Using meta-learning to recommend meta-heuristics for the traveling salesman problem

Jorge Y. Kanda*†, Andre C.P.L.F. de Carvalho*, Eduardo R. Hruschka*‡ and Carlos Soares §

*Instituto de Ciencias Matematicas e de Computacao
Universidade de Sao Paulo, Sao Carlos, Brazil 13566-590
Email: {kanda,andre,erh}@icmc.usp.br
†Instituto de Ciencias Exatas e Tecnologias
Universidade Federal do Amazonas, Itacoatiara, Brazil 69100-000
‡University of Texas at Austin, USA
§LIAAD-INESC Porto LA/Faculdade de Economia
Universidade do Porto, Porto, Portugal 4050-190
Email: csoares@fep.up.pt

***Abstract*—Several optimization methods can find good solutions for different instances of the Traveling Salesman Problem (TSP). Since there is no method that generates the best solution for all instances, the selection of the most promising method for a given TSP instance is a difficult task. This paper describes a meta-learning-based approach to select optimization methods for the TSP. Multilayer perceptron (MLP) networks are trained with TSP examples. These examples are described by a set of TSP characteristics and the cost of solutions obtained by a set of optimization methods. The trained MLP network model is then used to predict a ranking of these methods for a new TSP instance. Correlation measures are used to compare the predicted ranking with the ranking previously known. The obtained results suggest that the proposed approach is promising.**

***Keywords*-Algorithm selection, meta-learning, traveling salesman problem, multilayer perceptron network.**

I. Introduction

Optimization problems may be found in several application domains, like transport, engineering, manufacturing, planning and economics [1]. The value of the optimal solution to an optimization problem is provided by the minimum (or maximum) value of an objective function given a set of constraints. A solution to an optimization problem is represented by a combination of values for the variables of the problem without violating the constraints [2]. A classic example of optimization problem is the well-known traveling salesman problem (TSP). A TSP instance can be informally posed as: given a set of cities along with the cost of travel between each pair of them, find the cheapest way of visiting all cities and then return to the initial city [3]. A TSP instance may be represented by a graph, which can facilitate the identification of some properties such as symmetry and level of connectivity. An instance of TSP is symmetric if the travel cost between any adjacent cities is the same independently of starting point, and is strongly connected if there is at least one connection for each pair of cities [2].

Exhaustive search methods may have a high computational cost when searching for the best solution for many instances of TSP. Strongly connected instances with n cities have $(n-1)!$ possible routes and n mathematical operations are processed for each route. For example, the cost of all routes of a strongly connected TSP with 50 cities could require 10^{64} operations, which would be concluded only after 10^{45} centuries if a computer with a processing capacity of one million operations per second is used.

This high computational cost can be significantly reduced by using heuristics. Heuristics can quickly find good local solutions, although not necessarily the global optimum solution. However, heuristics may be too problem-specific. This limitation can be overcome by meta-heuristics, which try to avoid local optimum values by using more general strategies to look for promising regions in the search space. Several meta-heuristics have found good solutions to the TSP, such as: Tabu Search (TS) [4], Greedy Randomized Adaptive Search Procedure (GRASP) [5], Simulated Annealing (SA) [6] and Genetic Algorithms (GA) [7].

Although different meta-heuristics have been successfully applied to instances of TSP, each meta-heuristics has a bias, which makes it more suitable for a group of instances [8]. Therefore, for every new TSP instance, the most promising meta-heuristic should be selected. This issue is a typical algorithm selection problem, originally presented by Rice [9]. In an ideal scenario, in which the resources (time, memory, processor, etc.) are unlimited, the best alternative would be to run all the meta-heuristics available and choose the one with the best solution. However, this alternative is often not feasible in practice due to its high computational cost. In this study, we investigate the use of meta-learning to induce a predictive model able to suggest the best meta-heuristics for new instances of TSP. Given this context, the main contributions of this paper are: (i) investigation of the use of meta-learning for metaheuristic selection, particularly for the TSP; (ii) presentation of a new set of meta-features; (iii) study the recommendation of a ranking of the best algorithms for new TSP instances; (iv) comparison of three meta-learning approaches through experiments.

The remainder of this paper is organized as follows. Section II briefly describes meta-learning in the context of algorithm selection. Some related works are discussed in Section III. To introduce the notation adopted, a brief description of the TSP is contained in Section IV. The experimental settings and the main results are detailed in Section V. Finally, the conclusions are presented in Section VI.

II. Meta-learning to select algorithms using multilayer perceptron networks

Meta-learning allows the induction of knowledge, which can be used to select the most promising algorithms for a given problem. It can be applied to different problem domains, such as machine learning (ML) or optimization [10].

In meta-learning, the data built from a group of previously processed problems are called meta-data. Each example in the meta-data is associated with one problem instance and has input meta-features and target meta-features. Input meta-features are original problem characteristics considered to be relevant to describe the instances. Target meta-features usually represent the performance obtained by a set of methods when applied to this problem. The examples described by input meta-features and target meta-features constitute the meta-data. In this work, classification algorithms use the meta-data to induce a classifier to predict, for a new problem instance, the most promising optimization methods. The reader interested in more details about meta-learning may consult [10].

The proposed approach of using meta-learning in the selection of meta-heuristics for the TSP is illustrated in Figure 1. Basically, the induction of a meta-learning model is divided into two phases. First, meta-data are extracted from a set of TSP instances through the measurement of a set of selected TSP properties, i.e. the meta-features, and the performance values obtained by different meta-heuristics when applied to each instance. The use of a set of meta-features that contain information about the performance of meta-heuristics to the problems is essential to the success of a meta-learning approach [10]. In the second phase, a predictive model is induced by multilayer perceptron (MLP) networks [11] to predict the most promising meta-heuristics through a ranking.

Here, the problem is addressed as a label ranking task. In label ranking, the learning problem is to map the instances x from a dataset X to rankings $\succ_x$ (total strict orders) over a finite set of labels $\mathcal{L} = \{\lambda_1, ..., \lambda_c\}$, where $\lambda_i \succ_x \lambda_j$ means that, for instance x, label λ_i is preferred to λ_j. A ranking over $\mathcal{L}$ can be represented by a permutation as there exists a unique permutation τ such that $\lambda_i \succ_x \lambda_j$ iff $\tau(\lambda_i) < \tau(\lambda_j)$, where $\tau(\lambda_i)$ denotes the position of the label λ_i in the ranking [12]. For more details about label ranking can be found in [13].

MLP networks are frequently applied to nonlinear problems. A MLP network has one input layer, one or more hidden layers and one output layer [11]. The architecture of the MLP network used for meta-heuristics selection is illustrated in Figure 2. The input values are the meta-feature values for a TSP instance. The output layer produces a ranking of optimization methods for this TSP instance. The method identified in the top position (i.e., rank 1) is the most promising for this instance. The prediction of the rank associated with each method is conducted by a non-linear regression function, which is a way to estimate the conditional expectation of the dependent variable, keeping the independent variables fixed [14].

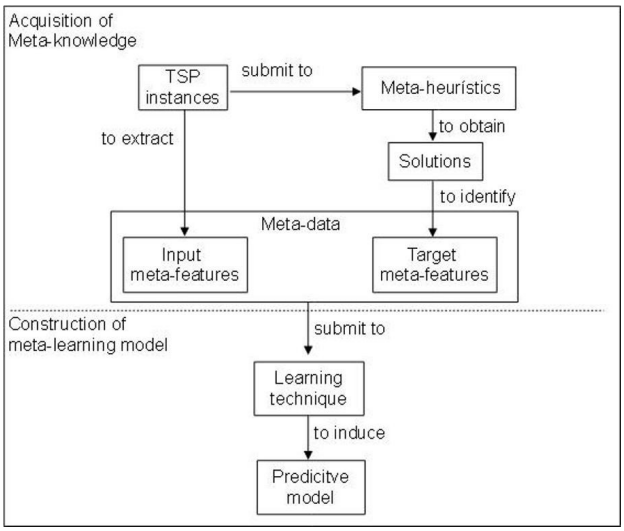

Fig. 1. Meta-learning process in the selection of algorithms for TSP.

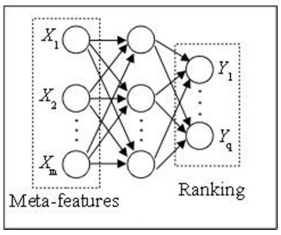

Fig. 2. MLP network architecture for algorithm selection.

In this work, each TSP instance is mapped to a ranking of meta-heuristics that are identified by labels set $\mathcal{L} = \{\text{TS, GRASP, SA, GA}\}$. For modeling the ranking learning problem, three MLP approaches were used. The first, called direct linear regression (DLR), uses a MLP with four neurons in the output layer, which identify the relative values corresponding to solutions of the metaheuristics for a TSP instance x submitted to a MLP input (Figure 3a). Based on the output values of the MLP, the order of the labels $\succ_x$ for the instance x is identified. The second approach, called single regression model (SRM), also uses a MLP with four outputs, but they indicate the position of the meta-heuristics in a ranking of recommendation (Figure 3b). Finally, in the third approach, called multiple regression model (MRM), based on the strategy for predicting the ranking classifiers [15], four models of MLP with one output neuron each (Figure 3c) are used to predict the position of a meta-heuristic in the ranking. The quality of the predictive model is measured by the correlation between the predicted and desired rankings.

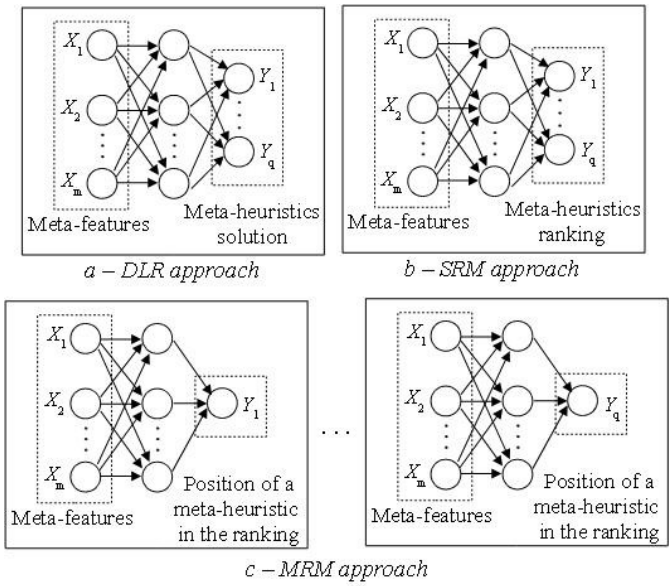

Fig. 3. MLP network architectures for meta-heurístic selection to TSP.

III. Related works

In recent years, meta-learning has been successfully applied to the selection of algorithms in various types of problems [10]. The meta-learning method described in [16] helps users to select Machine Learning (ML) algorithms for a given dataset. To that end, it takes into account the values of some meta-features, computed from available datasets, and then identifies and selects the dataset most similar to the one at hand. The final recommendation is then performed through the ranking of the ML algorithms for that selected dataset.

The idea of applying regression to meta-learning problems was first addressed in [17], where the error of a ML algorithm is predicted by using the meta-features defined in the STATLOG project. The authors evaluate several regression models to determine under what circumsntances a particular classification algorithm is applicable. Similarly, in [15] regression models is also used in the task of meta-learning, but to estimate the predictive accuracy of a classifier.

Although the research on meta-learning has been mainly focused on the selection of ML algorithms, some studies apply meta-learning in other areas such as optimization. In [18], the SATzilla approach selects optimization algorithms that solve examples of this type of problem. ML models are used to make predictions on the algorithms runtime for a particular problem. Such predictions are based on the meta-features of similar examples. In [19], a meta-learning model based on multilayer perceptron (MLP) networks is used to select optimization algorithms for the quadratic assignment problem. A meta-learning approach to recommend algorithms for TSP is described in [20], where TSP instances are classified according to the solutions obtained by a set of meta-heuristics. As the best solution for a given TSP instance may be achieved by more than one meta-heuristic, multi-label classification techniques are applied. Another work that applies meta-learning to the TSP is described in [21]. The meta-features are generated from two-dimensional location information of the cities. An MLP-based model is used to predict the effort that every available algorithm will need to find the best solution. Then, it assigns each instance to the most efficient algorithm.

MLPs have also been used for learning other types of rankings — usually in the Web search domains [22]–[24]. However, studies on recommendation algorithms for the TSP by means of ML techniques are recent [20], [21]. Our paper presents an approach to ranking learning based on MLPs that can recommend the most promising optimization methods for given TSP instances.

IV. The traveling salesman problem

Formally, a Traveling Salesman Problem (TSP) is defined by a graph $G = (V, E)$, in which $V = \{1, 2, ..., n\}$ is a set of vertices and $E = \{\langle i, j\rangle : i,j \in V\}$ is a set of edges. Each vertex $i \in V$ represents a city of the problem and each edge $\langle i, j\rangle \in E$ connects the vertices *i* and *j*. The cost of travel from the city *i* to city *j* is indicated by the value c_{ij} associated with the edge $\langle i, j\rangle$. If $c_{ij} = c_{ji} \; \forall \; \langle i, j\rangle \in E$, the TSP instance is symmetric. If $\exists \; \langle i, j\rangle \in E$ for each pair of vertices (i,j), the TSP instance is strongly connected. The best solution to a TSP is given by the Hamiltonian cycle of minimum cost. A cycle is Hamiltonian if all cities are visited only once and the route ends at the starting city [3].

Good TSP solutions can usually be obtained by means of meta-heuristics. In particular, they can provide solutions close to the optimal one and with relatively low computational cost. There have been several studies to improve existing meta-heuristics (*e.g.*, [25], [26]), as well as papers that compare the performance of different algorithms for the TSP [27]–[29]. In our work, four well-known meta-heuristics (TS, GRASP, SA, and GA) are used to illustrate the performance of our meta-learning model.

V. Experiments

To evaluate the proposed approach, a set of TSP examples have been used for learning models induced from three approaches of MLP described in Section II. The ranking predicted for an instance is evaluated by measuring its correlation to the known ranking for that instance. The quality of each model is the average rate correlation calculated in all meta-data instance.

A. Experimental setup

The correct prediction of a learning model depends, among other factors, on the availability of a significant amount of examples for training. Given that a larger number of TSP instances is not available, 2000 instances symmetric and strongly connected were generated from 10 files ($d1655$, $fl1400$, $nrw1379$, $pcb3038$, $pr2392$, $rl1889$, $u1817$, $vm1748$, $fnl4461$ and $rat783$) available in the TSPLIB library [30]. The file name indicates the number of cities involved in the problem (e.g. $d1291$ is a TSP with 1291 cities). For

each file, 200 different subproblems were generated from 100 cities selected randomly, keeping the original values of costs between the cities.

All 2000 instances were submitted to the meta-heuristics: Tabu Search (TS) [4], Greedy Randomized Adaptive Search Procedure (GRASP) [5], Simulated Annealing (SA) [6] and Genetic Algorithms (GA) [7]. The settings of each algorithm were: TS: size of the tabu list $= 3$; number of iterations with no improvement of the current solution $= 2$; GRASP: number of iterations $= 50$; $\alpha = 0.6$; SA: $\beta = 0.9$; $t_0 = 1$; $\alpha = 0.01$; GA: partial mapped crossover (PMX) [31] as the crossover operator; maximum number of individuals in the population $= 128$; tournament selection; mutation rate $= 5\%$; elitism selection. The selection of the parameters of the meta-heuristics was done taking into account that the goal of this work is not to optimize the performance of each meta-heuristic but to predict the ranking of the meta-heuristics. Therefore the parameters selection was based on a few preliminary experiments, just to ensure that each meta-heuristic could find a reasonable solution.

These meta-heuristics are stochastic and can provide different results after each run. Thus, for each TSP instance, every meta-heuristic was run 50 times (with different initial seeds), producing 50 solutions for each of them. The performance of each meta-heuristic was estimated by the best solution of those 50 runs. The same processing time was adopted as a stopping criterion for each of those 50 runs. The solutions for each instance were transformed into relative performance for the composition of a ranking of meta-heuristics. This ranking was used as the target meta-features by the meta-learning models in the SRM and MRM approaches. For the DLR approach, the target meta-features were represented by the solutions obtained by the meta-heuristics. Table I shows the relative frequency of the ranking of meta-heuristics. The most common ranking (GA, SA, GRASP and TS) was observed in 38.5% of the examples. This ranking majority is used in Section V-B as a reference for measuring the quality of the results of meta-learning models. For instance, the arbitrary choice of GA provides the best solution in 74.3% of TSP instances.

The identification of the most relevant properties of a set of problems is another key factor to improve the predictive ability of a meta-learning system. For the instances of TSP illustrated in this paper, some inherent properties of the corresponding graphs were transformed into meta-features. These properties are based on simple measurements and are extracted from the costs associated with edges and vertices. While the edge cost is easily identified in the graph, the vertex cost (c_i) is defined as the ratio between the sum of the costs of the edges associated with the vertex and the number (n) of edges connected to it:

$$c_i = \frac{\sum_{j=1}^{n} c_{ij}}{n} \tag{1}$$

Table II lists the 14 meta-features used to describe the TSP examples. The first seven meta-features are relative to the cost of vertices and the rest are related to the costs of edges. The meta-features V_{number} and E_{number} indicate respectively the number of cities and edges involved in the problem; V_{lowest} and $V_{highest}$ inform, respectively, the lowest and highest cost of vertice; Similarly, E_{lowest} and $E_{highest}$ represent the value of the edge of lower cost and greater, respectively; Values on the main statistical measures (mean, standard deviation and median) are obtained by the meta-features: V_{mean}, V_{sd} and V_{median} for the vertices and E_{mean}, E_{sd} and E_{median} for the edges; Finally, the meta-features V_{lower} and E_{lower} capture the sum of the V_{number} lower costs of vertex and edge, respectively. The meta-features identified as $\{F_8, ..., F_{14}\}$ and meta-feature F_1 are the same as those used in [20].

TABLE I
RELATIVE DISTRIBUTION OF RANKINGS OF META-HEURISTICS.

1o.	2o.	3o.	4o.	Frequence(%)
GA	SA	GRASP	TS	38.50
GA	SA	TS	GRASP	19.95
GA	GRASP	SA	TS	10.50
SA	GA	GRASP	TS	9.90
GRASP	GA	SA	TS	7.10
SA	GA	TS	GRASP	5.50
GA	TS	SA	GRASP	3.65
GRASP	SA	GA	TS	1.60
GA	TS	GRASP	SA	1.15
TS	GA	SA	GRASP	0.55
GA	GRASP	TS	SA	0.55
GRASP	GA	TS	SA	0.50
SA	GRASP	GA	TS	0.20
TS	SA	GA	GRASP	0.10
GRASP	TS	GA	SA	0.10
TS	GRASP	SA	GA	0.05
TS	GA	GRASP	SA	0.05
SA	TS	GA	GRASP	0.05

TABLE II
META-FEATURES FOR TSP USED THE TRAINING OF THE META-LEARNING ALGORITHM.

Code	Meta-features	Description
F_1	V_{number}	Number of vertex
F_2	V_{lowest}	The lowest cost of vertex
F_3	$V_{highest}$	The highest cost of vertex
F_4	V_{mean}	Average of the vertex costs
F_5	V_{sd}	Standard deviation of the vertex costs
F_6	V_{median}	Median of the vertex costs
F_7	V_{lower}	Sum of the V vertices of lower costs
F_8	E_{number}	Number of edges
F_9	E_{lowest}	The lowest cost of edge
F_{10}	$E_{highest}$	The highest cost of edge
F_{11}	E_{mean}	Average of the edge costs
F_{12}	E_{sd}	Standard deviation of the edge costs
F_{13}	E_{median}	Median of the edge costs
F_{14}	E_{lower}	Sum of the V_{number} edges of lower costs

Before using these meta-features as input parameters in a meta-learning model, the value of each meta-feature was normalized (A'_e) for each example of the set of meta-data using Equation 2.

$$A'_e = \frac{A_e - min_{A_e}}{max_{A_e} - min_{A_e}}, \tag{2}$$

where A_e is the value of the meta-feature in the example e; min_{A_e} and max_{A_e} represent the lowest and highest values, respectively, of the same meta-feature in the meta-data.

The value of each target meta-features was normalized (S'_{he}) for each example using Equation 3:

$$S'_{he} = \frac{S_{he}}{\sum_{r=1}^{q} S_{re}}, \quad (3)$$

where S_{he} is the performance (solution value or rank) of the meta-heuristic h on the example (i.e., TSP instance) e and q is the number of candidate meta-heuristics.

Each example of the set of meta-data of TSP is described by the normalized values of both meta-features and target meta-features. By following the standard practice of manipulating data in experiments with MLP network, the 2000 examples were randomly distributed and stratified into three datasets: training (60%), validation (20%) and test (20%). The first dataset is used by the learning algorithm to induce the predictor model; the second is to stop the training when a threshold value is reached and the third is to test the model generalization ability [32].

The MLP networks[1] were trained with the *backpropagation* algorithm [11] and cross-validation methodology [30] using the following settings: 14 neurons in the input layer, in which each neuron corresponds to a meta-feature extracted from the TSP; 4 neurons in the output layer for DLR and SRM approaches, and 1 output neuron for MRM approach. It is well-known that the best configuration of the hidden layer of an MLP is problem-dependent [32]. Therefore, we tried configurations with a number of neurons from 2 to 36. The configuration with the lowest mean squared error (MSE) on the validation set was then applied on the test set.

B. Experimental results

In our first experiment, the best parameter setting for each approach was identified from the performance of the MLP networks in the validation set. Table III shows that the best setting was quite different for each approach. The DLR approach had the smallest error due to its target meta-features, which represent the solution of the meta-heuristics. For each TSP example on the set of meta-data, the normalized values of the meta-heuristics solutions have become closer to each other than the normalized values corresponding to the ranks of the meta-heuristics.

TABLE III
LOWER VALUE OF MSE CALCULATED IN THE VALIDATION SET OF THREE LEARNING APPROACHES USING MLP.

Approach	MSE	Number of neurons in the hidden layer
DLR	8.562×10^{-5}	3
SRM	6.414×10^{-3}	19
MRM	7.307×10^{-3}	25

Next, we evaluated the best settings identified for each approach on the test dataset to measure the capacity of generalization of the models induced. The quality of prediction of these models was estimated using two measures of correlation: Spearman coefficient (SC) [33] and weighted Goodman-Kruskal coefficient (WGK) [34]. The first measure is basically the sum of squared rank errors and can be seen as the equivalent of normalized mean square error. For label ranking, the second measure is being used as a complement to SC since WGK correlates two sequences considering the magnitude of the values contained in them, calculating not only the trend of values but also the rate of variation of them. For these two measures, result +1 indicates a perfect correlation between the two sequences compared and -1 means that the sequences are completely discordant.

The MLPs performance measured by SC and WGK in each of the three learning approaches is shown in Tables IV and V, respectively. To assess the quality of the results, we compare the approaches proposed with suitable baselines. The performance of these methods serves as reference values (third column). The reference method used in this work consists of the most frequent ranking on the whole dataset. Since the semantic of the target meta-features is different in DLR approach, the magnitude of values is different in the desired ranking for this approach. Therefore, the reference value for WGK is different for DLR. The values presented in the fourth column of both tables indicate the degree of superiority of the meta-learning models in relation to a default prediction model whose performance is estimated by the reference value.

TABLE IV
PERFORMANCE OF THE META-LEARNING MODELS MEASURED BY SPEARMAN COEFFICIENT.

Approach	SC	Reference value (R)	(SC-R)/R
DLR	0.821	0.790	0.039
SRM	0.833	0.790	0.054
MRM	0.826	0.790	0.046

TABLE V
PERFORMANCE OF THE META-LEARNING MODELS MEASURED BY WEIGHTED GOODMAN-KRUSKAL COEFFICIENT.

Approach	WGK	Reference value (R)	(WGK-R)/R
DLR	0.811	0.752	0.078
SRM	0.835	0.648	0.289
MRM	0.760	0.648	0.173

The results shown in the tables indicate that the meta-learning models have a lower error rate in predicting the ranking of meta-heuristics for the TSP instances when compared with the standard recommendation of the majority ranking. Furthermore, these results suggest that the performance of the learning model was higher in the model (SRM) induced by the MLP model with 4 outputs, as shown by the values in the fourth column of Tables IV and V. In the DLR approach, learning the values of the solutions generated by optimization methods is more complex and has the disadvantage of being first necessary to order the values predicted before recommending the meta-heuristics for the user. The MRM approach has the cost of implementing as many networks as the number of meta-heuristics available to be applied to a given TSP and

[1]They were implemented in R software using the default values of the neuralnet package.

its performance was lower because there was no exchange of information between multiple networks.

VI. Conclusions

We presented an approach based on meta-learning to recommend meta-heuristics for Traveling Salesman Problem (TSP) instances. In particular, meta-learning models are induced by using multilayer perceptron (MLP) networks, which are trained from TSP instances for which the respective solutions of a set of meta-heuristics are *a priori* known. Our experimental evaluation shows that the proposed approach allows a suitable estimation of the ranking of the meta-heuristics for a given TSP instance. More precisely, the values for the correlation measures between predicted and known rankings show that our approach can provide better results than the *majority-based ranking*. As an additional contribution, we also introduced new meta-features, which can be seen as properties extracted from the graphical representation of the TSP instances. The identification of appropriate meta-features is essential to the success of a meta-learning system. In future work, we are going to investigate the extraction of different meta-features, such as those used in [21], in order to hopefully improve the prediction ability of our meta-learning model.

Acknowledgment

The authors acknowledge CAPES, CNPq, FAPESP and FAPEAM for their financial support. This work was partially supported by FCT projects Rank! (PTDC/EIA/81178/2006) and "Evolutionary algorithms for Decision Problems in Management Science" (PTDC/EGE-GES/099741/2008).

References

[1] M. Bazaraa, J. Jarvis and H. Sheralt. *Linear Programming and Networks Flows*. Wiley-Interscience, Hoboken, NJ, third edition, 2005.

[2] D. Bertsimas and J. Tsitsiklis. *Introduction to linear optimization*. Athena Scientific, Belmont, Mass, 1997.

[3] D. Applegate, R. Bixby, V. Chvátal and W. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton, 2006.

[4] F. Glover, E. Taillard and D. de Werra. "A user's guide to tabu search". *Annals of Operations Research*, vol. 41, pp. 3–28, 1993.

[5] T. Feo and M. Resende. "Greedy Randomized Adaptive Search Procedures". *Journal of Global Optimization*, vol. 6, pp. 109–133, 1995.

[6] S. Kirkpatrick, C. Gelatt and M. Vecchi. "Optimization by Simulated Annealing". *Science*, vol. 220, pp. 671–680, 1983.

[7] J. Holland. "Genetic Algorithms and the Optimal Allocations of Trial". *SIAM J. Comp.*, vol. 2, pp. 88–105, 1973.

[8] D. Wolpert and W. Macready. "No free lunch theorems for optimization". *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, 1997.

[9] J. Rice. "The Algorithm Selection Problem". *Advances in Computers*, vol. 15, pp. 65–118, 1976.

[10] P. Brazdil, C. Giraud-Carrier, C. Soares and R. Vilalta. *Metalearning: Applications to Data Mining*. Springer, Berlin, 2009.

[11] S. Haykin. *Neural Networks and Learning Machines*. Pearson Education inc, New York, N.Y., third edition, 2009.

[12] J. Fürnkranz, E. Hüllermeier, E. Loza Mencía and K. Brinker. "Multilabel classification via calibrated label ranking". *Mach. Learn.*, vol. 73, pp. 133–153, November 2008.

[13] S. Vembu and T. Gärtner. "Label Ranking Algorithms: A Survey". In *Preference Learning*, edited by J. Fürnkranz and E. Hüllermeier, pp. 45–64. Springer Berlin Heidelberg, 2011.

[14] G. Seber and C. Wild. *Nonlinear Regression*. John Wiley and Sons, New York, 1989.

[15] H. Bensusan and A. Kalousis. "Estimating the Predictive Accuracy of a Classifier". In *Proceedings of the 12th European Conference on Machine Learning*, edited by P. Flach and L. de Raedt, pp. 25–36. Springer, 2001.

[16] P. Brazdil, C. Soares and J. D. Costa. "Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results". *Machine Learning*, vol. 50, pp. 251–257, 2003.

[17] J. Gama and P. Brazdil. "Characterization of Classification Algorithms". In *Progress in Artificial Intelligence*, edited by C. Pinto-Ferreira and N. Mamede, pp. 189–200. Springer-Verlag, 1995.

[18] L. Xu, F. Hutter, H. H. Hoos and K. Leyton-Brown. "SATzilla: portfolio-based algorithm selection for SAT". *J. Artif. Int. Res.*, vol. 32, pp. 565–606, June 2008.

[19] K. Smith-Miles. "Towards insightful algorithm selection for optimisation using meta-learning concepts". In *Proceedings of the IEEE International Joint Conference on Neural Networks 2008*, volume 978, pp. 4118–4124, 2008.

[20] J. Kanda, A. Carvalho, E. Hruschka and C. Soares. "Selection of algorithms to solve traveling salesman problems using meta-learning". *International Journal of Hybrid Intelligent Systems*, vol. 8, no. 3, pp. 117–128, 2011.

[21] K. Smith-Miles, J. van Hemert and X. Lim. "Understanding TSP difficulty by learning from evolved instances". In *Proceedings of the 4th international conference on Learning and intelligent optimization*, volume 6073 of *LION'10*, pp. 266–280, Berlin, Heidelberg, 2010. Springer-Verlag.

[22] F. Scarselli, S. L. Yong, M. Gori, M. Hagenbuchner, A. C. Tsoi and M. Maggini. "Graph Neural Networks for Ranking Web Pages". In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, WI '05, pp. 666–672, Washington, DC, USA, 2005. IEEE Computer Society.

[23] L. Lu, R. Safavi-Naini, M. Hagenbuchner, W. Susilo, J. Horton, S. Yong and A. Tsoi. "Ranking Attack Graphs with Graph Neural Networks." In *ISPEC*, edited by F. Bao, H. Li and G. Wang, volume 5451 of *Lecture Notes in Computer Science*, pp. 345–359. Springer, 2009.

[24] S. L. Yong, M. Hagenbuchner and A. C. Tsoi. "Ranking Web Pages Using Machine Learning Approaches". In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 03*, pp. 677–680, Washington, DC, USA, 2008. IEEE Computer Society.

[25] J. Santos, F. L. Júnior, R. Magalhães, J. Melo and A. D. Neto. "A Parallel Hybrid Implementation Using Genetic Algorithms, GRASP and Reinforcement Learning for the Salesman Traveling Problem". In *Computational Intelligence in Expensive Optimization Problems*, edited by L. M. Hiot, Y. S. Ong, Y. Tenne and C.-K. Goh, volume 2 of *Adaptation, Learning, and Optimization*, pp. 345–369. Springer Berlin Heidelberg, 2010.

[26] P. Pop, O. Matei and C. Sabo. "A New Approach for Solving the Generalized Traveling Salesman Problem". In *Proceedings of the 7th international conference on Hybrid metaheuristics*, volume 6373, pp. 62–72, Berlin, Heidelberg, 2010. Springer-Verlag.

[27] T. Stützle, A. Grün, S. Linke and M. Rüttger. "A Comparison of Nature Inspired Heuristics on the Traveling Salesman Problem". In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, pp. 661–670, London, UK, 2000. Springer-Verlag.

[28] D. Johnson and L. McGeoch. "Experimental Analysis of Heuristics for the STSP". In *Local Search in Combinatorial Optimization*, pp. 369–443. Wiley & Sons, 2001.

[29] T. Öncan, I. Altinel and G. Laporte. "A comparative analysis of several asymmetric traveling salesman problem formulations". *Computers & Operations Research*, vol. 36, no. 3, pp. 637–654, 2009.

[30] G. Reinelt. "TSPLIB - A Traveling Salesman Problem Library". *Informs Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.

[31] D. Goldberg and R. L. Jr. "Alleles, loci, and the traveling salesman problem". In *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, edited by J. J. Grefenstette, pp. 154–159. Publishers Lawrence Erlbaum Associates, 1985.

[32] M. Smith. *Neural Networks for Statistical Modeling*. International Thomson Computer Press, Boston, 1996.

[33] C. Spearman. "The Proof and Measurement of Association between Two Things". *The American Journal of Psychology*, vol. 100, no. 3/4, pp. 441–471, 1987.

[34] R. Campello and E. Hruschka. "On comparing two sequences of numbers and its applications to clustering analysis". *Information Sciences*, vol. 179, no. 8, pp. 1025–1039, 2009.