. Genetic algorithm parameters are set as follows. The size of initial population is 100, and the number of generations is 1000. The uniform mutation rate is 0.001, while the normal mutation rate is 0.01. We use a 1-elitism strategy that the best individual survives and will be contained in the next population, while the other instances for the next population will be generated by uniform crossover of the instances with high fitness.

2.2 The conflicts between two single objective approaches

To observe whether there exist conflicts between two objectives, we get the efficiency of 2-opt on quality_instances and the effectiveness of 2-opt on swaps_instances. Then each of quality_instances and swaps_instances can be mapped into a 2-dimensional objective space.

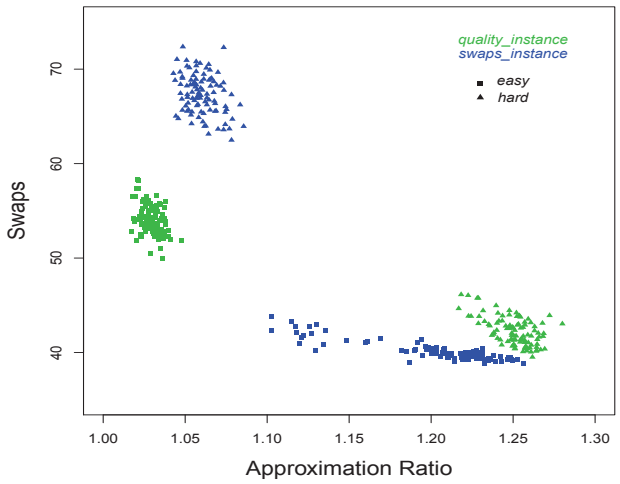


Fig. 1. The distribution of swaps_instances and quality_instances in the 2-dimensional objective space.

In Fig.1, instances are denoted as points, the x-axis represents the effectiveness of 2-opt on each instance, and the y-axis indicates the efficiency of 2-opt on each instance. Swaps_instances are represented in blue color, and quality_instances are in green color. Hard instances are denoted by triangles, while

easy instances are denoted as squares. It is shown in Fig.1 that 2-opt has lower efficiency on easy quality_instances than that on hard quality_instances, and has higher effectiveness on hard swaps_instances than that on easy swaps_instances. Therefore, there raise some conflicts that the instances which are considered as easy instances by one objective may be judged as hard ones based on the other objective, which implies evaluating the hardness of instances based on separate consideration of the efficiency objective or the effectiveness objective might be insufficient. To address this issue, we present a new approach to evaluate the hardness of instances which considers both the efficiency objective and the effectiveness objective. More precisely, for 2-opt, one instance is easier than another for 2-opt if 2-opt has higher efficiency and better effectiveness on the former, and vice versa. Based on the concept of Pareto optimality [6], we also evolve easy and hard instances which are denoted as "mul_instances" by a multi-objective optimization algorithm, and discover which features achieve the most influence on the hardness of instances for 2-opt.

3 Evolving TSP instances by multi-objective optimization

Since we evaluate the hardness of instances based on a multi-objective approach, we evolve TSP instances by a multi-objective optimization algorithm in this section. We first impose an additional concept into the multi-objective optimization algorithm as follows.

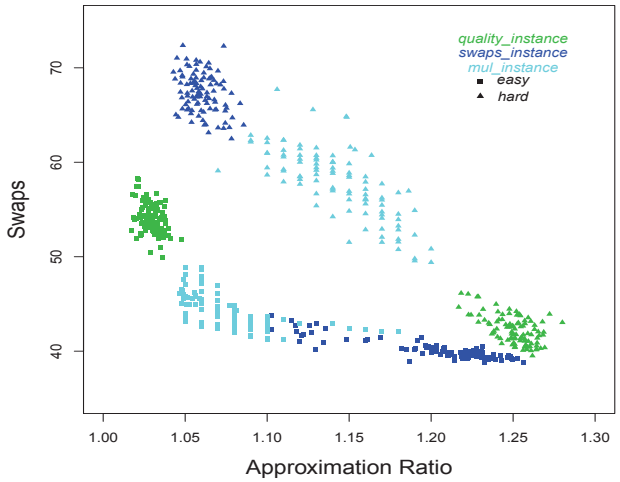


Fig. 2. The distribution of instances in the 2-dimensional objective space.

Given two individuals p and q in the population Pop, p dominates q (denoted by $p \succ q$) if they satisfy the following conditions, where $f_k(*)$ is the k_{th} objective of individual $*$:

– For all the objectives, p is not worse than q, i.e., $f_k(p) \leq f_k(q)$, $(k = 1, 2)$.
– There exists at least one objective such that p is better than q. That is, $\exists l \in \{1, 2\}$, s.t. $f_l(p) < f_l(q)$.

Algorithm 1 The construction of non-dominated individual set

```
for each p ∈ Pop do
  for each q ∈ Pop do
    if p dominates q then
      s_q = s_q ∪ {q} //the set of individuals dominated by the individual q
    end if
    if q dominates p then
      n_p = n_p + 1 //the number of individuals dominating the individual p
    end if
  end for
  if n_p = 0 then
    P_1 = P_1 ∪ {p}
  end if
end for
i = 1
while P_i ≠ ∅ do
  H = ∅
  for each p ∈ P_i do
    for each q ∈ s_p do
      n_q = n_q − 1 //the number of individuals dominating the individual q
      if n_q = 0 then
        H = H ∪ {q}
      end if
    end for
  end for
  i = i + 1
  P_i = H //the set of non-dominated individuals after the ith generation
end while
```

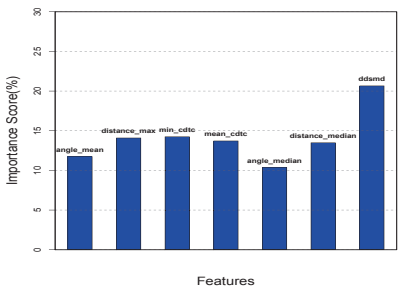
2-opt is also conducted on each instance 500 times, and we take the average number of swaps as the efficiency of 2-opt on the instance, while the average of approximation ratio is taken as the effectiveness of 2-opt on the instance. The parameters of the multi-objective optimization algorithm are also the same as the genetic algorithm used in the previous section. The key difference between the genetic algorithm [16] and the multi-objective optimization algorithm proposed in this paper is that we choose two instances from a non-dominated individual set randomly to evolve new instances by uniform crossover. We obtain the easy instances or the hard instances from the non-dominated individual set in the last generation. The pseudo-code (Algorithm 1) is used to build the non-dominated individual set in each generation.

Finally, we generate 100 easy instances and 100 hard instances with fixed sizes of 25 which are mapped into the objective space as well. Fig.2 shows that the easy and the hard instances locate in different regions and the mul_instances present

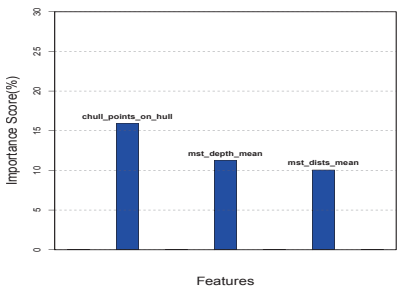
convex distribution as expected. Some hard mul_instances locate in those regions that hard swaps_instances or hard quality_instances locate in, which illustrates that the instances considered to be hard by the single objective approaches are also considered to be hard by the multi-objective approach.

4 The influential features to the hardness of instances

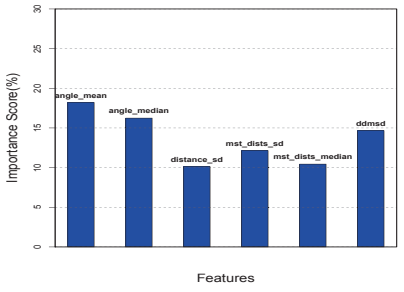
In this section, we investigate whether there are different combinations of features that affect the efficiency and the effectiveness of 2-opt most. We choose the features used in [16]. There are totally 47 features classified into 8 groups, including *Distance Features*, *Mode Features*, *Cluster Features*, *Nearest Neighbor Distance Features*, *Centroid Features*, *MST Features*, *Angle Features*, and *Convex Hull Features*. We calculate the features of all evolved instances and then conduct comparative analysis between the single objective approaches and the multi-objective approach.



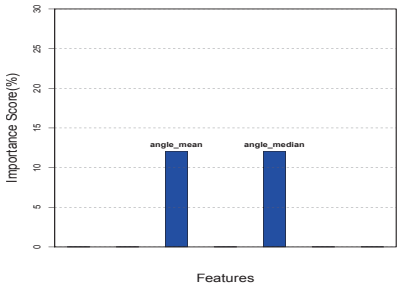
(a) Influential features discovered by mul_instances w.r.t. the effectiveness objective.



(b) Influential features discovered by quality_instances w.r.t. the effectiveness objective.



(c) Influential features discovered by mul_instances w.r.t. the efficiency objective.



(d) Influential features discovered by swaps_instances w.r.t. the efficiency objective.

Fig. 3. The importance scores of features to the hardness.

We discover the features that affect the hardness of instances most by training a prediction model based on the random forest [5]. The training set consists of 75 easy and 75 hard mul_instances, and the other 25 easy and 25 hard mul_instances compose the test set. We use the features of the test set to predict the corresponding efficiency and effectiveness of 2-opt. Root Mean Squared Error (RMSE) is used to indicate prediction error, which is defined as follow:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2}, \tag{1}$$

where y_i is the true value and $\hat{y_i}$ is the predicted value of the ith element. Then we delete each feature in turn, and record the percentage of error increase when removing a certain feature which is denoted as the importance score of this feature to the hardness. To find the features that affect the hardness of instances most, we select the features whose importance scores to the hardness are greater than 10%. Using the same approach on swaps_instances, we can get another combination of features that influence the efficiency of 2-opt most. The combination of features that influence the effectiveness of 2-opt most can be discovered on quality_instances as well.

Considering Fig.3, we can find the features that affect the effectiveness and the efficiency of 2-opt most discovered by the multi-objective approach are quite different from those discovered by the single objective approaches, which implies that new relationships between features and hardness are discovered by the multi-objective approach. The features that affect the efficiency of 2-opt most are also different from those affect the effectiveness of 2-opt most, which also explains that separately considering the efficiency objective or the effectiveness objective to evaluate the hardness of instances might be insufficient.

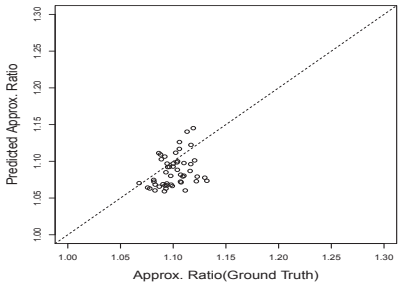
5 Validating the features on TSP instances

To clarify whether the new relationships discovered by the multi-objective approach are practically useful, we use the prediction model trained by mul_instances to predict the effectiveness and the efficiency of 2-opt on random TSP instances and TSPLIB instances based on the feature vectors of these instances in this section. In order to compare with the relationships discovered by the single objective approaches, the prediction model trained by quality_instances is used to predict the effectiveness of 2-opt and the prediction model trained by swaps_instances is used to predict the efficiency of 2-opt on these instances, respectively. In this section, RMSE is also used to indicate prediction error.

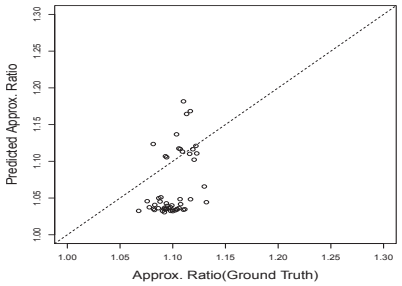
5.1 Validating the features on random TSP instances

We generate 50 TSP instances with fixed sizes of 25 randomly which compose the test set. There are three groups of training sets for building random forest:

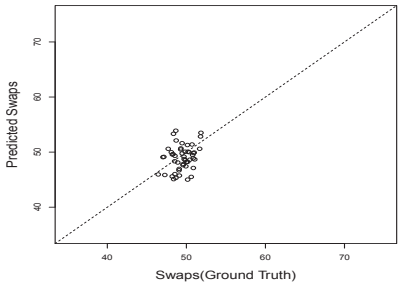
– Mul_Training_Set: consisting of 75 easy mul_instances and 75 hard mul_instances;
– Quality_Training_Set: consisting of 75 easy quality_instances and 75 hard quality_instances;
– Swaps_Training_Set: consisting of 75 easy swaps_instances and 75 hard swaps_instances.



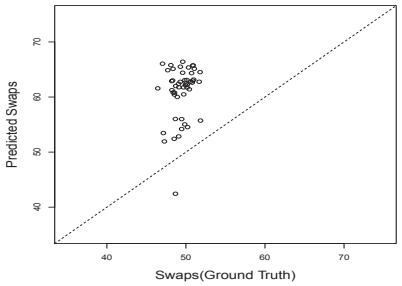
(a) prediction by mul_instances w.r.t. the effectiveness objective.



(b) prediction by quality_instances w.r.t. the effectiveness objective.



(c) prediction by mul_instances w.r.t. the efficiency objective.



(d) prediction by swaps_instances w.r.t. the efficiency objective.

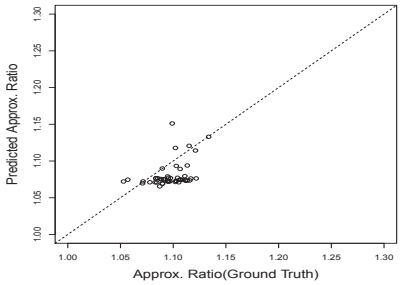
Fig. 4. Prediction comparision on random TSP instances.

We achieve a RMSE of 0.0253 when using the prediction model trained by Mul_Training_Set to predict the effectiveness of 2-opt on random instances, and the RMSE for the prediction model trained by Quality_Training_Set is 0.0523. The RMSE for the prediction model trained by Mul_Training_Set to predict the efficiency of 2-opt on random instances is 2.32, which is lower than that of 12.37 obtained by the prediction model trained by Swaps_Training_Set. The RMSE values in the two different objective dimensions are not in the same order of magnitude. This is because there exists a big gap between the magnitudes of the 2-opt swaps and the approximation ratio. Fig.4 shows that the prediction model trained by Mul_Training_Set is better to predict the efficiency and the effectiveness of 2-opt on random instances, which illustrates that the multi-objective approach can better discover the relationships between the features and the hardness than the single objective approaches.

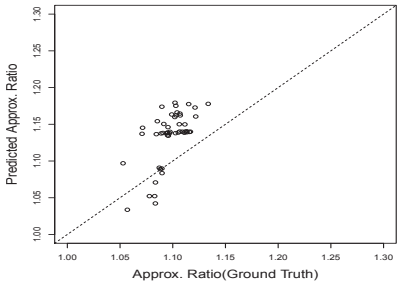
5.2 Validating the features on sampled TSPLIB instances

In order to further validate the new relationships between features and hardness, we use the prediction models trained by these three groups of training sets to predict the efficiency and the effectiveness of 2-opt on sampled TSPLIB instances. The instances on TSPLIB have different sizes. However, the training instances are all with the fixed sizes of 25. To be coincident with the training instances, we select 50 TSP instances from TSPLIB whose size is larger than 25 and extract 25 coordinates of cities from each of the TSPLIB instances randomly. Then we will obtain a test set with 50 sampled TSPLIB instances.

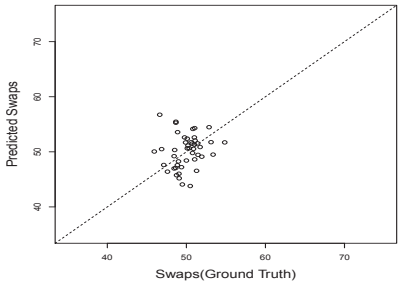
The prediction results are shown in Fig.5. The RMSE is 0.0464 for the prediction model trained by Quality_Training_Set, and RMSE obtained by the prediction model trained by Mul_Training_Set is 0.0247 when predicting the effectiveness of 2-opt on sampled TSPLIB instances. The prediction model trained by Mul_Training_Set achieves a better prediction with RMSE of 3.26 than the RMSE of 10.93 obtained by the prediction model trained by Swaps_Training_Set, which also implies that the model trained by Mul_Training_Set can better predict the efficiency of 2-opt on sampled TSPLIB instances. Overall, the multi-



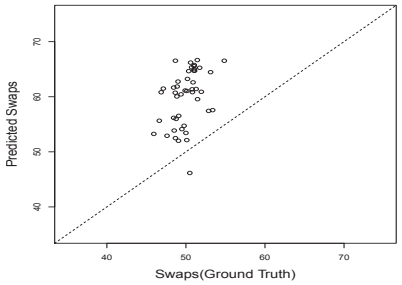
(a) prediction by mul_instances w.r.t. the effectiveness objective.



(b) prediction by quality_instances w.r.t. the effectiveness objective.



(c) prediction by mul_instances w.r.t. the efficiency objective.



(d) prediction by swaps_instances w.r.t. the efficiency objective.

Fig. 5. Prediction comparision on sampled TSPLIB instances.

objective approach achieves higher accuracy in predicting the distribution of

TSP instances in the objective space, which illustrates that the multi-objective approach can better discover the relationships between the features and the hardness of instances for 2-opt.

In this section, we validate the features on the sampled TSPLIB instances with fixed sizes of 25. Further investigation needs to be conducted on TSPLIB instances with real sizes, which needs us to evolve training instances with different sizes.

6 Conclusion

There are two existing approaches to evaluate the hardness of TSP instances for 2-opt based on single objective. However, the objectives may conflict with each other. The instances which are considered as easy instances by one single objective may be judged as hard ones w.r.t. the other objective. To address this challenge, we propose a new evaluation approach by combining both objectives. For 2-opt, one instance is easier than another instance if 2-opt has higher efficiency and better effectiveness on the former, and vice versa.

We use a multi-objective optimization algorithm to evolve hard and easy TSP instances. Then we study the relationships between features and hardness. To find the combinations of features that affect the efficiency and the effectiveness of 2-opt most, we employ the random forest to get the importance score of each feature. Experimental results show that the relationships between features and hardness discovered by the multi-objective approach is quite different from those discovered by single objective approaches. There are at least six features with the most effect on the efficiency and the effectiveness of 2-opt discovered by the multi-objective approach. In the end, we verify our finding on random TSP instances and TSPLIB instances, and the results show that the relationships discovered by the multi-objective approach can provide more help for us to predict the distribution of TSP instances in the objective space.

Acknowledgements This work was supported in part by the Fundamental Research Funds for the Central Universities under Grant DUT13RC(3)53, in part by the New Century Excellent Talents in University under Grant NCET-13-0073, in part by China Postdoctoral Science Foundation under Grant 2014M551083, and in part by the National Natural Science Foundation of China under Grant 61175062.

References

1. Aarts, E.H., Lenstra, J.K.: Local search in combinatorial optimization. Princeton University Press (2003)
2. Abbass, H.A., Sarker, R., Newton, C.: PDE: a pareto-frontier differential evolution approach for multi-objective optimization problems. In: Proceedings of the 2001 Congress on Evolutionary Computation. pp. 971–978. IEEE (2001)

3. Applegate, D., Bixby, R., Chvatal, V., Cook, W.: Concorde tsp solver (2011), http://www.tsp.gatech.edu/concorde.html
4. Bischl, B., Mersmann, O., Trautmann, H., Preuss, M.: Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In: Proceedings of the 14th International Conference on Genetic and Evolutionary Computation Conference. pp. 313–320. ACM (2012)
5. Breiman, L.: Random forests. Machine learning 45(1), 5–32 (2001)
6. Censor, Y.: Pareto optimality in multiobjective problems. Applied Mathematics and Optimization 4(1), 41–59 (1977)
7. Chen, L., Bostel, N., Dejax, P., Cai, J., Xi, L.: A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal. European Journal of Operational Research 181(1), 40–58 (2007)
8. Croes, G.: A method for solving traveling-salesman problems. Operations Research 6(6), 791–812 (1958)
9. Dorigo, M., Birattari, M.: Ant colony optimization. In: Encyclopedia of Machine Learning, pp. 36–39. Springer (2010)
10. Garey, M.R., Johnson, D.S.: Computers and intractability: a guide to the theory of NP-completeness. WH Freeman & Co., San Francisco (1979)
11. Goffe, W.L., Ferrier, G.D., Rogers, J.: Global optimization of statistical functions with simulated annealing. Journal of Econometrics 60(1), 65–99 (1994)
12. Goldberg, D.E.: Genetic algorithms. Pearson Education India (2006)
13. He, J., Chen, T., Yao, X.: On the easiest and hardest fitness functions (2012)
14. Macready, W.G., Wolpert, D.H.: What makes an optimization problem hard? Complexity 1(5), 40–46 (1996)
15. Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory landscape analysis. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation. pp. 829–836. ACM (2011)
16. Mersmann, O., Bischl, B., Trautmann, H., Wagner, M., Bossek, J., Neumann, F.: A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem. Annals of Mathematics and Artificial Intelligence 69(2), 151–182 (2013)
17. Qian, C., Yu, Y., Zhou, Z.H.: On algorithm-dependent boundary case identification for problem classes. In: Parallel Problem Solving from Nature-PPSN XII, pp. 62–71. Springer (2012)
18. Rice, J.R.: The algorithm selection problem. Advances in Computers 15, 65–118 (1976)
19. Smith-Miles, K., van Hemert, J., Lim, X.Y.: Understanding TSP difficulty by learning from evolved instances. In: Learning and intelligent optimization, pp. 266–280. Springer (2010)