Neurocomputing 205 (2016) 393–406

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Meta-learning to select the best meta-heuristic for the Traveling Salesman Problem: A comparison of meta-features

Jorge Kanda [a,*], Andre de Carvalho [b], Eduardo Hruschka [b], Carlos Soares [c], Pavel Brazdil [d]

[a] Instituto de Ciencias Exatas e Tecnologia, Universidade Federal do Amazonas Rua Nossa Senhora do Rosario, 3863, Tiradentes, 69103-128 Itacoatiara, Amazonas, Brazil
[b] Instituto de Ciencias Matematicas e de Computacao, Universidade de Sao Paulo Av. Trabalhador Sancarlense, 400, 13566-590 Sao Carlos, Sao Paulo, Brazil
[c] INESC TEC/Faculdade de Engenharia - Universidade do Porto, Porto, Portugal
[d] LIAAD-INESC Tec/Faculdade de Economia, Universidade do Porto, Porto, Portugal

ARTICLE INFO

Article history:
Received 15 October 2015
Received in revised form
16 February 2016
Accepted 7 April 2016
Communicated by K. Chan
Available online 12 May 2016

Keywords:
Meta-learning
Meta-features
Label ranking
Meta-heuristics
Traveling Salesman Problem

ABSTRACT

The Traveling Salesman Problem (TSP) is one of the most studied optimization problems. Various meta-heuristics (MHs) have been proposed and investigated on many instances of this problem. It is widely accepted that the best MH varies for different instances. Ideally, one should be able to recommend the best MHs for a new TSP instance without having to execute them. However, this is a very difficult task. We address this task by using a meta-learning approach based on label ranking algorithms. These algorithms build a mapping that relates the characteristics of those instances (i.e., the meta-features) with the relative performance (i.e., the ranking) of MHs, based on (meta-)data extracted from TSP instances that have been already solved by those MHs. The success of this approach depends on the quality of the meta-features that describe the instances. In this work, we investigate four different sets of meta-features based on different measurements of the properties of TSP instances: edge and vertex measures, complex network measures, properties from the MHs, and subsampling landmarkers properties. The models are investigated in four different TSP scenarios presenting symmetry and connection strength variations. The experimental results indicate that meta-learning models can accurately predict rankings of MHs for different TSP scenarios. Good solutions for the investigated TSP instances can be obtained from the prediction of rankings of MHs, regardless of the learning algorithm used at the meta-level. The experimental results also show that the definition of the set of meta-features has an important impact on the quality of the solutions obtained.

1. Introduction

The Traveling Salesman Problem (TSP) is one of the most intensively studied problems in combinatorial optimization and theoretical computer science. TSP has been used to represent applications from different domains, such as machine scheduling, DNA sequencing, transportation, and microchip manufacturing [1]. A TSP can be informally described as: given a set of cities and their respective pairwise distances, find the tour with the lowest possible cost that starts in one of the cities, visits all the other cities only once, and ends at the initial city [2]. The number of cities and how they are connected define different instances of the TSP. It is difficult to find a global optimal solution for a given TSP instance, since TSP belongs to the class of problems known as NP-hard [3]. Therefore, exhaustive search methods are not applicable to find the best solution for large TSP instances, due to their high computational cost. For example, there are approximately 1.22×10^{17} feasible solutions for a TSP with 20 cities. Thus, an exhaustive search to find a global optimum solution would take a long time.

The high computational cost involved in solving TSP problems can be significantly reduced by the use of Meta-Heuristics (MHs), which are often able to provide near-optimal solutions in reasonable time. MHs are high-level search strategies that guide the search to more promising regions of the solution space and try to escape from local optimal solutions [4]. Several MHs have been successfully used for TSP instances, including Tabu Search [5], Greedy Randomized Adaptive Search Procedure [6], Simulated Annealing [7], Genetic Algorithms [8], and Ant Colony Optimization [9].

In spite of their general success, different MHs have different biases, which make each of them more suitable for a particular class of instances [10]. Therefore, given a set of MHs and a new TSP instance to be solved, the best choice depends on the

* Corresponding author. Tel.: +55 92 35213519; fax: +55 92 35213603.
E-mail addresses: jkanda@ufam.edu.br (J. Kanda), andre@icmc.usp.br (A.d. Carvalho), erh@icmc.usp.br (E. Hruschka), csoares@fe.up.pt (C. Soares), pbrazdil@inescporto.pt (P. Brazdil).

http://dx.doi.org/10.1016/j.neucom.2016.04.027
0925-2312/

characteristics of the TSP instance. About a decade ago, the concept of hyper-heuristic was introduced as an alternative approach to the choice of MHs for solving optimization problems [11]. The key idea was to apply different MHs to different parts of the solution process according to the strength of each MH. Nevertheless, it is not possible to ensure the existence of interdependence between different parts of an optimization problem like the TSP. Furthermore, for each part of the problem, the most suitable MH must still be selected.

Mapping the characteristics of problem instances with the relative performance of a set of algorithms that can deal with these instances is a task that can be performed with meta-learning [12]. Meta-learning allows the selection of the most promising techniques using an inductive learning process. Recently, a meta-learning approach addressed the problem of recommending MHs for new TSP instances as a multilabel classification task [13]. However, when multiple MHs are recommended, the user is left without any guidance concerning which one should be initially tried. We address this problem by using a label ranking approach [14], which predicts a ranking of MH candidates, according to their expected performance for a new TSP instance. A ranking is more useful for the user, since it recommends the execution of the MHs in the predicted order until a satisfactory result is obtained.

Previous approaches have investigated only a single TSP scenario, usually containing only *symmetric and strongly connected* instances [15,13,16]. The TSP instance is symmetric if the cost of traveling from city i to the adjacent city j is equal to the cost of traveling from j to i; and it is asymmetric when the costs of traveling between cities i and j are different. The TSP instance is strongly connected when all cities are interconnected; and it is weakly connected if the TSP instance has at least one pair of cities that is not directly connected. In our study, we completed those earlier approaches by systematically investigating the four possible scenarios in terms of symmetry and connectedness: *symmetric and strongly connected*, *asymmetric and strongly connected*, *symmetric and weakly connected*, and *asymmetric and weakly connected*.

The main contributions of this work are:

- We show that meta-learning models can accurately predict rankings of MHs for different TSP scenarios.
- We investigate new meta-features for TSP to be applied to the meta-learning process.
- We offer evidence that good solutions for new TSP instances can be obtained from the prediction of rankings of MHs regardless of the learning algorithm used at the meta-level.

The remainder of this paper is organized as follows. A short description of the TSP is given in Section 2. The required background for meta-learning for algorithm selection and label ranking is provided in Section 3. Adaptations of some machine learning techniques to the label ranking problem are described in Section 4. Section 5 presents the sets of meta-features used to characterize the TSP instances. Practical application scenarios of interest are given in Section 6. Based on these scenarios, the experimental setting is detailed in Section 7, while the obtained results are reported and analyzed in Section 8. Finally, the main conclusions are presented in Section 9.

2. The Traveling Salesman Problem

Formally, the Traveling Salesman Problem (TSP) can be defined by means of a graph $G=(V,E)$, in which $V=\{v_1, v_2, \ldots, v_n\}$ is a set of vertices and $E=\{\langle v_i, v_j\rangle : v_i, v_j \in V\}$ is a set of edges. Each vertex $v_i \in V$ represents a city and each edge $\langle v_i, v_j\rangle \in E$ connects the vertices v_i and v_j. The cost associated with the edge $\langle v_i, v_j\rangle$ (i.e., the cost of traveling from city v_i to city v_j) is indicated by the value c_{ij}.

Given a TSP instance, the goal is to find the minimum value for Eq. (1):

$$\min \quad z=\sum_{i=1}^{n}\sum_{j=i}^{n} c_{ij}x_{ij} \tag{1}$$

subject to:

$$\sum_{i=1}^{n} x_{ij}=1 \quad \forall j \in V \tag{2}$$

$$\sum_{j=1}^{n} x_{ij}=1 \quad \forall i \in V \tag{3}$$

$$\sum_{i,j \in S} x_{ij} \leq |S|-1 \quad \forall S \subset V \tag{4}$$

$$x_{ij} \in \{0,1\} \quad \forall i \neq j \in V \tag{5}$$

If the tour traverses edge $\langle v_i, v_j\rangle$ then $x_{ij}=1$, and $x_{ij}=0$, otherwise.

The best solution for a TSP is given by the Hamiltonian cycle of minimum total cost. A cycle is Hamiltonian if all cities are visited only once and the route ends at the initial city [2]. Fig. 1 illustrates a TSP instance with six cities. The route given by the edges in bold indicates the optimal solution.

TSP instances can be characterized according to strength of their connections and to their symmetry. A TSP instance is strongly

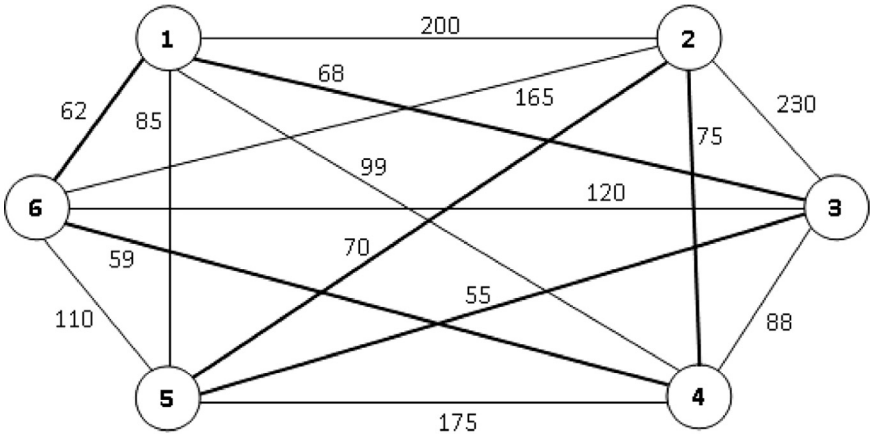

Fig. 1. A TSP instance and its optimal solution.

connected if $\forall(v_i, v_j)\exists\langle v_i, v_j\rangle \in E$. Otherwise, it is weakly connected. If $\forall\langle v_i, v_j\rangle \in E : c_{ij} = c_{ji}$, the TSP instance is symmetric. Otherwise, it is asymmetric. The TSP instance in Fig. 1 is symmetric and strongly connected.

The solution for a TSP instance can be identified by an adjacency matrix **A**, where $a_{ij}=x_{ij}$. For the TSP illustrated in Fig. 1, the solution can also be represented by the following adjacency matrix **A**.

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

In this adjacency matrix, the row number and the column number are vertex labels representing cities.

3. Meta-learning and label ranking

The selection of the best algorithm for a given task has been investigated in a sub-area of Machine Learning (ML) known as meta-learning [12]. Meta-learning studies how to improve the recommendation of the most promising techniques for a given task by learning from their previous use in related tasks. Meta-learning has been used to recommend algorithms for ML [17] and optimization [18] tasks. For example, the SATzilla method applies meta-learning to select algorithms to solve instances of the propositional satisfiability problems [19]. In [18], Multilayer Perceptron networks (MLPs) are meta-learners that recommend optimization algorithms to solve instances of the quadratic assignment problem.

Regarding the TSP, a meta-learning approach to recommend MHs is described in [13]. In their study, the authors addressed the recommendation of MHs as a classification task, in which the class associated with each instance is the MH that provides the best solution for the instance. As the best solution for a given TSP instance may be achieved by more than one MH, multilabel classification techniques were investigated. Another study that applies meta-learning in TSP is described in [15], where meta-features are generated from the two-dimensional location information of each city. The authors used MLP-based models to predict the search effort needed by different optimization algorithms to find their best solution. The search effort of a given algorithm is measured by the number of edges exchanged during the search for the best solution.

The general framework of a meta-learning approach to recommend MHs for TSP instances is illustrated in Fig. 2. Basically, the induction of the meta-learning model (meta-model) occurs in

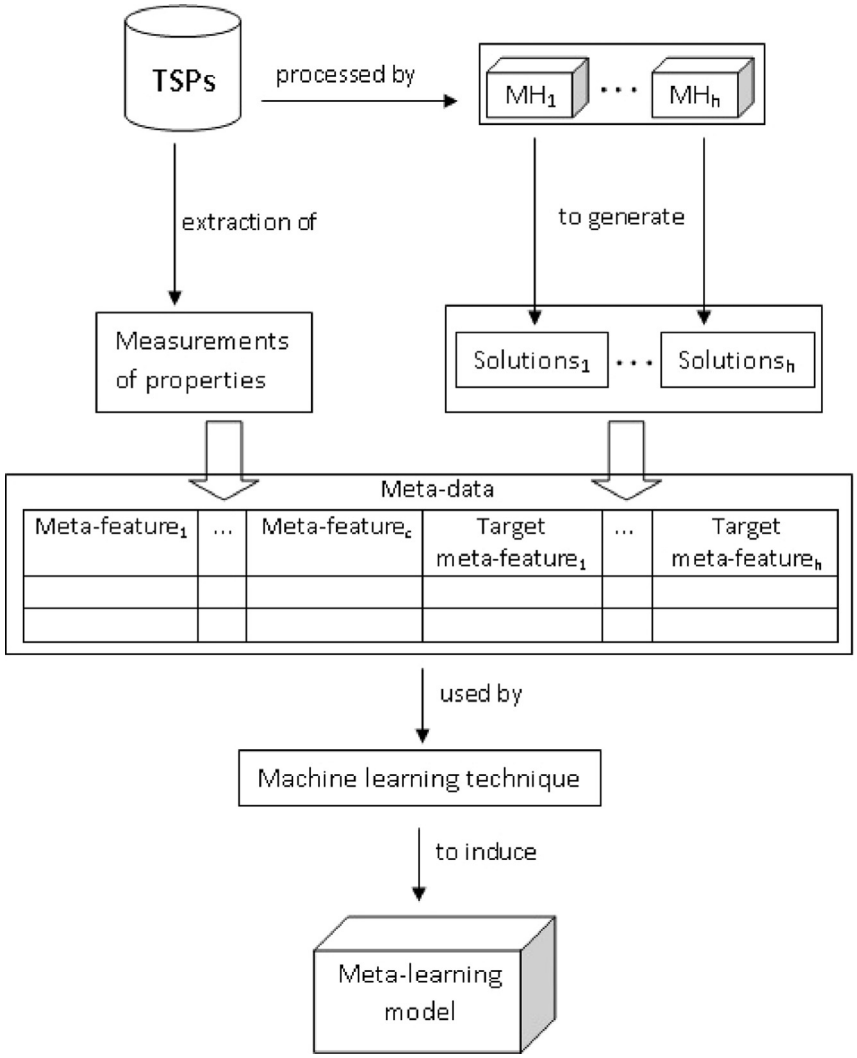

Fig. 2. Meta-learning approach to select meta-heuristics for the TSP.

two phases: (1) generation of meta-data and (2) creation of the meta-model. In the first phase, a set of TSP instances is used to generate the meta-data. The meta-data is stored in a matrix whose rows are meta-examples and columns are meta-features. A meta-example is a particular instance of TSP and the meta-features are attributes that represent relevant properties that describe those instances. Each meta-example is labeled with the quality of the solution obtained by different MHs when applied to the corresponding TSP instance. In the case of the TSP, it is the cost of the best solution obtained by the corresponding MH. In the second phase, a ML technique uses the meta-data to induce a meta-model. After the meta-model has been induced, it can be used to predict the performance value of the MHs for new TSP instances, which can then be used for MH recommendation.

The target of the meta-model can represent the performance of the MHs in different ways [12], like the cost of the solution obtained by each of the MHs. In this case, the meta-learning problem is addressed as multiple regression problems, with one target variable for each MH. Alternatively, the target may represent the best MH for the corresponding instance. If the target is a single MH, we have a traditional classification task. If the target consists of one or more MHs, representing the most promising methods for the corresponding instance, we have a multilabel classification task [13]. A drawback of using this approach for MH recommendation is that it does not inform the order with which the selected algorithms should be used. For instance, if the model recommends meta-heuristics MH1, MH3, and MH4 for a new problem and only one of them can be executed (e.g., due to the lack of computational resources), the user is still left with the decision of which of these three meta-heuristics to execute.

To overcome this difficulty, meta-learning can be used for a label ranking task [20]. In label ranking, the learning task is to map the instances x from a dataset X to rankings $\succ_x$ (total strict orders)[1] over a finite set of labels $\mathcal{L}=\{\lambda_1,\ldots,\lambda_q\}$, where $\lambda_i \succ_x \lambda_j$ means that, for the instance x, the label λ_i is preferred to λ_j. A ranking over $\mathcal{L}$ can be represented by a permutation, since there is only one 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. A survey on label ranking is provided in [21].

Evaluation of label ranking methods has issues similar to those found in other learning problems. Here we discuss the evaluation measures, baselines, and statistical validation of the differences in accuracy. The quality of label ranking methods is typically assessed using measures of ranking accuracy. The accuracy of a predicted ranking is measured by comparing it to the true ranking for the corresponding example. The comparison is often based on rank correlation coefficients, such as Spearman's correlation coefficient (r_S) [22]

$$r_S = 1 - \frac{6\sum_{i=1}^{q}(rr_i - ir_i)^2}{q^3 - q} \tag{6}$$

where rr_i and ir_i are the recommended and ideal ranks of algorithm i and q is the number of algorithms, respectively.

According to this coefficient, a value of $r_S = +1$ means full concordance — i.e. perfect correlation between the two rankings being compared — whereas $r_S = -1$ implies that the order of the labels between the two rankings is inverted. To estimate the accuracy of a label ranking algorithm, typical resampling procedures can be used, such as 10-fold cross validation [23]. As usual, the output is the mean of the ranking accuracies of all predictions.

To assess whether the learning algorithm is extracting any useful pattern, it is important to compare its accuracy with the accuracy of a simple baseline method. A typical baseline in label ranking is the average ranking on the whole dataset [17]. For each instance j from the dataset, a ranking of algorithms is computed. The average ranking position of each algorithm i is used to compose the baseline ranking br_i

$$br_i = \frac{\sum_{j=1}^{z}(ir_{ij})}{z} \tag{7}$$

Finally, it is necessary to assess whether the differences in accuracy between two different label ranking methods are statistically significant, using a suitable statistical test. In the experiments performed for this study, multiple methods are compared. Thus, to provide some reassurance about the validity and non-randomness of the obtained results, the outcomes of statistical tests, following the study of Demsar [24], are reported. Essentially, we compare multiple algorithms on multiple datasets by using the Friedman test, with a corresponding post hoc test. The Friedman test is a non-parametric statistic test equivalent to the repeated-measures ANOVA. If the null hypothesis, which states that the algorithms under study have similar performances, is rejected, then we proceed with the Nemenyi post hoc test for pair-wise comparisons between MHs.

4. Learning techniques for label ranking

Three adaptations of popular ML algorithms are used here for label ranking prediction: multi-layer perceptron (MLP) neural network trained by the backpropagation algorithm [25,26], the k-nearest neighbor (k-NN) algorithm [27,13], and the predictive clustering tree (PCT) induction algorithm [28].

A very simple adaptation of MLPs [13,26,16] is also adopted: it uses m MLPs to predict the rank of each one of the labels λ_i. The output values of the MLP networks are ordered to create the predicted ranking of the labels, indicating the most promising MHs for new TSP instances.

The employed adaptation of k-NN for label ranking was proposed in [12]. It is essentially the same k-NN used for classification or regression [23], except for the generation of the prediction process. It averages the rank of each label across the target rankings of the k training instances nearest to the new instance. The average ranks are ordered to obtain the predicted label ranking.

Our decision tree-based method for label ranking is based on the predictive clustering tree (PCT) [28]. In this adaptation of this decision tree induction algorithm for label ranking, two components are changed. The splitting criterion maximizes the gain in homogeneity of the target rankings in the nodes. Homogeneity is measured in terms of the average correlation between those rankings. The predicted ranking is generated like in the previous two algorithms.

We shall note that previous works [13,15] employed decision trees to recommend MHs. Other classifiers (SVM and Naive Bayes) were used in [13].

5. Meta-features for TSP

A key challenge for a meta-learning task is the characterization of the problem instances through suitable measures [29]. These measures, named meta-features, must include informative characteristics, i.e., properties of the instances that affect the performance of the MHs. As in any learning problem, the chances of inducing a good meta-model increase with the quality of the mapping between the characteristics of the problem instances and the performance of the MHs.

[1] The problem of label rankings can address less restrictive forms of rankings, such as partial orders [21].

Table 1
Meta-features based on edge and vertex measures (*EVM*).

Meta-features	Mathematical expression	Description
V_{number}	$length(V)$	Number of vertices (n)
C^V_{min}	$\min(C^V_1,\ldots,C^V_n)$	The lowest vertex cost
C^V_{max}	$\max(C^V_1,\ldots,C^V_n)$	The highest vertex cost
C^V_{avg}	$\frac{1}{n}\sum_{i=1}^{n} C^V_i$	Average vertex cost
C^V_{sd}	$\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(C^V_i - C^V_{avg})^2}$	Standard deviation of the vertex costs
C^V_{median}	$median(C^V_1,\ldots,C^V_n)$	Median of the vertex costs
C^V_{nn}	$\sum_{i=1}^{n}\min(c_{i1},\ldots,c_{in})$	Sum of the edge costs from each city to its nearest neighbor
E_{number}	$length(E)$	Number of edges (m)
C^E_{min}	$\min(C^E_1,\ldots,C^E_m)$	The lowest edge cost
C^E_{max}	$\max(C^E_1,\ldots,C^E_m)$	The highest edge cost
C^E_{avg}	$\frac{\sum_{i=1}^{m} C^E_i}{m}$	Average edge cost
C^E_{sd}	$\sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(C^E_i - C^E_{avg})^2}$	Standard deviation of the edge costs
C^E_{median}	$median(C^E_1,\ldots,C^E_m)$	Median of the edge costs
C^E_{lowest}	$\sum_{i=1}^{n} ord(C^E_i)$	Total cost of the n lowest edge costs

We investigate four sets of meta-features obtained by different approaches [13,26]. Since TSP instances can be represented as graphs, the first approach uses simple meta-features extracted from the graph structure representing an instance of TSP — see Section 5.1. The second set of meta-features represents TSP instance graphs by complex network characteristics (Section 5.2). The third proposal, named knowledge-based, extracts measures specific for the MHs used (Section 5.3). The final set of meta-features is based on a subsampling landmarkers approach [30] (Section 5.4).

Other meta-features were considered in the literature. For instance, features based on the two-dimensional location information of each city — such as the standard deviation of the pairwise distances between cities, distances from the nearest neighbor cities, the coordinates of the instance centroids, and other geometric measures, to name a few — were adopted in [15].

5.1. Edge and vertex measures (EVM)

This first group of meta-features, proposed in [26], extracts measures directly from a graph representing a TSP instance. Table 1 shows a simple set of *EVM* meta-features that describe properties from the edges and vertices of a graph. A simple meta-feature, the number of vertices, suggests the size of the search space. Since the meta-features investigated explore the neighborhood of promising solutions, properties associated with vertex cost and edge cost may provide important information regarding the most suitable search strategy for a particular graph.

The edges with lower costs are more likely to belong to the solutions generated by the MHs, so properties related to the costs of the edge (C^E) were also extracted. These measures can rapidly be obtained from a TSP graph. Another relevant information that we can extract from a graph is the vertex cost (C^V), which is measured as the average cost of the edges connected to the vertex. Other measures able to describe important features of the graph that represent a TSP were also used.

The route to an instance of TSP with n cities is composed of n adjacent cities. The cost of the best route is not smaller than the sum of n smallest costs of the edge. For this reason, the lower bound of the solution is also an important property to be identified. Given a set of values, the $ord(n)$ function returns n values in ascending order.

Table 2
Meta-features based on complex network measures (*CNM*).

MF	Mathematical expression	Description
AGD	$\frac{1}{m}\sum_{i\neq j}^{n} c_{ij}$	Average geodesic distance
GE	$\frac{1}{m}\sum_{i\neq j}^{n}\frac{1}{c_{ij}}$	Global efficiency
HM	$m\sum_{i\neq j}^{n}\frac{1}{c_{ij}}$	Harmonic mean of the geodesic distances
NV	$\max\{NV_1,\ldots,NV_n\};\ NV_i = \frac{GE - GE_i}{GE}$	Network vulnerability
CCT	$\frac{3N_\Delta}{N_3}$	Clustering coefficient for transitivity
ACC	$\frac{1}{n}\sum_{i=1}^{n}\frac{N_\Delta(i)}{N_3(i)}$	Alternative clustering coefficient
CCW	$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{s_i(m_i-1)}\sum_{j=1}^{n}\sum_{k=j+1}^{n}\frac{c_{ij}+c_{ik}}{2}a_{ij}a_{ik}a_{jk}$	Clustering coefficient for weighted networks
NCC	$\frac{1}{n}\sum_{i=1}^{n}\frac{2}{m_i(m_i-1)}\sum_{j=1}^{n}\sum_{k=1}^{n}\frac{1}{S_{ijk}}a_{ij}a_{ik}$	Network cyclic coefficient
MDV	$\max\{m_1,\ldots,m_n\}$	Maximum degree of the vertices
CED	$\frac{(1/m)\sum_{j>i} m_i m_j a_{ij} - [(1/2m)\sum_{j>i}(m_i+m_j)a_{ij}]^2}{(1/2m)\sum_{j>i}(m_i^2+m_j^2)a_{ij} - [(1/2m)\sum_{j>i}(m_i+m_j)a_{ij}]^2}$	Correlation of the degrees at both ends of the edges
EDD	$-\sum_{k=1}^{n} P(m_i)logP(m_i)$	Entropy of the degree distribution
TE	$-\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ji}c_{ij}log_2 c_{ij}$	Target entropy
PCV	$\frac{1}{n}\sum_{i=1}^{n} 1 - \sum_{s=1}^{y}\left(\frac{q_{is}}{n_i}\right)^2$	Participation coefficient of the vertices
ER	$\frac{1}{m}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}a_{ji}$	Edge reciprocity
CCA	$\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(a_{ij}-\overline{a})(a_{ji}-\overline{a})}{\sum_{i=1}^{n}\sum_{j=1}^{n}(a_{ij}-\overline{a})^2}$	Correlation coefficient of the adjacency matrix

5.2. Complex network measures (CNM)

The measures from the previous section are computationally very simple. However, they may not be able to capture important properties of the graph that represents a TSP instance. There is extensive research on more complex measures to characterize graphs, particularly in complex networks. A survey of measures used in the literature to characterize complex networks can be found in [31]. A new set of meta-features was created from these measures, here named *CNM* meta-features (Table 2).

Complex networks are commonly used to analyze graphs. The different variations of the TSP can be modeled using the resources available in a graph. Relevant information on the structure of a TSP, such as the distribution of connections between cities, can be extracted from the graph.

The first meta-feature, the average geodesic distance (*AGD*), measures the average distance between each pair of nodes. To capture information between vertices we compute by the global efficiency (*GE*) measure, which assumes that the efficiency for sending information between two vertices v_i and v_j is inversely proportional to their distance. The harmonic mean (*HM*) of the geodesic distances is also calculated.

Critical components of a network are directly related to the most vulnerable vertices. The vulnerability of a vertex i, named NV_i, can be measured by the decrease in efficiency of the network

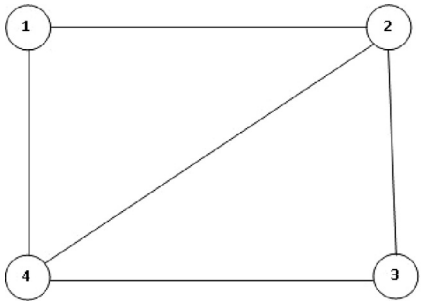

Fig. 3. Example of undirected and unweighted graph.

when the vertex i and all its edges are removed from this network (Eq. (8)). Network vulnerability is given by the maximum vulnerability of all its vertices (NV).

$$NV_i = \frac{1}{n-m_i}\left(\sum_{i\neq j}^{n}\frac{1}{c_{ij}} - \sum_{j=1}^{m_i}\frac{1}{c_{ij}}\right), \tag{8}$$

where m_i is the vertex degree that corresponds to the number of edges connected to the vertex i.

The clustering coefficient transitivity (CCT) depends on the number of triangles in the network (N_Δ) and the number of connected triples (N_3) calculated from (Eqs. (9) and 10), respectively

$$N_\Delta = \sum_{k>j>i} a_{ij}a_{ik}a_{jk}, \tag{9}$$

$$N_3 = \sum_{k>j>i} (a_{ij}a_{ik} + a_{ji}a_{jk} + a_{ki}a_{kj}), \tag{10}$$

where a_{ij} are elements of the adjacency matrix. The sum is taken over all triples of distinct vertices i, j, and k only once. One way of measuring the clustering coefficient is to calculate the average ratio between the number of triangles and the number of triples connected to each vertex (ACC).

In the graph in Fig. 3 there are two triangles $\Delta 124$ and $\Delta 234$. For each triangle, the value of N_3 is equal to 3, and for other combinations of three different vertices the value of N_3 is equal to 2.

For weighted graphs, we can use a specific clustering coefficient (CCW), which requires the sum of all the costs ($s_i = \sum_i^n c_{ij}$) from each vertex i. Another property of complex networks is related to the quality of the network cycles. The network cyclic coefficient (NCC) is calculated from the average of the cyclic coefficient of each vertex. The smallest cycle is a triangle ($S_{ijk} = 3$) but, as there is no loop passing through i, j, k, they are treelike connected ($S_{ijk} = \infty$).

The network in-degree is given by the highest vertex in-degree (MDV). The degree correlation (CED) is measured by the Pearson correlation coefficient. The heterogeneity of a network can be identified by the entropy of the degree distribution (EDD). Exchanges of messages can occur in a network, thus entropy is used to quantify the predictability of the message flow. Assuming that messages always flow through the shortest paths and all pairs of vertices exchange the same number of messages at same rate, the value of target entropy (TE) is a relevant information. Low values for this entropy mean that the vertex that will send the next message to vertex i can be easily predicted.

In a network, a set of vertices can be seen as a community that is generally defined in a strong or a weak sense. In a strong sense, a subgraph is a community if all of its vertices have more connections between them than with the rest of the network. In a weak sense, a subgraph is a community if the sum of all vertex in-degrees inside the subgraph is larger than outside it. In order to identify if the edges of a vertex i are "well distributed" among the different communities, the participation coefficient (PCV) can be calculated. For this purpose, the number of edges (q_{is}) from vertex i to community s and the degree of this vertex (m_i) are required. The coefficient value is equal to 0 if all edges are in their own community. The coefficient is equal to 1 if its edges are uniformly distributed among all communities.

Edge reciprocity is a measurement that helps us to characterize a network in terms of existence of undirected edges between two vertices. This measurement is captured by the ER meta-feature, which calculates the ratio between the number of undirected edges and the total number of edges (m). Finally, we used the mean value ($\overline{a}$) of the adjacency matrix to calculate the correlation coefficient (CCA) of the adjacency matrix, which is another way to measure the edge reciprocity.

5.3. Meta-heuristics properties (MHP)

We use five MHs: Tabu Search (TS) [5], Greedy Randomized Adaptive Search Procedure (GRASP) [6], Simulated Annealing (SA) [7], Genetic Algorithms (GAs) [8], and Ant Colony Optimization (ACO) [9]. They were chosen because they make different assumptions about the search space. Therefore, each one will be more suitable for some spaces. In this set of meta-features, we extract characteristics of the search space of TSP instances that are expected to affect the search procedures of these MHs.

Given its size, it is impossible to generate an accurate characterization of the search space for all but the most trivial of the TSP instances. Therefore, we estimate those characteristics on a sample of randomly generated solutions. The cost of computing these meta-features can thus be controlled by setting the number of solutions generated. The set of meta-features based on meta-heuristics properties, named MHP (Table 3), is defined according with the following notation:

- $S_i = \{s_i^1, \ldots, s_i^w\}$: set of all feasible solutions for the TSP instance i;
- $R_i = \{r_i^1, \ldots, r_i^y\}$: set of y randomly generated solutions such that $R_i \subset S_i$;

Table 3
Meta-features (MF) based on meta-heuristic properties (MHP).

MF	Mathematical expression	Description
QBN	$\frac{1}{v}\sum_{k=1}^{v} I(c(n_i^{k*}) < c(r_i^k))$	Expected proportion of neighbors with a better solution
RNS	$\frac{1}{v}\sum_{k=1}^{v} \frac{c(n_i^{k*})}{c(r_i^k)}$	Expected ratio between the costs of the best neighbor of a given solution
QNS	$\frac{1}{nv}\sum_{j=1}^{n}\sum_{k=1}^{v} I(c(n_i^j) < c(r_i^k))$	Quality index of the neighbors of the random solutions
QGS	$\frac{1}{ny}\sum_{j=1}^{n}\sum_{k=1}^{y} I(c(g_i^j) < c(r_i^k))$	Quality index of the greedy solution
RGR	$\sum_{j=1}^{n}\sum_{k=1}^{y} \frac{c(g_i^j)}{c(r_i^k)}$	Average ratio between the route cost of the greedy solution and the cost of the random solution
QBO	$\sum_{j=1}^{t} \frac{I(c(best(\check{r}_i^{j,1}, \check{r}_i^{j,2})) < c(best(\hat{r}_i^{j,1}, \hat{r}_i^{j,2})))}{t}$	Quality of the best offspring solution
RCP	$\frac{1}{t}\sum_{j=1}^{t} \frac{c(best(\check{r}_i^{j,1}, \check{r}_i^{j,2}))}{c(best(\hat{r}_i^{j,1}, \hat{r}_i^{j,2}))}$	Average ratio between the cost of the best offspring solution and the cost of the best solution of their parents
RRQ	$\frac{1}{4t}\sum_{j=1}^{t}\sum_{k=1}^{2}\sum_{l=1}^{2} I(c(\check{r}_i^{j,k})) < c(\hat{r}_i^{j,l}))$	Reproduction quality rate
ISE	$\frac{2}{n(n-1)}\sum_{j=1}^{n}\sum_{k=j+1}^{n} \frac{length(E_i^j \bigcap E_i^k)}{n}$	Index of shared edges
RFM	$\frac{1}{n}\sum_{j=1}^{n} I(e_{mc} \in g_i^j)$	Relative frequency of the most common edge in greedy solutions

- $G_i = \{g_i^1, \ldots, g_i^n\}$: set of n greedy solutions generated from n different cities such that $G_i \subset S_i$;
- $N_i^j = \{n_i^{j1}, \ldots, n_i^{jv}\}$: set of v neighbor solutions of s_i^j, where the neighbors are the solutions obtained by swapping two adjacent cities of s_i^j. The best neighbor solution (n_i^{j*}) of s_i^j is the solution with the lowest route cost among all its neighboring solutions, $c(n_i^{j*}) = \min\{c(n_i^{j1}), \ldots, c(n_i^{jv})\}$.

The first meta-feature (*QBN*) is the proportion of random solutions r_i^j in R_i that have a better solution in their neighboorhood N_i^j. The Indicator function $I(c(x) < c(y))$ returns 1 if the value of $c(x)$ is smaller than the value of $c(y)$ and returns 0 otherwise. The average ratio between the cost of the best neighbor solution and the generated solution is given by *RNS*. The expected quality of a randomly generated solution is measured by *QNS*. It is the average number of neighbors of the solutions r_i^j in R_i that are better than r_i^j. The *QGS* meta-feature compares the greedy solutions in G_i with the random solutions in R_i, returning the relative number of greedy solutions that are better than the random solutions. The average ratio between the cost of greedy routes and the cost of random routes is captured by *RGR*.

The next three meta-features (*QBO*, *RCP*, and *RRQ*) are related with properties of genetic algorithms. They are based on a set of t parent solutions, $\hat{R}_i = \{(\hat{r}_i^{1,1}, \hat{r}_i^{1,2}), \ldots, (\hat{r}_i^{t,1}, \hat{r}_i^{t,2})\}$. These solutions are randomly selected from R_i. Each pair of selected solutions, $(\hat{r}_i^{k,1}, \hat{r}_i^{k,2})$, is combined using a crossover operator — e.g. partial mapped crossover operator [32] — to generate two new solutions, $(\check{r}_i^{k,1}, \check{r}_i^{k,2})$. This results in a new set of offspring solutions, $\check{R}_i = \{(\check{r}_i^{1,1}, \check{r}_i^{1,2}), \ldots, (\check{r}_i^{t,1}, \check{r}_i^{t,2})\}$. The *QBO* meta-feature measures the average number of times that the best offspring solution in a pair is better than the best solution of the corresponding parents. This measure is calculated by using the $best(x,y)$ function, which returns x if $c(x)$ is smaller than $c(y)$ and y otherwise. The *RCP* meta-feature is the average of the ratio between the route cost of the best offspring solution in each pair and the route cost of the best of the corresponding parents. *RRQ* is the average number of times that an offspring is better than its parents.

The *ISE* meta-feature captures the information about the average number of edges shared among the solutions generated by the greedy procedure. For this meta-feature, $E_i^j = \{e_{i1}^j, \ldots, e_{in}^j\}$ is the set of edges that compose the route of the s_i^j solution in G_i, and *length* (X) is a function that returns the number of elements from the set X. Considering $q : E \rightarrow \Re$, a function that maps each edge to the number of solutions that use it, the most common edge, e_{mc}, is identified by $e_{mc} = \arg\max_{e_i}\{q(e_1), \ldots, q(e_m)\}$. Finally, the *RFM* meta-feature indicates the relative frequency of e_{mc}, in which the $I(e \in X)$ function returns 1 if $e \in X$ and 0 otherwise.

5.4. Subsampling landmarkers properties (SLP)

Landmarkers are fast estimates of algorithm performance on a given task. The estimates can be obtained by running simplified versions of the algorithms [12]. As an example, a small number of ants can be a landmarker for an ant colony optimization metaheuristic. Another way of obtaining fast performance estimates is to run the algorithms whose performance is to be estimated on a sample from the target task, obtaining the so-called *subsampling landmarkers* [12]. This method uses the sample size as a parameter. Some authors have used increasing sample sizes (partial learning curve), which has been claimed to produce better results [33].

As we used five MHs to investigate the predictive ability of the meta-models, five different *SLP* meta-features were extracted from subsampling landmarkers properties, as shown in Table 4. These meta-features are related with the algorithms performance after processing the simplified version of these MHs. The performance of these MHs were estimated on a subsample, sub(i), from each TSP instance i. In this study, performance means the value of the solution provided by an optimization algorithm.

Table 4
Meta-features (MF) based on subsampling landmarkers properties (*SLP*).

MF	Mathematical expression	Description
PTS	*tabu*(i)	solution value provided by TS fast estimate for sub(i)
PGR	*grasp*(i)	solution value provided by GRASP fast estimate for sub(i)
PSA	*sa*(i)	solution value provided by SA fast estimate for sub(i)
PGA	*genetic*(i)	solution value provided by GAs fast estimate for sub(i)
PACO	*ant*(i)	solution value provided by ACO fast estimate for sub(i)

6. Recommendation scenarios

Previous meta-learning approaches for TSP [15,13,26] have considered a single scenario: the recommendation of MHs for TSP instances that are *strongly connected and symmetric*. In this scenario, all cities are adjacent to each other and the cost of traveling from one city to an adjacent city is the same, regardless of the initial city. While this scenario is realistic in some tasks that can be represented as a TSP, such as optimizing the path of a computerized numerical control (CNC) machine to drill holes in a circuit board [34], it is not adequate for other tasks. For instance, in the original TSP scenario, there are no direct roads between all the cities. This means that the graph is not strongly connected. Additionally, if the transportation is by commercial flights and the cost function is the cost of the tickets, then the graph is not symmetric, as the price of flying from city A to city B is often different from the opposite flight. Therefore, it is important to consider different types of graphs. We consider three additional scenarios:

- *Weakly connected and symmetric* scenario: There is no edge between some pairs of cities and for each pair of connected cities A and B, the cost to go from A to B is equal to the cost to go from B to A.
- *Strongly connected and asymmetric* scenario: Although there is an edge for each pair of cities, the cost to go from A to B can be different from the cost to go from B to A.
- *Weakly connected and asymmetric* scenario: There is no edge between some pairs of cities and for each pair of connected cities A and B, the cost to go from A to B can be different from the cost to go from B to A.

Fig. 4 shows examples of graphs representing these four TSP scenarios. It can be seen that in symmetric scenarios (Fig. 4a and 4c), the connection between two vertices occurs by an undirected edge, while directed edges connect vertices in asymmetric scenarios (Fig. 4b and 4d).

The best solution for the illustrated instances is a route that visits all cities of the problem in the following order: A, B, D, C, and A. In this case, the route is a Hamiltonian cycle whose cost is equal to 14.

7. Experimental setup

Our datasets have been generated from some benchmark instances of TSP found in the TSPLIB library [35]. Due to computational

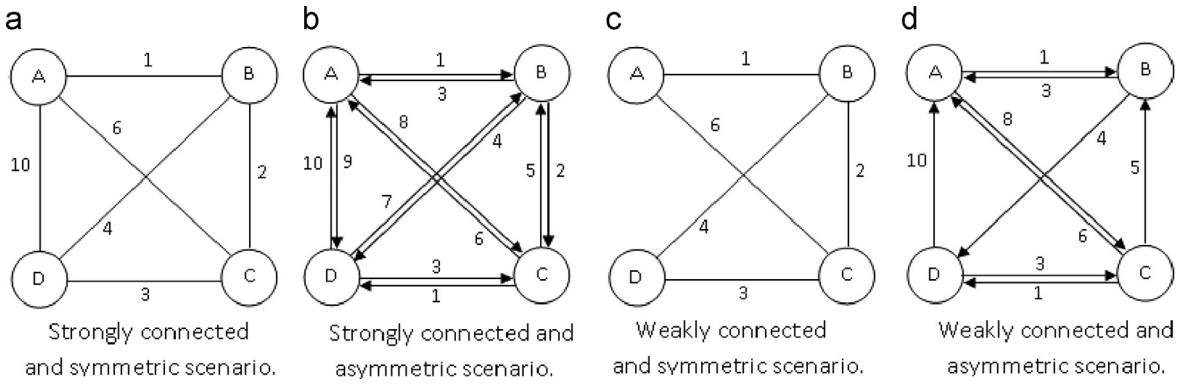

Fig. 4. Different TSP scenarios, varying the connectedness and symmetry of the graph.

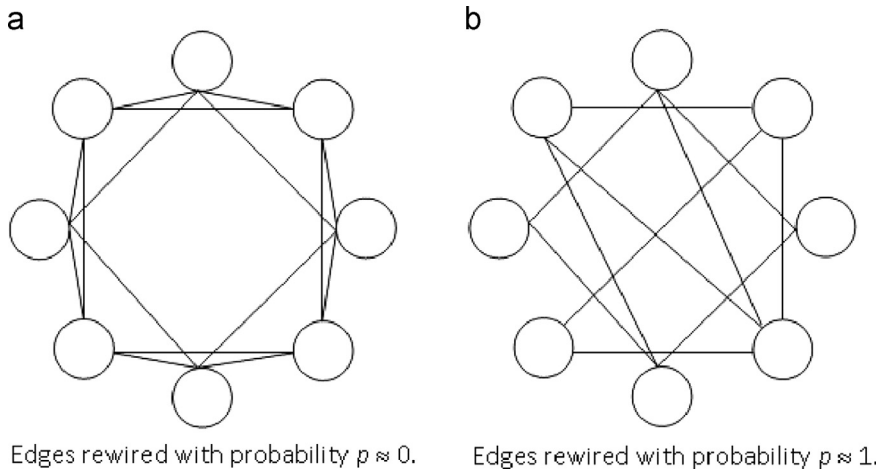

Fig. 5. Construction of a small-world network from a regular network. Figure adapted from [31].

limitations to run our experiments,[2] we did not use instances with a large number of cities in the experiments. For each scenario, the meta-data has 600 meta-examples, corresponding to 600 subproblems generated from four TSP files. From each TSP file, 150 different subproblems (subset of instances) of the same size were generated. For the symmetric scenarios, the files *eil*76, *ch*150, *lin*318, and *u*724 were chosen to generate subproblems with 10, 25, 50, and 100 cities, respectively. For the asymmetric scenarios, the files *p*43, *kro*124*p*, *ftv*170, and *rbg*443 were used for the generation of subproblems with 10, 25, 50, and 100 cities, respectively.

The subproblems in the weakly connected scenarios (symmetric and asymmetric) were generated from the strongly connected subproblems by applying the small-world model [36]. In a network generated with the small world model, most of the vertices can be reached from all other vertices using a small number of edges. In our experiments, the network was built connecting each vertex with its k-nearest neighbors, where the value of k corresponds to 60% of the number of cities of the problem. Then, each edge was randomly reconnected with probability $p=0.1$. When this parameter is equal to 0, the network has an ordered structure with a large number of loops constituted by three vertices. On the other hand, if $p=1$ then the network is a random graph with short distances and few loops. The values of k and p were chosen according to [31]. Fig. 5 shows the construction of a small world network from a regular network.

[2] Recall that for training a meta-model a dataset with many instances is necessary. Besides, the solutions for the meta-heuristics must be known for every instance.

As previously mentioned, five meta-heuristics were used in our experiments: TS, GRASP, SA, GAs, and ACO. After preliminary experiments with different parameter values, the following parameter settings were adopted:

- TS: tabu list size=2; number of iterations with no improvement of the current solution=2;
- GRASP: number of iterations=10; level of randomness and greedy search=0.5;
- SA: initial temperature=0.1; temperature increase rate=0.1; acceptance rate of neighbor solution=0.9; cooling rate=0.01;
- GA: crossover operator=partial mapped crossover (PMX) [32]; population size=20; tournament selection; mutation rate=0.05; elitism rate=0.5;
- ACO: number of ants=5; pheromone evaporation rate=0.5; pheromone influence=1; heuristic information influence=1.

We shall stress that our main goal was neither to optimize the performance of each MH, nor to empirically compare their relative performances. We are not trying to promote any particular MH. Instead, the focus is on the prediction of the ranking of MHs, with particular emphasis on how the user can take advantage of this ranking to obtain better solutions for new TSP instances.

Since these MHs are stochastic, different results can be obtained for each run. Thus, for each TSP instance, every MH was run 30 times (with different initial seeds), producing 30 solutions each. The same maximum processing time was adopted as a stopping criterion for each of these 30 runs of the MHs. This criterion has two main advantages: (1) it simulates a scenario in which the amount of time available to obtain a solution is limited

and known a priori; (2) the same opportunity is given to all MHs to search for a good solution. The average cost of the route obtained from those 30 solutions was adopted as the performance of each MH, which was used to build the target ranking of MHs for each instance.

To estimate the predictive performance of the meta-learning approach, ten-fold cross-validation was used. MultiLayer Perceptron (MLP), K-Nearest Neighbor (K-NN), and Decision Tree (DT) were implemented[3] from the packages nnet, FNN, and Clus, respectively. For the MLP networks, we used brackpropagation with momentum and 1 hidden layer with the number of neurons set to half of the number of predictive attributes and neurons in the output layer [37].

To generate the *SLP* meta-features from fast executions of a simplified version of the MHs, we used the same settings that were previously described, except for the following parameters:

- TS: number of neighbors=2; number of iterations with no improvement of the current solution=1;
- GRASP: number of iterations=1;
- SA: initial temperature=1; cooling rate=0.5;
- GA: population size=4;
- ACO: number of ants=2.

8. Experimental evaluation

This section presents the experimental results obtained by comparing the rankings recommended using different sets of meta-features and different label ranking methods in the four application scenarios presented earlier.

The average rank of each MH that generates a solution for different TSP instances in our experiments can be viewed in Table 5. Note that there is a variation in the rank of a particular MH, indicating that the computational model has to learn about different rankings.

Table 6 summarizes the data obtained from the performance of MHs in different scenarios of TSP. Higher performance variations occurred in weakly connected scenarios due to lack of some edges that influence the search for solutions. Although there was a diversity of rankings, it is important to analyze the predominance of the majority ranking. In our experiments, the frequency of the majority ranking ranged from 25% to 46%. These results point out the difficulty of learning the predictive models. Among the rankings obtained a uniform distribution was not observed as shown by the standard deviation of their frequencies.

8.1. Strongly connected and symmetric graph scenario

Table 7 shows the ranking accuracy ($\overline{r_S}$) of the predictive models for each set of meta-features. According to the statistical test, all learning models induced from different ML techniques and different sets of meta-features had better predictive performance than the baseline ($\overline{r_S} = 0.89$). Note that this represents a high accuracy ranking.

These values suggest that the meta-models induced from the same ML technique have similar predictive performance regardless of the set of meta-features used. The comparison between different algorithms leads to similar conclusions, as the differences are very small.

[3] Using the default values for their parameters.

Table 5
Average rank and standard deviations of MHs for different TSP scenarios.

Graph scenario	TS	GRASP	SA	GA	ACO
Strongly connected and symmetric	4 ± 1	1 ± 0	4 ± 1	3 ± 1	2 ± 1
Strongly connected and asymmetric	3 ± 2	2 ± 1	4 ± 1	3 ± 1	4 ± 2
Weakly connected and symmetric	3 ± 1	3 ± 1	3 ± 1	3 ± 1	1 ± 0
Weakly connected and asymmetric	3 ± 1	3 ± 1	3 ± 1	3 ± 1	1 ± 1

Table 6
Metadata of the MHs ranking for different TSP scenarios.

Graph scenario	Number of different rankings	Frequency of the majority ranking	Standard deviation of the frequencies of rankings
Strongly connected and symmetric	10	227/600	77
Strongly connected and asymmetric	18	150/600	42
Weakly connected and symmetric	26	181/600	42
Weakly connected and asymmetric	58	275/600	36

Table 7
Spearman Correlation coefficient for different sets of meta-features using meta-models induced by different learning algorithms in the strongly connected and symmetric graph scenario.

Set of meta-features	MLP	K-NN	DT
EVM	0.96 ± 0.08	0.95 ± 0.09	0.95 ± 0.09
CNM	0.96 ± 0.09	0.94 ± 0.10	0.96 ± 0.07
MHP	0.94 ± 0.11	0.95 ± 0.07	0.96 ± 0.09
SLP	0.95 ± 0.09	0.95 ± 0.10	0.96 ± 0.07

Table 8
Spearman Correlation coefficient for different sets of meta-features using meta-models induced by different learning algorithms in the strongly connected and asymmetric graph scenario.

Set of meta-features	MLP	K-NN	DT
EVM	0.94 ± 0.12	0.93 ± 0.12	0.94 ± 0.12
CNM	0.92 ± 0.17	0.93 ± 0.14	0.94 ± 0.12
MHP	0.91 ± 0.17	0.93 ± 0.13	0.93 ± 0.13
SLP	0.93 ± 0.14	0.93 ± 0.15	0.93 ± 0.14

8.2. Strongly connected and asymmetric graph scenario

In this scenario, the predictive accuracy of the baseline ranking is lower ($\overline{r_S} = 0.59$). This is explained by the higher diversity of the target rankings, which indicates that the problem is harder to learn than the problem from Section 8.1. However, the experimental results in Table 8 are similar to those obtained in the previous set of experiments indicating that, even in a more difficult scenario, models with good predictive abilities were found. As in the previous experiment, the results are very similar both across different sets of meta-features and learning algorithms.

8.3. Weakly connected and symmetric graph scenario

As in the first scenario, the accuracy of the baseline ranking is high ($\overline{r_S} = 0.81$). This occurred because several meta-heuristics were not always able to find a feasible solution for weakly connected TSP instances. In the experiments, MH5 found most often a feasible solution. Nevertheless, meta-learning was also able to find

Table 9
Spearman Correlation coefficient for different sets of meta-features using meta-models induced by different learning algorithms in the weakly connected and symmetric graph scenario.

Set of meta-features	MLP	K-NN	DT
EVM	0.93 ± 0.14	0.93 ± 0.14	0.93 ± 0.15
CNM	0.93 ± 0.14	0.93 ± 0.14	0.93 ± 0.13
MHP	0.92 ± 0.16	0.93 ± 0.14	0.92 ± 0.15
SLP	0.92 ± 0.18	0.92 ± 0.16	0.93 ± 0.15

Table 10
Spearman Correlation coefficient for different sets of meta-features using meta-models induced by different learning algorithms in the weakly connected and asymmetric graph scenario.

Set of meta-features	MLP	K-NN	DT
EVM	0.83 ± 0.27	0.85 ± 0.25	0.86 ± 0.25
CNM	0.83 ± 0.29	0.84 ± 0.26	0.86 ± 0.24
MHP	0.80 ± 0.33	0.83 ± 0.28	0.86 ± 0.23
SLP	0.78 ± 0.27	0.79 ± 0.27	0.80 ± 0.24

models with a superior predictive accuracy, as shown in Table 9. As in the previous experiments, the differences are statistically significant and the results are very similar both across different sets of meta-features and learning algorithms.

8.4. Weakly connected and asymmetric graph scenario

In this scenario, the variance in the results is higher than in the previous scenarios, as shown in Table 10. The baseline performance is slightly lower than in the previous scenario, ($\overline{r_S} = 0.77$), but is still high. In general, meta-learning obtained higher accuracy than the baseline, although the gain is small in some cases. The results of the statistical tests indicate the existence of significant differences between the baseline and the meta-learning approach only for the following meta-models: MLP with *MHP* meta-features, MLP with *SLP* meta-features, and K-NN with *SLP* meta-features.

Table 10 also shows that the variance in the accuracy within the same experiment is also higher than in the previous experiments. The values of the standard deviation indicate that there is a high variation in the accuracy predictions by the induced meta-models. This is likely due to the weakly-connected nature of the instances. The lack of edges directly influences the optimization performance of the MHs. This is particularly important for those that search for a good solution from random initial solutions. With fewer connections, the path is probably longer and harder to find. The diversity in the solution quality was observed particularly for instances with small number of cities.

The results obtained with different sets of meta-features lead to interesting observations. The *SLP* meta-features obtained the worst results, with statistical significance, regardless of the ML technique used. This can be explained by the impossibility, for many TSP instances, to estimate the solutions by executing the MHs on problems defined by subgraphs of the original instance. If the original graphs are weakly connected, which makes it hard for the MHs to find solutions, the problem becomes more difficult in subgraphs of those instances.

Considering the set of *MHP* meta-features, some of them require generating random solutions and comparing them to their neighboring solutions. For weakly connected TSP instances, many random solutions and their neighboring solutions are not feasible. This problem affected mainly the MLP algorithm. This seems to be caused by some particularly poor predictions, as indicated by a higher than usual predictive variance. The statistical test confirms this observation by detecting a significant difference between the meta-features *EVM* and *MHP* when they were used with the MLP algorithm.

8.5. Computational effort

For the TSP scenarios discussed here, the meta-models presented high predictive performance regardless of the set of meta-features investigated. Since the predictive performances obtained by the learning algorithms were similar, it is important to investigate the computational cost to generate each set of meta-features. This cost should be lower than the cost to run all the MHs, otherwise running all of them would be the best strategy for algorithm selection.

The average computational time to run all the five available MHs is adopted as a reference value to evaluate the processing time for each set of meta-features. The computational time saved by generating the meta-features with respect to running all the MHs is shown in Tables 11–14.

To understand the values in the tables, consider the hypothetical situation in which the time to run all MHs for a given TSP instance is equal to 5 s, while the time to generate the set of meta-features for the same instance is 1 s. In this case, the use of meta-learning represents a saving of 80% in computational time. A negative value shows that running all MHs is faster than computing the set of meta-features. For example, if the time necessary to generate the set of meta-features in the previous example is 10 s, there was an increase equivalent to 100% in processing time. In this case, we will adopt a negative signal notation (-100%).

According to Table 11, the *EVM* meta-features have a low processing cost, saving, in the worst case, 80% of the time necessary to run all MHs. This is expected as these are the simplest measures investigated.

For the set of *SLP* meta-features, significant gains were not observed in the processing time for small TSP instances, probably because good solutions can be quickly found by the MHs when applied to the original version of the instances. There is some gain in computational time for larger instances but there is not a clear trend. This may be due to the constructive meta-heuristics considered (e.g. MH2), which require almost the same time to build a solution regardless of the version of the instance used (original or simplified).

The runtime to generate the set of *MHP* meta-features largely increases with the number of cities. This occurs because the

Table 11
Average relative time saved in generating meta-features instead of running all the candidate meta-heuristics in the strongly connected and symmetric graph scenario.

TSP size	*EVM* (%)	*MHP* (%)	*SLP* (%)	*CNM* (%)
10 cities	93	81	0	77
25 cities	93	44	12	59
50 cities	92	−31	26	23
100 cities	82	−499	19	−226

Table 12
Average relative time saved in generating meta-features instead of running all the candidate meta-heuristics in the strongly connected and asymmetric graph scenario.

TSP Size	*EVM* (%)	*MHP* (%)	*SLP* (%)	*CNM* (%)
10 cities	93	85	0	77
25 cities	92	47	15	51
50 cities	92	3	26	20
100 cities	80	−394	22	−274

complexity of some of these meta-features is quadratic with the number of cities. Thus, this set of meta-features would not be a good choice for many TSP instances. A similar behavior can be observed with the *CNM* set of meta-features. The justification is the same as some of them are also quadratic on the number of cities. Thus, these approaches would not be good choices to extract meta-features for the recommendation task. Similar results were obtained for the analysis of the computational times for the *strongly connected and asymmetric* graph scenario, which are shown in Table 12.

Table 13
Average relative time saved in generating meta-features instead of running all the candidate meta-heuristics in the weakly connected and symmetric graph scenario.

TSP Size	*EVM* (%)	*MHP* (%)	*SLP* (%)	*CNM* (%)
10 cities	97	90	11	88
25 cities	97	66	14	75
50 cities	96	20	28	52
100 cities	91	−277	20	−109

Table 14
Average relative time saved in generating meta-features instead of running all the candidate meta-heuristics in the weakly connected and asymmetric graph scenario.

TSP Size	*EVM* (%)	*MHP* (%)	*SLP* (%)	*CNM* (%)
10 cities	97	90	12	89
25 cities	96	66	14	77
50 cities	97	28	23	59
100 cities	92	−228	18	−83

The results for the two scenarios with weakly connected TSP instances are presented in Tables 13 and 14. The gain in time (positive values) were similar in both scenarios, no matter the set of meta-features or TSP size. It is worth noticing that the set of *EVM* meta-features, which already had the best result in the previous scenarios, obtained computational cost savings always higher than 90%. In these scenarios, the absence of edges between cities leads to the use of a few values to compute the measures based on the edge and vertex cost. Consequently the generation of the *EVM* meta-features was faster. Additionally, due to reduction in the number of edges, the *MHP* and *CNM* sets of meta-features were also computed faster in the weakly connected scenarios than in the strongly connected scenarios.

In all the TSP scenarios, the average time required to calculate the *SLP* meta-features was similar, except for small instances belonging to weakly connected scenarios. In these cases, the computational time required was smaller, due to the few feasible solutions that could be found in the search space of the subgraph of the TSP instance.

The experimental results show that only the *EVM* and *SLP* sets of meta-features had lower runtimes than the execution of all MHs for the different sizes of TSP instances. On the other hand, the negative values obtained with the other two sets indicate that they should not be used to induce a meta-model for MH recommendation for a new TSP instance.

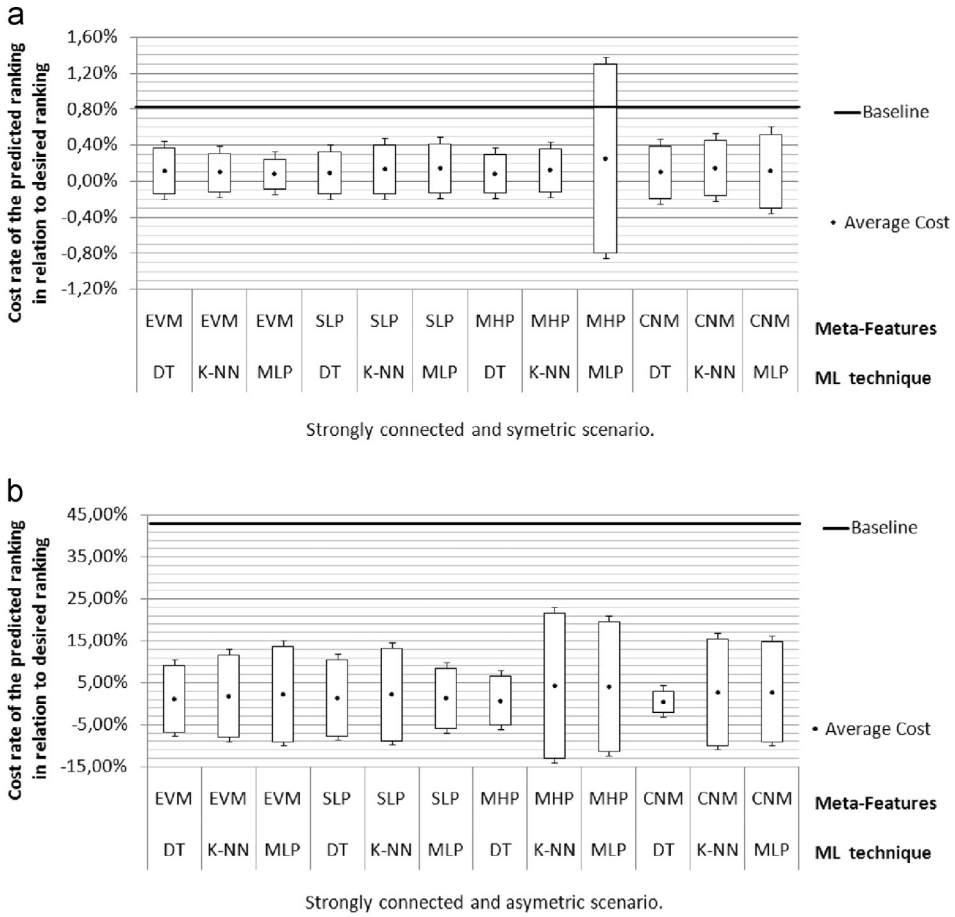

Fig. 6. Cost rate when using the predicted ranking of MHs regarding the ideal ranking for a strongly connected TSP instance.

8.6. Analysis of the recommended solutions

Although the meta-models were able to predict rankings of MHs more similar to the ideal ranking than the baseline ranking, it is also important to evaluate the magnitude of the solutions (cost of routes) generated by the ranked MHs. This evaluation can be carried out using the desired ranking and the value of the solutions provided by MHs as a reference for each TSP instance. If the highest ranked recommended MH does not match the MH in the first position of the desired ranking, the route obtained will have a cost higher than the route associated with the first suggested MH. To illustrate this situation, consider an example in which the ideal ranking $\mathcal{L}_i = \{MH_1, MH_2, MH_3, MH_4\}$ has the following values: $MH_1 = 100, MH_2 = 120, MH_3 = 130, MH_4 = 150$. If the recommended ranking is $\mathcal{L}_i = \{MH_2, MH_1, MH_4, MH_3\}$, the user will have a higher cost solution when using the MH in the first ranking position (120 instead of 100). The MH in the third position will also produce a higher cost solution (150 instead of 130). The sum of the differences, in this case the resulting value is 40, represents the extra cost by following the recommended ranking, and this is equivalent to 8% (40/500) of the total costs of the solutions generated by the MHs.

Figs. 6 and 7 illustrate the results of the additional cost rates of the solutions generated by MHs recommended by the meta-models. The closer the average cost is to zero percent, the lower the rate of extra cost if the ranking recommended by the meta-model is adopted. All meta-models presented computational cost lower than the baseline in the symmetric and strongly connected scenario (Fig. 6a). In this scenario, the extra cost ratio did not exceed 0.25%. The meta-model induced from an MLP with EVM meta-features presented the best performance (average rate = 0.08% and standard deviation = 0.17%). The standard deviation of the results generated by the MLP with MHP meta-features was higher than the other learning algorithms and meta-features because the most recommended MH for some instances of TSP were not in the top positions of the desired ranking.

In the asymmetric and strongly connected scenario (Fig. 6b), the additional cost generated by the use of meta-models, regardless of the meta-features and learning techniques, was lower than the baseline model. The model induced from the decision tree induction algorithm with CNM meta-features presented the best performance (average rate = 0.44% and standard deviation = 2.58%). This may have occurred because most of the features extracted from the complex networks are related to connectivity between network nodes, not taking into account the weighting of the edges.

Although the baseline cost rate was lower in the symmetric and weakly connected scenario (Fig. 7a), the meta-models were able to maintain a good performance in predicting the MH ranking. In this scenario, the use of the CNM meta-features by the PCT algorithm obtained again the best results in terms of extra costs (average rate = 0.53 and standard deviation = 4.48%). Finally, Fig. 7b shows that the performance obtained by meta-models in the asymmetric and weakly connected scenario — average rate = 4.08% and standard deviation = 15.12% — was not as good as those observed in the other scenarios. There were large variations in the quality of the predicted rankings, resulting in high standard deviation values. The learning process may have been harmed by the lack of information about some of the connections between the vertices. If a user follows the recommended rankings to solve a TSP instance

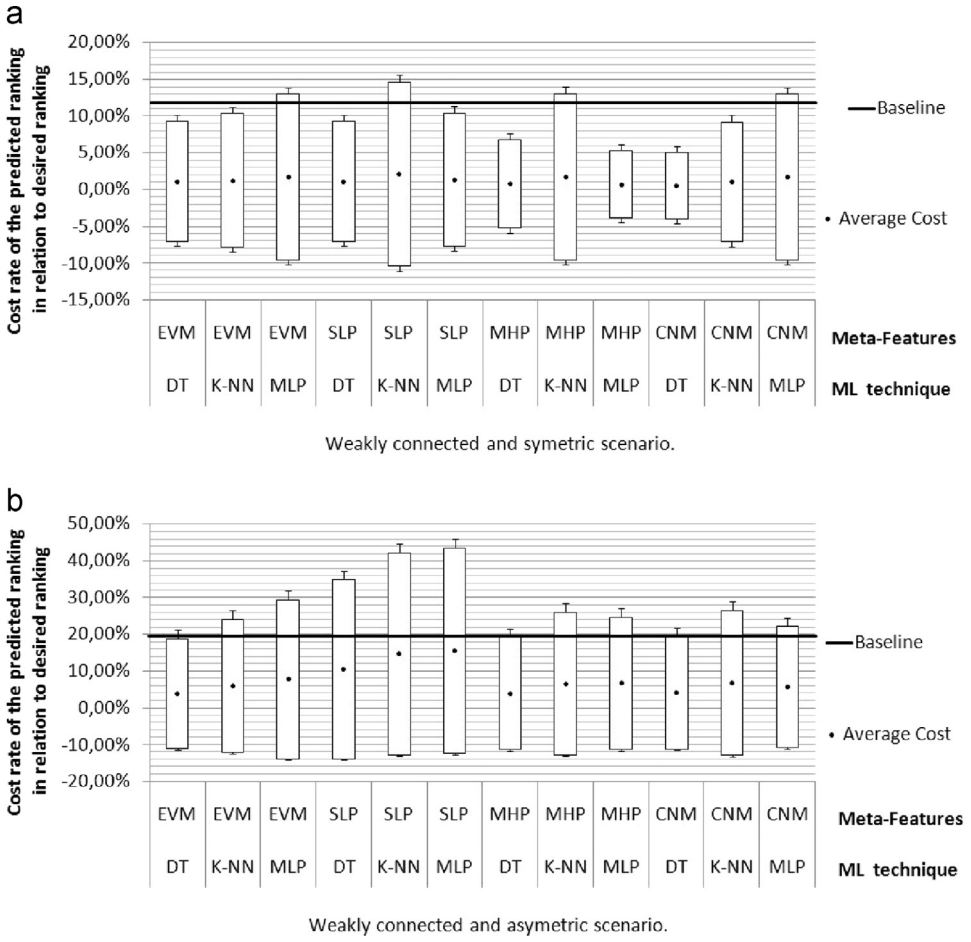

Fig. 7. Cost rate when using the predicted ranking of MHs regarding the ideal ranking for a weakly connected TSP instance.

with the characteristics described in this scenario, the estimated cost of the solution will be at least 4% higher than the ideal cost.

9. Conclusion

We addressed the problem of recommending the ranking of the best MHs for a given instance of the TSP. To do so, a meta-learning approach was used. In particular, we used machine learning techniques to induce a model able to associate properties of the TSP instances with the optimization performance of MHs. Adaptations of MLP, K-NN, and Decision Tree were employed for label ranking. Four sets of meta-features were evaluated: edge and vertex measures (*EMV*), meta-heuristics properties (*MHP*), subsampling landmarkers properties (*SLP*), and complex network measures (*CNM*). Additionally, four different scenarios of the symmetry and level of connectivity of the graphs representing the TSP instances were considered.

Experimental results show that it is possible to predict the ranking of MHs and that, by following the recommendations from the rankings, it is possible to obtain solutions better than simpler selection strategies, like recommendation based on average ranking. Even in the more complex scenarios, like those that have asymmetric costs of travel and weak connectivity, the meta-learning approaches showed good predictive performance, independent of the set of meta-features.

We also analyzed the average time necessary to generate each set of meta-features and the gains compared with not using meta-learning. Some meta-features have a computational complexity that is quadratic on the number of cities and result in processing time longer than the time necessary to run all MHs. As expected, this is especially true in the larger TSP instances. Thus, only two sets of meta-features obtained effectively satisfactory results: the set based on measures from edges and vertex and the set based on properties of subsampling landmarkers.

To address the computational issues of two of the approaches to compute the meta-features, we plan to combine them with the subsampling landmarkers approach. This means computing them on a subgraph of the graph representing the instance. Additionally, more extensive experimentation is required to confirm whether the different approaches to compute the meta-features are equivalent in terms of predictive accuracy. If this observation is confirmed, it is important to understand the reasons that make it hold. In addition, we are going to use a feature selection technique to improve the ranking accuracy. It is also important to evaluate the performance of the meta-learning system in terms of the base level TSP solution found by the MHs selected by the meta-models.

Additional future work includes using a meta-learning approach for label ranking (e.g. decision tree) to induce meta-models from the meta-data of each scenario and evaluate their predictive performance in other scenarios. Decision trees generate more intelligible meta-models, since they show the rules used by the meta-model to produce the result on their output. Finally, we want to improve the composition of the ranking of MHs taking into account the probability of each MH to find a feasible solution.

Acknowledgments

The authors acknowledge CAPES, CNPq, FAPESP, and FAPEAM for their financial support. This work was also partially supported by FCT project “Evolutionary algorithms for Decision Problems in Management Science” (PTDC/EGE-GES/099741/2008); by projects (”NORTE-07-0124-FEDER-000057”) and SML (”NORTE-07-0124-FEDER-000059”), which are financed by the North Portugal Regional Operational Programme (ON.2 O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF), and by national funds, through the Portuguese funding agency, Fundação para a Ciência e a Tecnologia (FCT); and by the ERDF European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT Fundacao para a Ciencia e a Tecnologia (Portuguese Foundation for Science and Technology) within project FCOMP - 01-0124-FEDER-022701.

References

[1] G. Gutin, A. Punnen, The Traveling Salesman Problem and Its Variations, Kluwer Academic Publishers, Netherlands, 2002.
[2] D. Applegate, R. Bixby, W. Cook, The Traveling Salesman Problem: A Computational Study, Princeton University Press, New Jersey, 2006.
[3] C.H. Papadimitriou, The euclidean traveling salesman problem is np-complete, Theor. Comput. Sci. 4 (3) (1977) 237–244.
[4] M. Gendreau, J.-Y. Potvin, Handbook of Metaheuristics, 2nd Edition, Springer Publishing Company, Incorporated, New York; 2010.
[5] F. Glover, E. Tillard, D. Werra, A user's guide to tabu search. in p.l, Ann. Oper. Res. 41 (1993) 3–28.
[6] T. Feo, M. Resende, Greedy randomized adaptive search procedures, J. Glob. Optim. 6 (1995) 109–133.
[7] S. Kirkpatrick, C. Gelatt, M. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[8] J. Holland, Genetic algorithms and the optimal allocations of trial, SIAM J. Comput. 2 (1973) 88–105.
[9] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput. 1 (1), 1997, 53-66.
[10] D. Wolpert, W. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1997) 67–82.
[11] E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, S. Schulenburg, Hyper-heuristics: an emerging direction in modern search technology, in: F. Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics, International Series in Operations Research and Management Science, vol. 57, Springer, New York, 2003, pp. 457–474.
[12] P. Brazdil, C. Giraud-Carrier, C. Soares, R. Vilalta, Metalearning: Applications to Data Mining, Springer, Berlin, 2009.
[13] J. Kanda, A. Carvalho, E. Hruschka, C. Soares, Selection of algorithms to solve traveling salesman problems using meta-learning, Int. J. Hybrid Intell. Syst. 8 (3) (2011) 117–128.
[14] O. Dekel, Y. Singer, C.D. Manning, Log-linear models for label ranking, in: S. Thrun, L. Saul, B. Schölkopf (Eds.), Advances in Neural Information Processing Systems, vol. 16, MIT Press, Cambridge, Massachusetts, London, England; 2004, pp. 497–504.
[15] K. Smith-Miles, J. van Hemert, X. Lim, Understanding tsp difficulty by learning from evolved instances, in: Proceedings of the 4th International Conference on Learning and Intelligent Optimization, LION10, Vol. 6073, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 266–280.
[16] J. Kanda, C. Soares, E. Hruschka, A. de Carvalho, A meta-learning approach to select meta-heuristics for the traveling salesman problem using mlp-based label ranking, in: Proceedings of the 19th International Conference on Neural Information Processing—Volume Part III, ICONIP'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 488–495.
[17] P. Brazdil, C. Soares, J. Costa, Ranking learning algorithms: using *IBL* and meta-learning on accuracy and time results, Mach. Learn. 50 (2003) 251–257.
[18] 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, vol. 978, 2008, pp. 4118–4124.
[19] L. Xu, F. Hutter, H. Hoos, K. Leyton-Brown, SATzilla: portfolio-based algorithm selection for SAT, J. Artif. Intell. Res. 32 (2008) 565–606.
[20] J. Fürnkranz, E. Hüllermeier, E. Mencía, K. Brinker, Multilabel classification via calibrated label ranking, Mach. Learn. 73 (2008) 133–153.
[21] S. Vembu, T. Gärtner, Label ranking algorithms: a survey, in: J. Fürnkranz, E. Hüllermeier (Eds.), Preference Learning, Springer, Berlin, Heidelberg, 2011, pp. 45–64.
[22] C. Spearman, The proof and measurement of association between two things, Am. J. Psychol. 15 (1904) 72–101.
[23] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Pearson Education, Inc., Boston, 2006.
[24] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[25] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart, J.L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1, MIT Press, Cambridge, MA, USA, 1986, pp. 318–362.

[26] J. Kanda, A. de Carvalho, E. Hruschka, C. Soares, Using meta-learning to recommend meta-heuristics for the traveling salesman problem, in: 2011 10th International Conference on Machine Learning and Applications and Workshops (ICMLA), vol. 1, 2011, pp. 346–351.
[27] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1) (1967) 21–27.
[28] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, H. Blockeel, Decision trees for hierarchical multi-label classification, Mach. Learn. 73 (2) (2008) 185–214.
[29] K. Smith-Miles, L. Lopes, Review: measuring instance difficulty for combinatorial optimization problems, Comput. Oper. Res. 39 (5) (2012) 875–889.
[30] R. Leite, P. Brazdil, Predicting relative performance of classifiers from samples, in: Proceedings of the 22nd International Conference on Machine learning, ICML '05, ACM, New York, NY, USA, 2005, pp. 497–503.
[31] L. Costa, F. Rodrigues, G. Travieso, P.V. Boas, Characterization of complex networks: a survey of measurements, Adv. Phys. 56 (2007) 167–242.
[32] D.E. Goldberg, R. Lingle, Alleles, loci, and the traveling salesman problem, in: J. J. Grefenstette (Ed.), Proceedings of the First International Conference on Genetic Algorithms and Their Applications, Lawrence Erlbaum Associates, Publishers, 1985, pp. 154–159.
[33] R. Leite, P. Brazdil, Active testing strategy to predict the best classification algorithm via sampling and metalearning, in: Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence, IOS Press, Amsterdam, The Netherlands, The Netherlands, 2010, pp. 309–314. http://dl.acm.org/citation.cfm?id=1860967.1861029.
[34] Y.-J. Park, G.-B. Lee, Application of heuristic approaches to minimization of energy consumption in inner layer scrubbing process in PCB manufacturing, Int. J. Precis. Eng. Manuf. 13 (7) (2012) 1059–1066, http://dx.doi.org/10.1007/s12541-012-0138-8, URL ⟨http://link.springer.com/10.1007/s12541-012-0138-8⟩.
[35] G. Reinelt, TSPLIB—a traveling salesman problem library, ORSA J. Comput. 3 (1991) 376–384.
[36] D. Watts, Small Worlds: the Dynamics of Networks Between Order and Randomness, Princeton University Press, Princeton, 1999.
[37] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. Witten, The weka data mining software: an update, SIGKDD Explor. Newsl. 11 (1) (2009) 10–18.

Jorge Kanda received a M.Sc. degree in Informatics from Federal University of Amazonas (UFAM), Brazil, in 2004, and a Ph.D. in Science from University of São Paulo (USP) at São Carlos, Brazil, in 2012. He is a Professor at the Institute of Exact Sciences and Technology (ICET) of the UFAM at Itacoatiara since 2007. Kanda is collaborator of the Post Graduation Program in Science and Technology for Amazonian Resources, UFAM. His research interests are Machine Learning, Meta-learning, and Data Mining, among others.

André de Carvalho is a Full Professor in the department of Computer Science, University of Sao Paulo, Brazil. His main research interests are data mining, data science and machine learning. Prof. de Carvalho supervised more than 20 Ph.D. students. He is currently a member of the International Association for Statistical Computing (IASC) Council and director of the Center of Machine Learning in Data Analysis of the University of S ao Paulo.

Eduardo Hruschka is a Chief Data Scientist at Big Data Brazil and also a Professor (currently on unpaid sabbatical leave) at the Computer Science Department of the University of Sao Paulo (USP) at São Carlos. He received his Ph.D. from COPPE/Federal University of Rio de Janeiro, Brazil, in 2001. In 2006, he was awarded with a "Young Investigator Award", which is given as a Research Grant by FAPESP (São Paulo Research Foundation). He is also an Advanced Research Fellow of the Brazilian National Research Council (CNPq). From 2010 to 2012, he worked as a Visiting Researcher at the University of Texas at Austin, USA.

Carlos Soares received a B.Sc. degree in Systems Engineering and Informatics from Universidade do Minho, Portugal, a M.Sc. degree in Artificial Intelligence and a Ph.D. in Computer Science from Universidade do Porto, Portugal. After 15 years at the Faculty of Economics, he is now an Associate Professor at the Faculty of Engineering of Universidade do Porto. Carlos is a Researcher at INESC TEC, with interests on Data Mining, Business Intelligence and Evolutionary Computation. He was awarded the Scientific Merit and Excellence Award of the Portuguese AI Association.

Pavel Brazdil is a Full Professor at FEP, University of Porto. Most of his research is carried at LIAAD INESC Tec. His research interests lie in the areas of Data Mining, Machine Learning, Meta-learning and Text Mining, among others. He has supervised 12 Ph.D. students. He published or edited 5 books and more than 120 papers referenced on Google Scholar, of which about 80 are also on ISI/DBLP/Scopus. He is a Member of the editorial board of the Machine Learning Journal and is a Fellow of ECCAI.