Parameter Prediction Based on Features of Evolved Instances for Ant Colony Optimization and the Traveling Salesperson Problem

Samadhi Nallaperuma, Markus Wagner, and Frank Neumann

Optimisation and Logistics
School of Computer Science
The University of Adelaide, Australia

Abstract. Ant colony optimization performs very well on many hard optimization problems, even though no good worst case guarantee can be given. Understanding the reasons for the performance and the influence of its different parameter settings has become an interesting problem. In this paper, we build a parameter prediction model for the Traveling Salesperson problem based on features of evolved instances. The two considered parameters are the importance of the pheromone values and of the heuristic information. Based on the features of the evolved instances, we successfully predict the best parameter setting for a wide range of instances taken from TSPLIB.

1 Introduction

Ant colony optimization (ACO) [3] has become very popular in recent years to solve a wide range of hard combinatorial optimization problems. Throughout the history of heuristic optimization, attempts have been made to analyze ACO algorithm performance theoretically [6, 17] and experimentally [11, 19]. However, much less work has been done towards the goal of explaining the impact of the problem instance structure and the algorithm parameters on performance.

The study in [19] provides an overview of existing parameter prediction/tuning approaches for ACO in two major directions: (1) parameter choosing before running the algorithm (offline configuration and tuning), and (2) adaptation during runtime (online tuning). It has been shown in [12] that offline tuning outperformes online tuning for the Max-Min Ant System (MMAS) applied to the Traveling Salesperson problem (TSP). However, the drawback of existing offline parameter configuration techniques is that they are time consuming and use a lot of computing power as they need to run iteratively on training instances. We refer the reader to [5] for a discussion on general parameter tuning and prediction methods. To the best of our knowledge, none of the existing approaches have taken structural features of evolved problem instances into consideration when setting the algorithm's parameters.

In early research, the problem hardness analysis of the TSP was based on only a few features that describe the edge cost distribution [14, 20], and the algorithms were typically run on predetermined instances. Later on, more sophisticated methods were

T. Bartz-Beielstein et al. (Eds.): PPSN XIII 2014, LNCS 8672, pp. 100–109, 2014.

introduced for the instance generation, and the investigated problem features have become more diverse [7, 10, 15]. However, a comprehensive analysis of ACO and TSP problem features has not been conducted so far.

We study the potential of feature-based characterization to be used in automatic algorithm configuration for ACO and consider the well-known Max-Min Ant System [18] for the TSP. One important question in the configuration of ACO algorithms is to what extent the pheromone values and the heuristic information should influence the behaviour of the algorithm—the importance of these two components is determined by the parameters α (for pheromone values) and β (for heuristic information). We first investigate statistical features of evolved (hard, easy, and in-between) instances from [9] and their impact on the appropriate choice of these two parameters. Based on this we build a prediction model in order to predict the right choice for instances of TSPLIB [13]. The potential strength of the prediction model relies on the wide range and on the diversity of the evolved instances, and on the expressiveness of the selected structural features. Our experimental investigations show that the considered features and evolved instances are well suited to predict an appropriate choice for setting the parameters α and β of MMAS.

The outline of the paper is as follows. In Section 2, we introduce the algorithm and the framework of our investigations. In Section 3, we report on easy and hard instances for different parameter combinations and carry out a feature-based analysis. Subsequently, we use these insights to predict parameters for given instances from TSPLIB in Section 4, and we finish with some concluding remarks.

2 Preliminaries

The Traveling Salesperson problem (TSP) is one of the most famous NP-hard combinatorial optimization problems. Given a set of n cities $\{1, \ldots, n\}$ and a distance matrix $d = (d_{ij})$, $1 \leq i, j \leq n$, the goal is to compute a tour of minimal length that visits each city exactly once and returns to the origin. We consider the still NP-hard Euclidean TSP, where cities are given by points in the plane and distances are given by the Euclidean distances between these points.

As above-mentioned, our study is focused on the well-known ACO algorithm called Max-Min Ant System (MMAS) [18]. Solutions are constructed by ants visiting cities sequentially, according to a probabilistic formula defined as

$$p_{ij} = \frac{[\tau_{ij}]^{\alpha} * [\eta_{ij}]^{\beta}}{\left(\Sigma_{h \in N_k} [\tau_{ih}]^{\alpha} * [\eta_{ih}]^{\beta}\right)},$$

where N_k represents the set of unvisited nodes of ant k, $[\tau_{ih}]$ and $[\eta_{ih}]$ having exponents α and β that represent pheromone and heuristic information respectively. A detailed description and analysis of this algorithm on TSP can be found in the textbook of Dorigo and Stützle (Chapter 3) [3].

To evolve easy and hard instances for the ant algorithms we use the evolutionary algorithm approach previously studied on 2-opt [7] and approximation algorithms [10] for the TSP. The only difference in the instance generation process here is that we

consider several algorithm instances with different parameter settings instead of a single algorithm.

The approximation ratio $\alpha_A(I)$ of an algorithm A for a given instance I is defined as

$$\alpha(I) = A(I)/OPT(I)$$

where $A(I)$ is the tour length produced by algorithm A for the given instance I, and *OPT(I)* is the value of an optimal solution of I. *OPT(I)* is obtained by using the exact TSP solver Concorde [2].

3 Features of Hard and Easy Instances

For each ACO algorithm instance with a specific parameter setting of α and β, a set of 100 random TSP instances is generated in two-dimensional unit square $[0, 1]^2$ and placed on a discretized grid. The evolutionary algorithm runs on them for 5000 generations in order to generate a set of hard and a set of easy instances. Each ACO execution is limited to two seconds. In each iteration, the ACO algorithm is run once on a single instance, and then the approximation ratio is calculated. In separate runs, either a higher approximation ratio is favoured to generate hard instances, or a lower ratio is favoured to generate easy instances. This process is repeated for instances of sizes 25, 50, 100 and 200 with the goal of generating easy and hard instances respectively. The instance generation is performed on an Unix cluster with 48 nodes where each node has 48 cores (4 AMD 6238 12-core 2.6Ghz CPUs) and 128GB memory (2.7GB per core).

This study considers 47 features including distances of edge cost distribution, angles between neighbors, nearest neighbor statistics, mode, cluster and centroid features as well as features representing minimum spanning tree heuristics and of the convex hull. A detailed description of these features can be found in [10].

The algorithm parameters considered in this study are the most popular and critical ones in any ACO algorithm, namely the exponents α and β, which represent the influence of the pheromone trails and of the heuristic information respectively. We consider three parameter settings for our analysis: setting 1 represents default parameters ($\alpha = 1$, $\beta = 2$), and settings 2 and 3 represent extreme settings with highest and lowest values in a reasonable range ($\alpha = 0$, $\beta = 4$ and $\alpha = 4$, $\beta = 0$). The general idea behind the choice is that we have to isolate the conditions to investigate the effect which is usually considered in traditional scientific experiments. The rest of the parameters are set in their default values ($\rho = 0.2$, ants = 20) as in the original MMAS implementation by Stützle [16].

3.1 Feature Analysis

Our experimental results for the MMAS with the first parameter setting ($\alpha = 1$, $\beta = 2$) show the following.[1] For the first and the second parameter settings, the standard deviation of angles of the easy instances are significantly smaller than the values of the

[1] Due to space limitations here we present only a few significant findings. We refer to Nallaperuma et al. [9] for some preliminary results of the feature analysis.

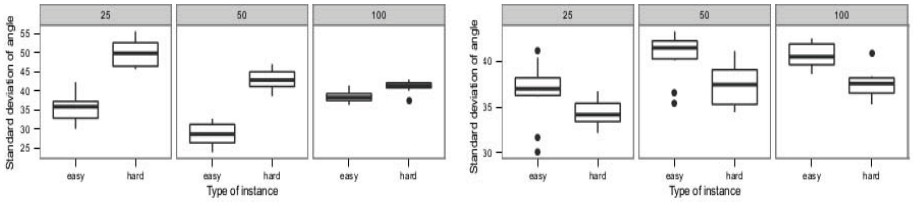

Fig. 1. Boxplots of the standard deviations of the angles between adjacent cities on the optimal tour for parameter setting 2 ($\alpha = 0$, $\beta = 4$) on the left and setting 3 ($\alpha = 4$, $\beta = 0$) on the right

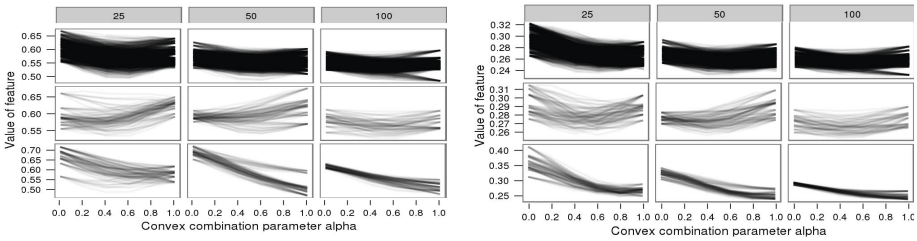

Fig. 2. Feature variation with instance difficulty for mean (left) and standard deviation (right) of distances for the three parameter settings 1 (top), 2 (middle) and 3 (bottom)

hard instances. With increasing instance size, these values change differently for easy and hard instances. Interestingly, this structural difference is even obvious to a human observers who perceive different "shapes" for easy/hard and smaller/larger instances. In contrast to the patterns of the first two parameter settings, the third combination ($\alpha = 4$, $\beta = 0$) shows an increasing pattern of standard deviation values (with increasing instance size), whereas these values follow a decreasing pattern in the case of the second setting (see Figure 1).

We also study the feature variation for the instances of intermediate difficulty. In order to do this, it is required to generate instances with varying difficulty levels in-between the two extreme difficulties hard and easy. This can be achieved through morphing, where we create instances with varying difficulty levels by forming convex combinations of easy and hard instances. Here, the point matching is done using a greedy strategy where the points of minimum Euclidian distance are matched. These matched instances are then used to produce a set of instances with intermediate difficulty by taking the convex combination based on the convex combination parameter $\alpha_c \in \{0, 0.2, ..., 0.8, 1\}$ where 0 represents hardest instances and 1 easiest.

Generally, for all three considered parameter settings, most features show similar patterns exhibiting systematic nonlinear relationships with instance difficulty. However, there are a few "contrast patterns" (the feature is increasing in value over instance difficulty for one parameter setting and decreasing for another parameter setting) observed among different parameter settings. For example, the distance mean and the standard deviation show contrast patterns for the second parameter setting ($\alpha = 0$, $\beta = 4$) from the other two (see Figure 2). Moreover, we observe that the sharp increasing pattern over instance difficulty for the third parameter setting ($\alpha = 4$, $\beta = 0$) has slowed down

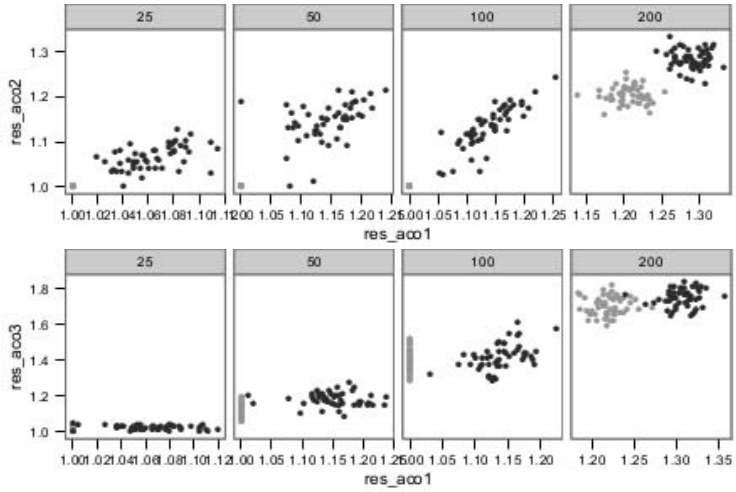

Fig. 3. Performance of the second parameter setting (top) and the third (bottom) on the easy (grey) and hard (black) instances of the first parameter setting

for the default parameter setting, and even converted to a decreasing pattern for the second setting. This provides strong evidence on the impact of parameters. Similar contrast patterns are observed in the other feature groups as well, such as convex hull and nearest neighbour. These contrast patterns suggest the dependence of problem hardness on the algorithm parameters. This dependence further indicates that algorithms with different settings can have complimentary problem-solving capabilities. We believe that such capabilities can provide insights to automatic parameter configuration. Therefore, we further investigate these capabilities by comparing the approximation ratios of the three algorithms achieved on each others' easy and hard instances.

3.2 Comparison of Parameter Settings

As shown in Figure 3, both the second ($\alpha = 0$, $\beta = 4$) and the third ($\alpha = 4$, $\beta = 0$) parameter settings have obtained worse approximation ratios for the easy instances of the first parameter setting ($\alpha = 1$, $\beta = 2$) than the first parameter setting. In the case of the hard instances, the second parameter setting has achieved better approximation ratios than the first parameter setting itself. The outcomes of the other two cross-checks are comparable: given the hard instances of one algorithm configuration, the other two settings achieve better results. This is strong support for our previous conjecture on the complimentary capabilities of different parameter settings.

4 Parameter Prediction

In order to build a reliable model, we significantly extend our collection of data gained from the experiments in Section 3. We generate 1500 instances: 10 hard and easy ones, with sizes 25, 50 and 100, and for each of the 25 parameter combinations $\alpha, \beta \in \{0, 1, 2, 3, 4\}$.

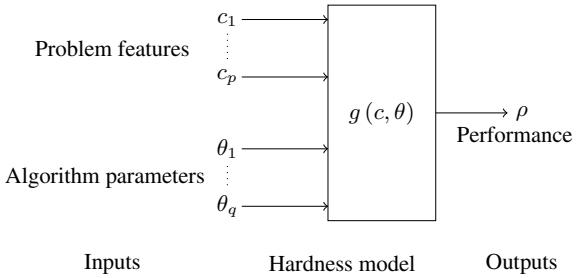

Fig. 4. Prediction Model. It predicts the algorithm performance based on the problem features c_i, $1 \leq i \leq p$ and the possible algorithm parameters θ_j, $1 \leq j \leq q$.

4.1 Prediction Model

We build a simple prediction model merely as a proof of concept that a problem hardness model can be used for ACO parameter prediction. Therefore, we use a popular basic technique for model building. A high level overview of the model is shown in Figure 4. Instead of predicting the allegedly optimal parameters, we actually predict the approximation values given the 25 possible parameter combinations. Then, we select amongst those 25 the combination that achieves the best approximation as the model's output. Hence, the actual model construction is based on the approximation ratio as the dependent variable. Note that a similar model architecture is used in the recent work of Munoz et al. [8] for the prediction of algorithm performance based on landscape features and parameters.

To build our prediction model, we use the classical pattern classification technique introduced by Aha et al. [1], as implemented in the Weka data mining framework [4]. In the training phase, we feed the generated instances into this nearest neighbour search based classifier. As we have seen in the previous hardness analysis, not all problem features appear to be significantly different for easy and hard instances. Consequently, a smaller subset with 15 *strong* features out of the whole (47) feature set is selected: angle mean, angle median, angle sd, centroid mean distance to centroid, centroid max distance to centroid, points on hull, distance mean, distance median, distance max, distance sd, mst distance mean, mst distance max, mst distance sd, nearest neighbour distance sd and nearest neighbour distance coefficient of variance.

4.2 Prediction Results

First, we test our model on a set of 30 randomly generated TSP instances of instance size 100. In the first step, their approximation values are calculated (by averaging the outcomes of 50 repetitions) for all (25) considered parameter combinations. Then the actual best-performing parameter setting is found based on those 25 approximation values (see Table 1 (a) for results).

For more than half of the 30 instances, the model predicts the *correct* minimal approximation ratio, as both winning parameters are the best actual values. Almost all remaining predictions are close to the optimal combination, predicting the actual second best parameter combination. Even though our model is relatively simple, we believe

Table 1. (a): Predicted and actual parameter settings (α, β) for 30 random test instances of size 100. The columns "best" and "second" show the parameter combinations for which the best approximation and second best approximations are achieved in both prediction and actual experiments. **(b):** Results of the Wilcoxon signed rank tests on the predicted and actual approximation ratios for same instances for the hypothesis "actual $>$ predicted" (Test 1) and "predicted $>$ actual" (Test 2), positive rank sum (W) and confidence (p) values are displayed accordingly.

(a)

inst	predicted best/second	actual best/second	match	second match
1	(4 , 4) / (1, 4)	(1, 4)	no	yes
2	(1, 2)	(1,3) / (1, 2)	no	yes
3	(4, 3)	(4, 3)	yes	
4	(1, 3)	(1, 3)	yes	
5	(1, 4)	(1, 4)	yes	
6	(1, 4)	(1, 4)	yes	
7	(1, 4)	(1, 4)	yes	
8	(1, 4)	(1, 4)	yes	
9	(1, 3)	(1, 3)	yes	
10	(2, 3)	(2, 3)	yes	
11	(1, 3)	(1, 3)	yes	
12	(1, 3)	(1, 4)/(1, 3)	no	yes
13	(1, 3)	(1, 3)	yes	
14	(4, 4)	(1, 4)/(4, 4)	no	yes
15	(1, 2)	(1, 4)/(1, 2)	no	yes
16	(3, 4)	(3, 4)	yes	
17	(1, 4)	(1, 3)/ (1, 4)	no	yes
18	(1, 2)	(1, 3)/ (1, 2)	no	yes
19	(4, 4)	(2, 4)/ (4, 4)	no	yes
20	(1, 3)	(1, 2)/ (1, 3)	no	yes
21	(4, 4)	(4, 4)	yes	
22	(1, 4)	(1, 4)	yes	yes
23	(1, 1)	(1, 3)	no	no
24	(1, 1)	(1, 1)	yes	
25	(1, 4)	(1, 3)/(1, 4)	no	yes
26	(1, 4)	(1, 4)	yes	
27	(1, 2)	(1, 2)	yes	
28	(1, 3)	(1, 3)	yes	
29	(1, 3)	(1, 3)	yes	
30	(3, 4)/ (1, 4)	(1, 4)	no	yes

(b)

inst	Test 1		Test 2	
	W	p-value	W	p-value
1	99	0.9305	201	0.0714
2	565.5	0.6797	659.5	0.3221
3	1296.5	0.5371	1331.5	0.4639
4	2191	0.6233	2369	0.3774
5	3404	0.6666	3736	0.3339
6	4956.5	0.7049	5483.5	0.2954
7	6808.5	0.7276	7556.5	0.2727
8	8735	0.8028	9986	0.1973
9	11117	0.7959	12536	0.2042
10	14274.5	0.5590	14405.5	0.4411
11	16863	0.6171	17328	0.3831
12	20273.5	0.6086	20767.5	0.3915
13	24308.5	0.4869	23896.5	0.5132
14	28605	0.4325	27675	0.5676
15	32799.5	0.4094	31461.5	0.5907
16	37968.5	0.3269	35567.5	0.6732
17	43214	0.2709	39814	0.7291
18	49679	0.1532	43849	0.8468
19	54671	0.1863	49069	0.8138
20	61524	0.1248	53916	0.8753
21	68491.5	0.0712	58264.5	0.9288
22	75075	0.0710	64053	0.9290
23	80356.5	0.1497	71719.5	0.8504
24	88700.5	0.0856	76899.5	0.9144
25	96134	0.0750	82967	0.9250
26	104622.5	0.0627	89753.5	0.9373
27	113339	0.0552	96937	0.9449
28	122681.5	0.0362	103446.5	0.9639
29	130368	0.0564	112188	0.9436
30	140646.5	0.0394	119634.5	0.9606

that this first result already supports our initial claim that parameters can be predicted based on preceding instance analyses.

Although the model cannot produce the best parameter setting for all instances, the raw approximation values for the predicted and the actual performance are very similar. Therefore, we conduct a rank test to observe any significant difference between the predicted values. We choose the Wilcoxon signed rank test [21], as there is no guarantee about the distribution, and the results are paired as they are based on the TSP instance on which the approximation ratio is obtained. For each TSP instance the predicted and actual approximation ratios obtained for all parameter settings are considered for the test. For the first test we set the hypothesis that the actual values are greater than the predicted values, and then the test is repeated with the counter hypothesis. For both tests and for most instances, the resulting p values are reasonably large, hence both of the alternative hypothesis are rejected (see Table 1 (b)). Thus, we fail to reject the null hypothesis, meaning that both distributions are equal. Only for two instances we

Table 2. Predicted and actual parameter settings (α, β) for 25 TSPLIB instances of size in range 51–264. Note, that the underlying model is based only on our analysis of instances of size 100.

inst	predicted best/second	actual best/second	match	second match
bier127.tsp	(2, 3)	(2, 3)	yes	
ch150.tsp	(1, 3)/(2,3)	(2, 3)	no	yes
eil51.tsp	(1, 3)	(1, 3)	yes	
kroA100.tsp	(3,3)/(1,3)	(1, 3)	no	yes
kroB100.tsp	(1,3)	(1, 3)	yes	
kroC100.tsp	(1,3)	(1, 3)	yes	
kroD100.tsp	(1,2)	(1, 2)	yes	
pr107.tsp	(2,4)	(2, 4)	yes	
pr76.tsp	(2, 3)	(2, 3)	yes	
st70.tsp	(1, 1)	(1, 1)	yes	
ch130.tsp	(2, 3)	(2, 4)/(2,3)	no	yes
eil101.tsp	(2, 2)/(2,3)	(2, 3)	no	yes
kroA150.tsp	(2, 2)	(2, 2)	yes	
lin105.tsp	(3,2)/(1,3)	(1, 3)	no	yes
pr124.tsp	(4,3)/(1,3)	(1,3)	no	yes
rat99.tsp	(2, 4)/(1,2)	(1, 4)/(1,3)	no	no
kroB150.tsp	(1, 2)/(1, 3)	(2, 3)/(3,4)	no	no
eli79.tsp	(3,4)/(1,2)	(1, 3)/(1,2)	no	yes
kroE100.tsp	(1,1)/(1,3)	(1, 4)/(1,3)	no	yes
kroA200.tsp	(3, 3)	(3, 3)	yes	
kroB200.tsp	(3, 4)	(3, 4)	yes	
tsp225.tsp	(2, 3)/ (3, 4)	(4, 4)/(3, 4)	no	yes
pr264.tsp	(4, 4)	(4, 4)	yes	
gil262.tsp	(2, 3)/(1, 4)	(4, 4)/ (4, 3)	no	no
pr226.tsp	(2, 3)/(1, 3)	(4, 3)/(3, 3)	no	no

observe p values less than 0.05, and thus we fail to reject the alternative hypothesis with 95% significance (thus reject the null hypothesis) that they are different.

Second, we test our model on a set of famous benchmark instances from TSPLIB [13] and the results are shown in Table 2. Interestingly, they are qualitatively similar to the results of the test on random TSP instances (Table 1), even though these "real world" instances have never been part of the model building process. Therefore, this second investigation provides further evidence on the accuracy of the model-based performance predictions. We conjecture that the reasons for this strong performance are (1) the large distribution of the training set varying from extreme hard to extreme easy TSP instances and (2) the strength of the selected feature set in expressing problem hardness for ACO algorithm instances with specified parameter settings. In order to use this model for prediction, a very short prepossessing step is required that calculates the 15 above-mentioned feature values for the input instance.

5 Conclusions

In this paper, we have shown how to predict the parameter setting of ACO algorithms based on features of evolved problem instances. We considered the parameters α and β which determine the importance of the pheromone concentration and heuristic information, respectively. Based on instance features for the classical Traveling Salesperson Problem, we built a prediction model to determine the values of α and β. Our investigations on a wide range of instances from TSPLIB show that the instance features allow

for a reliable prediction of well-performing algorithm setups. For future work, we plan on improving the prediction model by integrating other ACO parameters such as the number of ants and the pheromone update strength.

Acknowledgements. We thank Bernd Bischl for early discussions, Heike Trautmann and Olaf Mersmann for the feedback on the preliminary version of this research. This research has been supported by the Australian Research Council (ARC) under grant agreement DP140103400.

Bibliography

[1] Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Mach. Learn. 6(1), 37–66 (1991)
[2] Applegate, D., Cook, W.J., Dash, S., Rohe, A.: Solution of a Min-Max Vehicle Routing Problem. Journal on Computing 14(2), 132–143 (2002)
[3] Dorigo, M., Stützle, T.: Ant Colony Optimization. Bradford Company (2004)
[4] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: An update. SIGKDD Explororations Newsletter 11(1), 10–18 (2009)
[5] Hoos, H.: Automated algorithm configuration and parameter tuning. In: Hamadi, Y., Monfroy, E., Saubion, F. (eds.) Autonomous Search, pp. 37–71. Springer, Heidelberg (2012)
[6] Kötzing, T., Neumann, F., Röglin, H., Witt, C.: Theoretical analysis of two ACO approaches for the traveling salesman problem. Swarm Intelligence 6, 1–21 (2012)
[7] 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. In: Annals of Mathematics and Artificial Intelligence, pp. 1–32 (2013)
[8] Muñoz, M.A., Kirley, M., Halgamuge, S.K.: A meta-learning prediction model of algorithm performance for continuous optimization problems. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part I. LNCS, vol. 7491, pp. 226–235. Springer, Heidelberg (2012)
[9] Nallaperuma, S., Wagner, M., Neumann, F.: Ant colony optimisation and the traveling salesperson problem: Hardness, features and parameter settings (extended abstract). In: 15th Annual Conference Companion on Genetic and Evolutionary Computation Conference Companion (GECCO Companion), pp. 13–14. ACM (2013)
[10] Nallaperuma, S., Wagner, M., Neumann, F., Bischl, B., Mersmann, O., Trautmann, H.: A Feature-based Comparison of Local Search and the Christofides Algorithm for the Travelling Salesperson Problem. In: International Conference on Foundations of Genetic Algorithms, FOGA (2013)
[11] Pellegrini, P., Favaretto, D., Moretti, E.: On $\mathcal{MAX}$ – $\mathcal{MIN}$ ant system's parameters. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 203–214. Springer, Heidelberg (2006)
[12] Pellegrini, P., Stützle, T., Birattari, M.: Off-line *vs.* on-line tuning: A study on $\mathcal{MAX} - -\mathcal{MIN}$ ant system for the TSP. In: Dorigo, M., Birattari, M., Di Caro, G.A., Doursat, R., Engelbrecht, A.P., Floreano, D., Gambardella, L.M., Groß, R., Şahin, E., Sayama, H., Stützle, T. (eds.) ANTS 2010. LNCS, vol. 6234, pp. 239–250. Springer, Heidelberg (2010)
[13] Reinelt, G.: TSPLIB – A Traveling Salesman Problem Library. ORSA Journal on Computing 3(4), 376–384 (1991)

[14] Ridge, E., Kudenko, D.: Determining Whether a Problem Characteristic Affects Heuristic Performance. In: Cotta, C., van Hemert, J. (eds.) Recent Advances in Evol. Comp. SCI, vol. 153, pp. 21–35. Springer, Heidelberg (2008)
[15] Smith-Miles, K., van Hemert, J., Lim, X.Y.: Understanding TSP difficulty by learning from evolved instances. In: Blum, C., Battiti, R. (eds.) LION 4. LNCS, vol. 6073, pp. 266–280. Springer, Heidelberg (2010)
[16] Stützle, T.: Software package: Acotsp.v1.03.tgz (2012)
[17] Stützle, T., Dorigo, M.: A short convergence proof for a class of Ant Colony Optimization algorithms. IEEE Trans. on Evolutionary Computation, 358–365 (2002)
[18] Stützle, T., Hoos, H.H.: MAX-MIN Ant system. Future Generation Computer Systems 16(9), 889–914 (2000)
[19] Stützle, T., López-Ibáñez, M., Pellegrini, P., Maur, M., Montes de Oca, M., Birattari, M., Dorigo, M.: Parameter Adaptation in Ant Colony Optimization. In: Autonomous Search, pp. 191–215. Springer (2012)
[20] Stützle, T., Hoos, H., Merz, P.: An Analysis of the Hardness of TSP Instances for Two High-performance Algorithms. In: 6th Metaheuristics International Conference (MIC), pp. 361–367 (2005)
[21] Wilcoxon, F.: Individual Comparisons by Ranking Methods. Biometrics Bulletin 1(6), 80–83 (1945)