International Journal of Hybrid Intelligent Systems 8 (2011) 1–13
DOI 10.3233/HIS-2011-0133
IOS Press

Selection of algorithms to solve traveling salesman problems using meta-learning

Jorge Kanda[a,b,*] Andre Carvalho[a,c] Eduardo Hruschka[a] and Carlos Soares[d]
[a]*Instituto de Ciencias Matematicas e de Computacao, Universidade de Sao Paulo, Sao Carlos, Brazil*
[b]*Instituto de Ciencias Exatas e Tecnologias, Universidade Federal do Amazonas, Itacoatiara, Brazil*
[c]*School of Computing, University of Kent, Canterbury, CT2 7NF, UK*
[d]*LIAAD-INESC Porto LA/Faculdade de Economia, Universidade do Porto, Porto, Portugal*

Abstract. Many real-world problems, like microchip design, can be modeled by means of the well-known traveling salesman problem (TSP). Many instances of this problem can be found in the literature. Although several optimization algorithms have been applied to TSP instances, the selection of the more promising algorithm is, in practice, a difficult decision. In this paper, a new meta-learning-based approach is investigated for the selection of optimization algorithms for TSP instances. Essentially, a learning model is trained with TSP instances for which the performance of a set of optimization algorithms is known a priori. Then, the learned model is used to predict the best algorithm for a new TSP instance. Each instance is described by meta-features that capture characteristics of the TSP that affect the performance of the optimization algorithms. Given that the best solution for a given TSP instance can be obtained by several algorithms, the meta-learning problem is considered here to be a multi-label classification problem. Several experiments illustrate the performance of the proposed approach, with promising results.

Keywords: Meta-learning, multi-label classification, traveling salesman problem, meta-heuristics, algorithm selection

1. Introduction

In optimization problems, the goal is typically to find values for a set of parameters that maximizes (or minimizes) the output value of an objective function, without violating a set of restrictions. This rather abstract setting underlies many real-world applications found in Management, Economy and Engineering, such as vehicle routing, resource allocation, strategic planning, and scheduling, just to name a few. Under this context, combinatorial optimization problems – for which the search space of the possible solutions is discrete [2] – are ubiquitous. A classical example of a combinatorial optimization problem is the well-known traveling salesman problem (TSP), which has several instances. A TSP instance can be informally posed as: given a set of cities and their respective pair-wise distances, find the shortest tour that visits each city exactly once. TSP can be instantiated in many different ways, e.g., depending on the number of cities and the distribution of connections among them, travel times, prize collecting and (a)symmetry.

The TSP has been extensively studied [15] and, more formally, has been proved to be in the class of NP-hard problems [12]. Therefore, although there are exact algorithms that are reasonably efficient for small problems, approximation and heuristic (suboptimal) algorithms are frequently employed in larger practical applications. It is also well-known that, according to the "no free lunch" theorems, if any algorithm does particularly well on average for one class of problems, then

*Corresponding author: Instituto de Ciencias Matematicas e de Computacao, Universidade de Sao Paulo, Av. Trabalhador sao-carlense, 400, Centro, 13560-970, Sao Carlos, Sao Paulo, Brazil. Tel.: +55 16 3373 8161; Fax: +55 16 3373 9650; E-mail: {kanda, andre, erh}@icmc.usp.br, csoares@fep.up.pt.

[1]This is the extended version of the paper: Kanda et al. "Using Meta-learning to Classify Traveling Salesman Problems," in Proc. of the 11th Brazilian Symposium on Neural Networks (SBRN 2010), pp. 73–78, IEEE Press, 2010.

it must do worse, on average, over the remaining problems [28]. In other words, the (lack of) success of a given algorithm is dependent on the TSP instance. Given that each algorithm has its own bias, resulting in good performance on a subset of the universe of possible problems and bad performance in the remaining problems, an important question to be answered in practice is: What is the best algorithm for a particular TSP instance? In order to answer this question, one could be tempted to consider running all the available algorithms for a particular application (e.g., vehicle routing) and then get the one that provides the best performance on this particular instance. However, this approach is frequently not feasible, except for very small TSP instances. For example, suppose that a manager needs to organize the delivery of several orders in the following week with minimal transportation costs and taking into account some constraints (e.g., the delivery data for each order). In this scenario, and assuming reasonably large TSP instances, running all the available heuristics is likely to be computationally prohibitive. Thus, better approaches are needed. This work addresses the practical difficulty that involves choosing, from a number of available algorithms, the one that will likely provide the best performance on a given instance of an optimization problem.

A potentially promising approach for selecting the best algorithm for a given TSP instance involves using concepts of meta-learning, which has been successfully used for tackling (the broadly defined) algorithm selection problems (e.g., see [6,22]). Meta-learning consists of generating a meta-model that maps characteristics of problems to the performance of algorithms when applied to these problems [5]. The present paper proposes a meta-learning approach to recommend optimization algorithms for new TSP instances. The algorithms considered here are a set of (meta-)heuristics, which have been commonly used for the TSP: Tabu Search (TS) [14], Greedy Randomized Adaptive Search Procedure (GRASP) [11], Simulated Annealing (SA) [19, 9] and Genetic Algorithms (GA) [16]. However, as it hopefully will become evident latter in the paper, the main ideas underlying this work are not limited to the use of such meta-heuristics, which serve mostly for illustrative purposes.

The main contributions of this paper are: (i) the description and evaluation of a meta-learning-based approach for the algorithm selection problem, particularly for the TSP. We do so by training machine learning classification algorithms to predict the most promising optimization technique(s) for TSP instances; (ii) motivated by the fact that multiple algorithms can provide the best solution for a given TSP instance, we cast the algorithm selection problem as a multi-label classification task (as opposed to traditional classification, where each instance is assigned to a single class); (iii) the proposal of measures for the characterization of TSP instances based on their graph representation. This paper is an extended version of previous work [18]. Here, we provide a more detailed description of the proposed approach and offer further details on related work. Besides, we here present a more in-depth analysis of our experimental results, shedding light on some important aspects not discussed in [18].

The remainder of this paper is organized as follows. Section 2 provides a brief description of the TSP that allows us to introduce the adopted notation. It also gives an overview of methods usually applied to solve TSP instances. Sections 3 and 4 briefly describe the meta-learning approach to algorithm selection and fundamental concepts related to multi-label classification, respectively. Section 5 presents the proposed meta-learning approach to select algorithms for TSP instances. Section 6 outlines our experimental design and Section 7 reports the results obtained. Finally, the conclusions are presented in Section 8.

2. The Traveling Salesman Problem (TSP)

Informally, the TSP can be defined as follows. Given a set of cities and the traveling cost between every pair of them, the goal is to find the cheapest way of visiting all of the cities and returning to the starting point. More formally, a TSP instance with n cities can be defined as a graph $G = (V, E)$, where $V = \{1, 2 \ldots, n\}$ is a set of nodes, $E = \{\langle i, j\rangle\colon i,j \in V\}$ is a set of edges. Each city corresponds to a specific $v \in V$ and each edge $\langle i, j\rangle$ connects nodes i and j. The cost of traveling from node i to node j, using the corresponding edge, can be represented by c_{ij}. A TSP is symmetric if $c_{ij} = c_{ji}$ for all edges $\langle i, j\rangle$; otherwise it is asymmetric. A TSP is complete if $c_{ij} > 0$ for all edges $\langle i, j\rangle$. Under this setting, the best solution for a TSP instance is the set of paths that visit each city only once with minimum total travel cost [1].

As discussed in Section 1, approximation algorithms and heuristic (suboptimal) algorithms are frequently used in practical applications of the TSP. These algorithms can find solutions of acceptable quality in reasonable time. From this perspective, some papers have addressed performance comparisons of heuristic algo-

rithms for the TSP e.g., see [24,17]. More recently, Oncam et al. [20] performed an experimental comparison of 16 algorithms in 10 asymmetric TSP instances.

Meta-heuristics have also been used for the TSP. In brief, a meta-heuristic can be viewed as top-level general strategy that guides other heuristics. As such, meta-heuristics usually make only a few assumptions about the optimization problem being tackled and allow searching over large spaces of candidate solutions. For such, stochastic optimization based procedures, which usually do not offer optimality guarantees, are frequently employed. In this paper, four meta-heuristics – Tabu Search (TS) [14], Greedy Randomized Adaptive Search Procedure (GRASP) [11], Simulated Annealing (SA) [19,9] and Genetic Algorithms (GA) [16] – are used to illustrate our proposed approach for algorithm selection based on meta-learning.

3. Algorithm Selection with Meta-learning

The problem of selecting the best algorithm for a given problem has been successfully dealt with in the last years by a sub-area of Machine Learning (ML) known as meta-learning [5].

In short, meta-learning consists of generating a meta-model that maps characteristics of problems (i.e., *meta-features*) to the performance of algorithms when they are applied to these problems. To illustrate how it works, let us consider the problem of selecting the best algorithm for classification problems. In this case, the problems are classification problems represented by classification datasets and the algorithms are classification algorithms. The meta-features could include a combination of simple characteristics (e.g., number of classes in classification datasets) or of more complex features (e.g., mean mutual information between nominal attributes and class) [5]. In our work, the problems are TSP instances and the algorithms are meta-heuristics. The use of meta-learning for TSP problems will be explained later (Section 5).

When meta-learning is used for algorithm selection, the values of the meta-features are derived from a set of problems and the performance of the algorithms on these problems is estimated by running the algorithms. The data that are gathered in this way are the *meta-data*. A suitable ML algorithm can then be applied to these meta-data to obtain a meta-model. This meta-model can be used to predict the relative performance of the investigated algorithm on future problems.

Figure 1 shows a schematic diagram for meta-model construction in the algorithm selection problem using meta-learning. The construction of the meta-model is carried out in two phases. The first is the knowledge acquisition, in which problem instances are obtained from the problem space. Afterwards, problems characteristics are identified and the meta-features are extracted. The candidate algorithms are applied to the selected problem instances and their performance is estimated. The resulting values are used to generate the meta-target variable. The meta-data consists of both the meta-features and meta-target. The second phase is to create the meta-model. This is achieved by applying ML techniques on the meta-data gathered in the first phase. The meta-model obtained can later be used to predict the relative performance of the candidate algorithms on a new instance and to recommend the most promising algorithm. Like in any other ML task, the meta-models should be empirically evaluated. If meta-learning is considered to be a classification task, the accuracy rate can be used to evaluate its performance; if seen as a regression task, its performance can be evaluated by a similarity measure, like the correlation coefficients of Goodman-Kruskal [13] and Weighted Goodman-Kruskal [7].

Although most studies concerned with meta-learning focuses on the recommendation of classification algorithms, meta-learning can be used to recommend algorithms for any problem, including algorithms for optimization problems, as discussed in [22], in which aspects of meta-learning were applied to a set of 28 instances of the Quadratic Assignment Problem (QAP) to estimate the percentage deviation of three meta-heuristic algorithms compared to the best known solutions averaged over 10 random starts of the algorithms. The instances were characterized by fitness-distance metrics. In [23], instances of TSP are evolved using an evolutionary algorithm to produce distinct classes of instances, which are intentionally made easy or hard for certain algorithms. Meta-data contains these instances characterized by a set of 12 features whose impact on the difficulty degree associated with each algorithm is investigated.

The target variable of the meta-data is based on the performance of the meta-heuristics. To define its value, we apply the meta-heuristics to the TSP instances and measure the best solution found after a stopping criterion is met. If the problem is addressed as a classification task, the class of each instance is the meta-heuristic with the best solution. However, in many cases, the best solution is obtained by more than one meta-

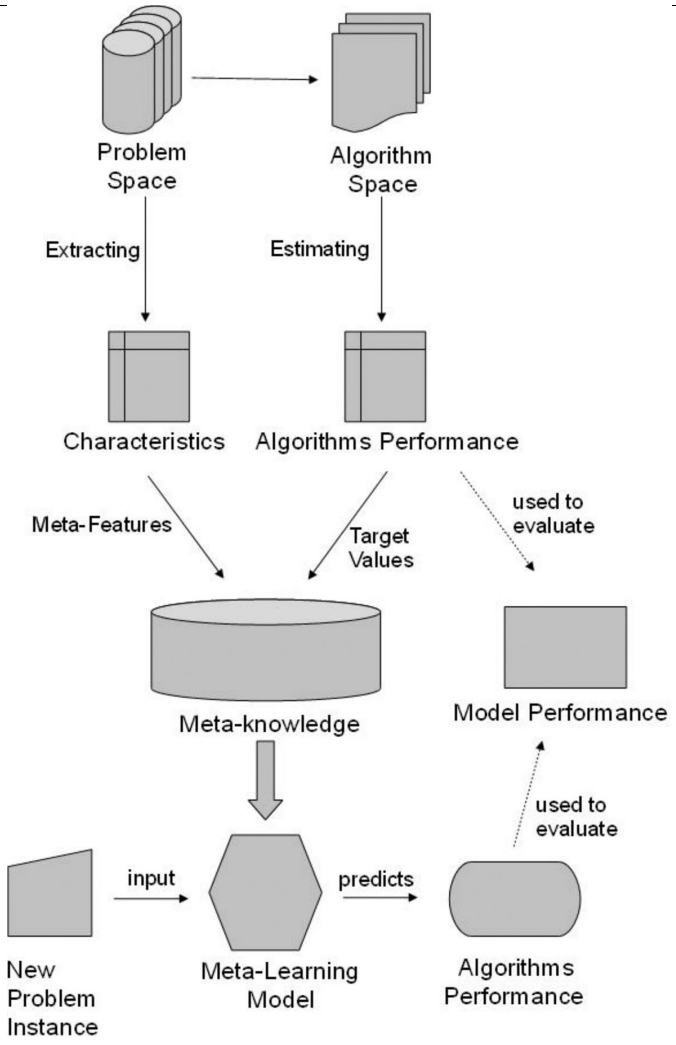


Fig. 1. Construction of the meta-learning model for the algorithm selection problem.

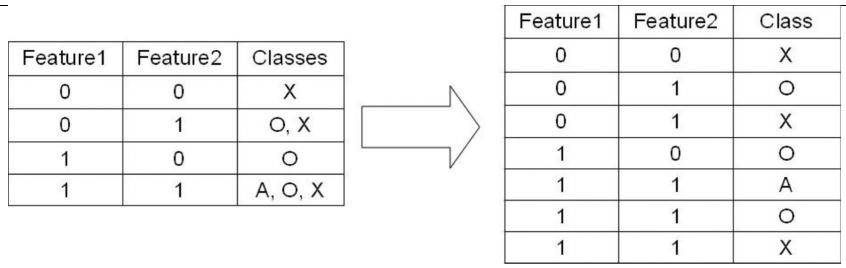
heuristic. In these cases, rather than choosing one of the best methods as the class, we address the problem as a multi-label classification problem. In practice, this means that each example may be labeled with several classes and methods that are able to deal with multi-label classification problems are used, as described in the following section.

4. Multi-label classification

In ML, most of the classification problems are single-label classification problems, where one label or class is assigned to each example [8].

However, there are classification problems where each instance can simultaneously have more than one class. These problems are known as multi-label classification problems. Different methods for multi-label classification problems have been proposed in the literature. Basically, these methods transform a dataset with multi-label instances into a dataset with only single-label instances. The following transformation methods are investigated in this work.

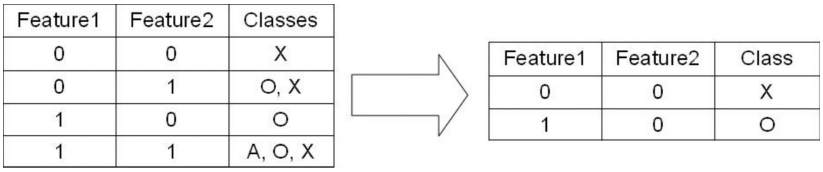
– Method 1: Decomposition of multi-label instances into several single-label instances [26]. Figure 2 shows an example where each multi-label instance is decomposed into "*r*" instances, where "*r*" corresponds to the number of classes associated with the instance. A disadvantage of this method is that it will generate inconsistent instances, with the same meta-feature values but different target attribute values.



Feature1	Feature2	Classes
0	0	X
0	1	O, X
1	0	O
1	1	A, O, X

Feature1	Feature2	Class
0	0	X
0	1	O
0	1	X
1	0	O
1	1	A
1	1	O
1	1	X

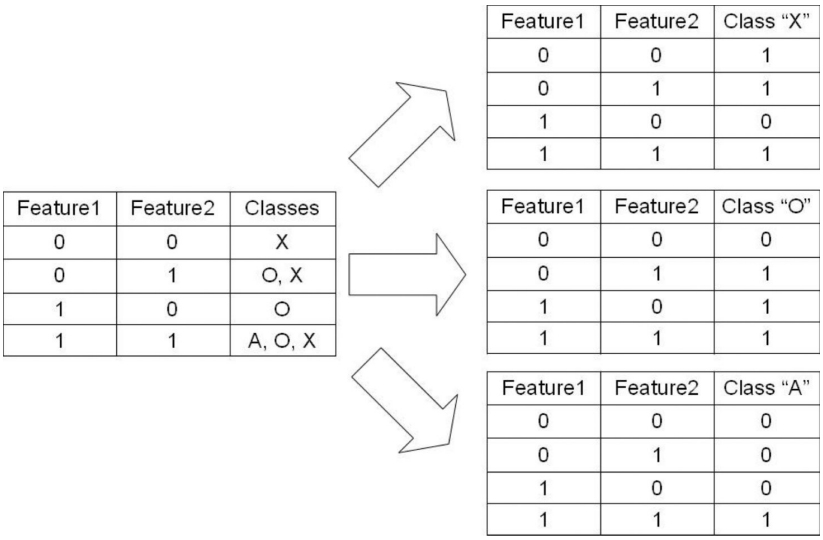
Fig. 2. Method 1: Decomposition of multi-label instances into single label instances.



Feature1	Feature2	Classes
0	0	X
0	1	O, X
1	0	O
1	1	A, O, X

Feature1	Feature2	Class
0	0	X
1	0	O

Fig. 3. Method 2: Elimination of multi-label instances.



Feature1	Feature2	Classes
0	0	X
0	1	O, X
1	0	O
1	1	A, O, X

Feature1	Feature2	Class "X"
0	0	1
0	1	1
1	0	0
1	1	1

Feature1	Feature2	Class "O"
0	0	0
0	1	1
1	0	1
1	1	1

Feature1	Feature2	Class "A"
0	0	0
0	1	0
1	0	0
1	1	1

Fig. 4. Method 3: Transformation into several binary classification problems.

– Method 2: Elimination of multi-label instances [3]. In this method, exhibited in Fig. 3, all multi-label instances are eliminated from the dataset. One problem with this method is that the instances representing relevant information to characterize the problem domain can be excluded and, as a result, jeopardize the induction of the correct model.
– Method 3: Binary representation [8]. New classification problems are created, one for each label, using the label as positive class and all the other labels as negative class. Figure 4 illustrates this method. The predicted labels are defined by combining the results obtained by the classifiers.

5. Meta-learning for TSP meta-heuristic selection

The application of a meta-learning approach to the problem of selecting meta-heuristics for TSP is essentially straightforward. Meta-data represent problem instances and the performance of the meta-heuristics on these instances. The goal is to obtain a model that can predict the relative performance of these meta-heuristics for new problems.

For such, it is necessary to define meta-features that are adequate for TSP. The meta-features should represent properties of the instances that affect the relative performance of the meta-heuristics [5]. The perfor-

Table 1
TSP Meta-features used to train a meta-learning model

Meta-feature	Description
V	Number of vertices
E	Number of edges
E_{low}	Lowest edge cost
E_{hig}	Highest edge cost
E_{avg}	Average edge cost
E_{std}	Standard deviation of the edge cost
E_{qmode}	Mode quantity of the edge cost
E_{fmode}	Frequency of the mode of the edge cost
E_{mode}	Average of the costs that occur the most frequently in the edges
E_{median}	Median of the edge costs
E_{lavg}	Quantity of edges whose costs are lower than the average of the edge costs
V_{lower}	sum of the V edges of lowest values

mance of different heuristics is affected by the structure of the graph that represents an instance (e.g., edges and corresponding costs). Thereby, the meta-features can be based on information measures extracted from the graph that represents each TSP example. In this work, we used the meta-features shown in Table 1.

A few simple measures were used. The first two meta-features, V and E, represent the number of cities (i.e., the order of the graph) and the number of edges between pairs of cities (i.e., the size of the graph), respectively. A graph is strongly connected if there are edges connecting all pairs of vertices. In our work, we analyzed only strongly connected graphs. Each edge is associated with a cost. The minimum and maximum costs are represented by E_{low} and E_{hig}, respectively.

The next set of measures characterize the TSP instances in terms of the costs of the edges: E_{avg} is the mean edge cost and E_{std} is the corresponding standard deviation. In statistics, the mode is the value that occurs the most frequently in a variable. When there are two modes, the dataset is said bimodal, while a variable with more than two modes is known as multimodal. E_{qmode} is the meta-feature that gives the number of modes in the costs and E_{fmode} represents the frequency of the most common cost value(s). E_{mode} is the mean of all modal values. E_{lavg} represents the frequency of the values which are lower than the mean cost. E_{median} is the median cost of the edges. Finally, V_{lower} represents the sum of the V edges with the lowest values. The solution for a TSP instance has a cost that cannot be less than V_{lower}. All TSP graphs in our study are both symmetric and complete, thus these characteristics were not selected for meta-features.

6. Experimental Setup

Several benchmark instances of TSP can be found in the TSPLIB library [21]. However, most of these TSP instances have a large number of cities, what raises an important problem for our approach. For each instance considered, we must run all meta-heuristics to evaluate the solution they obtain. Moreover, to train a meta-learning model, we must build a dataset with many instances. Therefore, the experiments to collect the data for meta-learning are very computationally demanding. As our goal is to show that it is possible to use a meta-learning approach to choose a meta-heuristic for the TSP, we decided to work with smaller instances, either generated synthetically or obtained by selecting sub-problems of real TSP instances.

We generated a synthetic dataset with 535 TSP instances (with at most 50 cities) which can be categorized into twelve groups, according to the values of their meta-features. The number of instances in each group was sampled from a uniform probability distribution in the domain represented by the interval [30, 70].

The remaining instances are sub-problems generated from five symmetric TSP ($d1291$, $fl1400$, $nrw1379$, $pcb1173$ and $pr1002$) obtained from the TSPLIB Library. The number of cities represented can be inferred from their name, e.g., $d1291$ is a TSP with 1291 cities. To reduce the computational cost of obtaining the corresponding meta-data, we have sampled a number of cities from each file by arbitrarily establishing an upper bound for the number of cities, namely 100 cities. Since in the synthetic TSP problems the maximum number of cities is 50, in this case we created TSP instances whose number of cities vary from 60 to 100. More precisely, we have randomly chosen 100 sub-problems (subsets of instances) for every number of cities (60, 70, 80, 90 and 100) from every instance. This resulted in 2500 instances of TSP (5 original instances $\times$ 5 different number of cities $\times$ 100 random subsets of instances).

As previously mentioned, four meta-heuristics have been investigated in our work, namely: TS, GRASP, SA

Table 2
Performance of the meta-heuristics in TSP instances generated from synthetic data

Meta-heuristic	Performance (%)
TS	39.25
GRASP	33.08
SA	31.59
GA	20.37

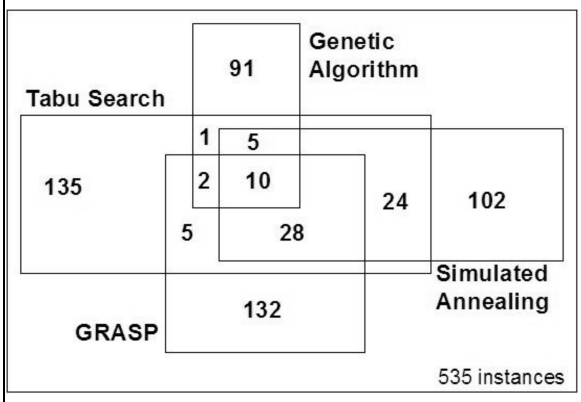


Fig. 5. Quantities of best solutions for each meta-heuristic in TSP instances generated from synthetic data.

Table 3
Performance of the meta-heuristics – real data

Meta-heuristic	Value (%)
TS	15.48
GRASP	59.00
SA	26.40
GA	26.56

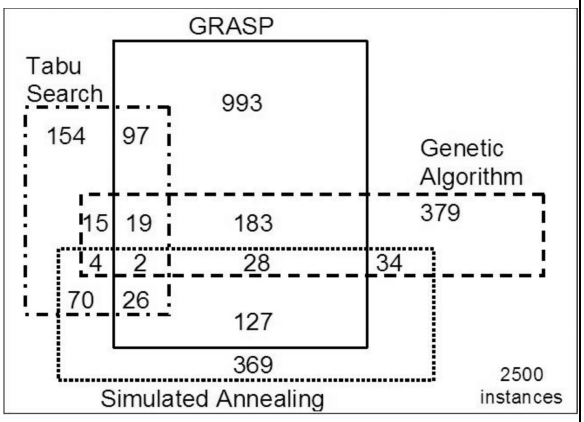


Fig. 6. Quantities of best solutions for each meta-heuristic in TSP instances generated from real data.

and GA.[1] These are stochastic meta-heuristics, whose results may vary for one run to another. Thus, for each TSP instance, every meta-heuristic was run 50 times (with different initial seeds), thus producing 50 solutions for each of them. The performance of each meta-heuristic was then estimated by the average of such 50 runs. It is important to observe that the same processing time was adopted as a stopping criterion for each of those 50 runs. By doing so, the performance of the studied meta-heuristics is assessed under the same *computational limitation*, thus providing a more fair comparison among them. Additionally, this scenario represents a better simulation of a practical application where computational resources and/or time are limited. The meta-heuristics adopted have a set of free parameters whose values influence their performances. In the experiments carried out, the following parameter settings were used for the meta-heuristics:

- TS: size of the tabu list $= 2$; number of iterations with no improvement of the current solution $= 2$;
- GRASP: number of iterations $= 50$; $\alpha = 0.1$;
- SA: $\beta = 0.95$; $t_0 = 1$; $\alpha = 0.01$;
- GA: edge recombination crossover (ERX) [27] as the crossover operator; maximum number of individuals in the population $= 128$; tournament selection; mutation rate $= 5\%$; elitism selection.

[1] These meta-heuristics were implemented in C Language.

Table 2 shows the relative performance of each meta-heuristic in the artificial TSP instances. The sum of the values is higher than 100% because, for some instances, more than one meta-heuristic presented the best performance. This table shows that TS presented the best overall performance, i.e., it has shown the best performance in 39.25% of the TSP instances. This percentage – sometimes referred to as *default predictive accuracy* – can be considered as a baseline measure for the accuracy of the assessed classifiers. Figure 5 illustrates the number of times each meta-heuristic produced the best average solution for the 535 TSP instances. The overlapping areas represent the instances whose best solution was found by more than one meta-heuristic.

The relative performance of each meta-heuristic in the 2500 TSP instances is shown in Table 3. As more than one meta-heuristic presented the best performance for some instances the sum of the values is higher than 100%. Table 3 shows that GRASP presented the best overall performance. In particular, it has shown to be the best algorithm in 59% of the instances. This proportion is our baseline measure for the accuracy of the used classifiers. Figure 6 illustrates the number of times each meta-heuristic produced the best average

Table 4
Accuracy results – synthetic data

Method	K-NN	DT	SVM	NB
1	0.52±0.10	**0.54**±0.09	0.53±0.10	0.53±0.11
2	**0.63**±0.13	**0.63**±0.13	**0.63**±0.08	0.62±0.12
3	0.82±0.04	**0.90**±0.04	0.82±0.02	0.76±0.06

Table 5
AUC results – synthetic data

Method	K-NN	DT	SVM	NB
1	**0.79**±0.08	0.77±0.07	0.75±0.07	0.74±0.08
2	**0.88**±0.06	0.81±0.07	0.84±0.07	0.84±0.09
3	0.91±0.02	**0.95**±0.03	0.79±0.01	0.82±0.04

solution (considering all the 2500 TSP instances). The overlapping areas represent the instances whose best solution was found by more than one meta-heuristic, thus naturally leading to the multi-label representation adopted in this paper.

The input attributes of the data used for meta-learning are the meta-features described earlier (Section 5). The output (target) attribute represents the meta-heuristic with the best average performance for the TSP instance in consideration (class label). When more than one meta-heuristic presents the best performance, the output attribute has more than one label, thus producing a multi-label dataset. Four classification algorithms – K-Nearest-Neighbors (K-NN), Decision Tree (DT), Support Vector Machine (SVM), and Naïve Bayes (NB)[2] – have been used in our meta-learning experiments. As addressed in Section 4, we study three methods for dealing with the multi-label classification problem that arises in our algorithm selection setting. In brief, Method 1 involves replicating every *c*-labeled instance into *c* single labeled instances. Method 2 just excludes multi-labeled instances from the dataset. Finally, Method 3 decomposes the multi-label classification problem into *c* binary classification problems. For all these methods, the standard 10-fold cross-validation process was used to estimate the generalization capability of the employed classifiers. Two widely known measures have been adopted for performance evaluation of the adopted classifiers [25] – *Accuracy rate* and *Area under the Receiver Operating Characteristic Curve* (AUC for short).

In order to provide some reassurance about the validity and non-randomness of the obtained results, the outcomes of statistical tests, following the study of Demsar [10] 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 algorithms (here classifiers).

[2] From the Weka System, Version 3.6 [4], using its default parameters, except for the number of neighbors of the K-NN classifier, for which *manual optimization* suggest that k = 11 is appropriate for the datasets in hand.

7. Experimental Results

We carried out experiments with both synthetic and real-world data. The obtained results, as well as the respective analyses, are described in Sections 7.1 and 7.2, respectively.

7.1. Synthetic Data

Tables 4 and 5 show the results of meta-learning in terms of accuracy and AUC respectively, obtained by means of Methods 1, 2 and 3. The algorithm with the best results (or if the results are not statistically worse than the best ones) for each method are presented in bold. Although the results for Method 1 have shown to be good, they are not exciting – though not disappointing. In this sense, please note that the transformation of the multi-labeled instances into several single-labeled instances naturally lead to inconsistent instances (by definition), which make the learning process more difficult.

The accuracy and AUC values obtained by means of Method 2 are higher than those achieved with Method 1. This is actually an expected result because a number of inconsistent instances – the multi-labeled ones – have been removed from the dataset used by Method 1 (e.g., compare Figs 2 to 3). Therefore, the results obtained with the two methods are not really comparable. Additionally, we must take into account that Method 2 reduces the number of training instances, what may be an additional problem – particularly when small datasets are available.

Finally, the results obtained with the third Method are clearly better than with the other two.[3] The best results were obtained by decision trees (DT) in the datasets generated by Method 3 (decomposition of the multi-

[3] Even if they are not comparable with the results of Method 2 for the reasons given earlier.

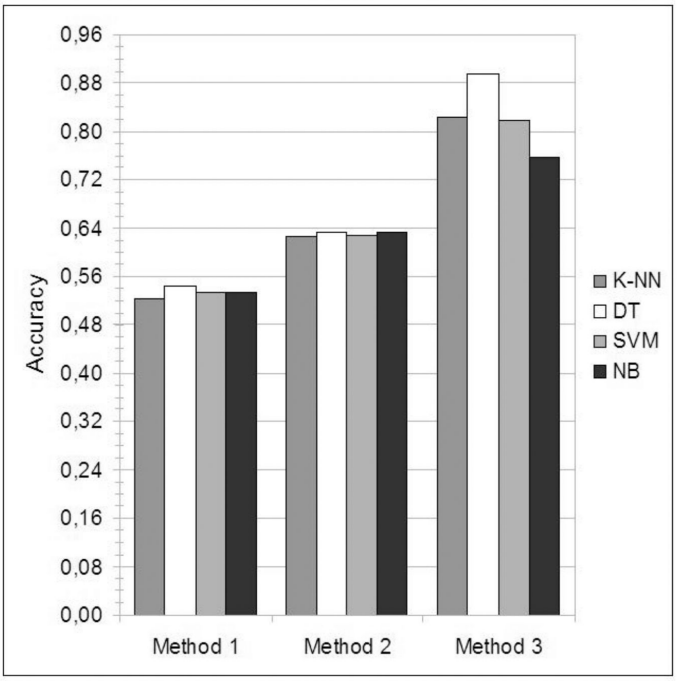


Fig. 7. Summary of the accuracies for each multi-label classification method – synthetic data.

Table 6
Classifier rankings for accuracy – synthetic data

Dataset from method	K-NN	DT	SVM	NB
Method 1	4	1	2.5	2.5
Method 2	4	1	3	2
Method 3 – Class TS	3	1	2	4
Method 3 – Class GRASP	2	1	3	4
Method 3 – Class SA	3	1	2	4
Method 3 – Class GA	2	1	3	4
Average Ranking	3.00	1.00	2.58	3.42

Table 7
Classifier rankings for AUC – synthetic data

Dataset from method	K-NN	DT	SVM	NB
Method 1	1	2	3	4
Method 2	1	4	2	3
Method 3 – Class TS	2	1	4	3
Method 3 – Class GRASP	2	1	4	3
Method 3 – Class SA	2	1	3	4
Method 3 – Class GA	2	1	4	3
Average Ranking	1.67	1.67	3.33	3.33

label classification problem into $c = 4$ binary classification problems). This method keeps the multi-label nature of the classification problem and, accordingly, seems to be the method of choice (even considering that it has the highest computational cost among the evaluated methods).

Let us now focus our attention on the summary of the accuracy and AUC results, illustrated in Figs 7 and 8, respectively. By taking into account the complexity of the problem being addressed, we shall note that such an average AUC rate (around 0.96) is very good in practical terms. Thus, this approach is promising and deserves further investigation. One may also wonder if, by taking into account all the evaluated methods (1, 2, and 3), there is a particular classifier that is better suited to this type of problem. In order to analyze this aspect, we present the results of statistical tests by following the study of Demsar [10] (Section 6). The overall procedure is based on the analysis of the rankings shown in Tables 6 and 7, where better accuracies and AUC rates are represented by smaller ranks.

Considering the accuracy, a p-value of 0.0066 was obtained from the Friedman Test. This value is the probability of obtaining results at least as extreme as those observed, assuming that the null hypothesis (equal accuracy results) is true. Usually one rejects the null hypothesis if the p-value is smaller than or equal to the adopted significance level (α). For instance, for $\alpha = 5\%$ the null hypothesis is rejected, and the Nemenyi Test is applied. This test shows that there are significant differences in pair-wise comparisons only between (K-NN, DT) and (NB, DT). For AUC, a p-value of 0.0186 was obtained from the Friedman Test. However, the Nemenyi Test failed to detect such a significant difference for $\alpha = 5\%$. As observed by Dem-

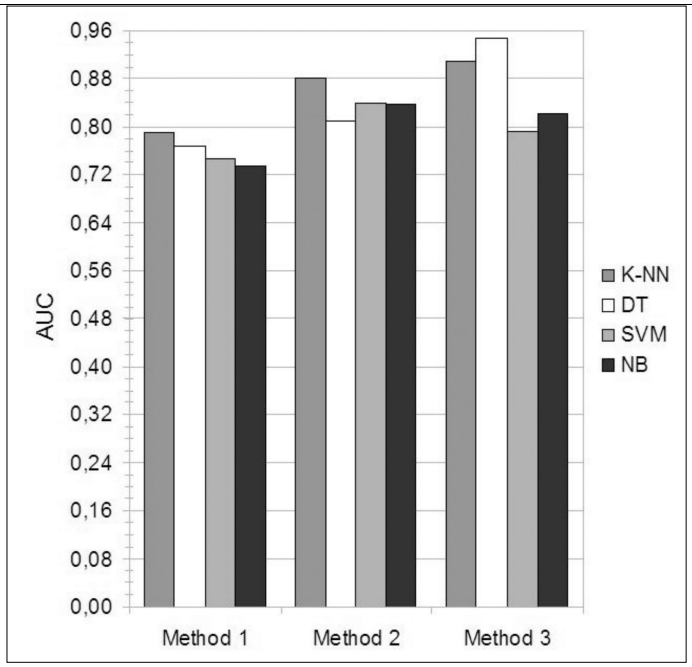


Fig. 8. Summary of the AUCs for each multi-label classification method – synthetic data.

Table 8
Accuracy results – real data

Method	K-NN	DT	SVM	NB
1	0.50±0.02	0.50±0.02	0.53±0.02	0.39±0.03
2	0.63±0.01	0.61±0.03	0.64±0.01	0.45±0.03
3	0.77±0.02	0.77±0.02	0.76±0.02	0.60±0.04

Table 9
AUC results – real data

Method	K-NN	DT	SVM	NB
1	0.66±0.08	0.63±0.06	0.66±0.07	0.67±0.06
2	0.75±0.12	0.70±0.10	0.72±0.08	0.73±0.08
3	0.72±0.08	0.66±0.12	0.60±0.09	0.71±0.06

sar [10], "sometimes the Friedman test reports a significant difference but the post-hoc test fails to detect it. This is due to the lower power of the latter. No other conclusions than that some algorithms do differ can be drawn in this case."

7.2. Real Data

Tables 8 and 9 show the obtained accuracy and AUC results, respectively. Table 8 shows that the results for Method 1 are not good, since in this case the classifiers were less accurate than the baseline measure. Although the accuracy values for Method 2 have been better if compared to Method 1, they were lower than those obtained by Method 3. From this result, we can infer that Method 3 is the most promising to be applied to the classification of similar TSP instances. Besides, the similar accuracies for the three classifiers (K-NN, DT e SVM), for each method, suggest that the computational cost of new experiments can be reduced by choosing only one of these classification algorithms for a dataset obtained from Method 3. We believe that the weak performance shown by NB was due to a probable violation of the premise of attribute independence. For the AUC criterion, Table 9 shows better results for Method 2. Despite the datasets produced by Method 3 being larger than those produced by Method 2, which tends to improve learning, we can suppose that the best performance shown by Method 2 may have been influenced by the smaller imbalance between the classes. While for the dataset produced by Method 2 had 46% of the examples in the majority class, the datasets produced by Method 3 had, in average, 73% of the examples in the majority class.

Figures 9 and 10 illustrate the accuracy and AUC results, respectively. Except for the NB, which presented the worse performance, accuracies for the other classifiers were very similar. We believe that probably the independence assumption of NB was not satisfied and therefore its predictive performance was harmed. Looking at AUC, one can see that Method 2 has shown to be slightly better than the others. By taking into account that accuracy and AUC are different measures for

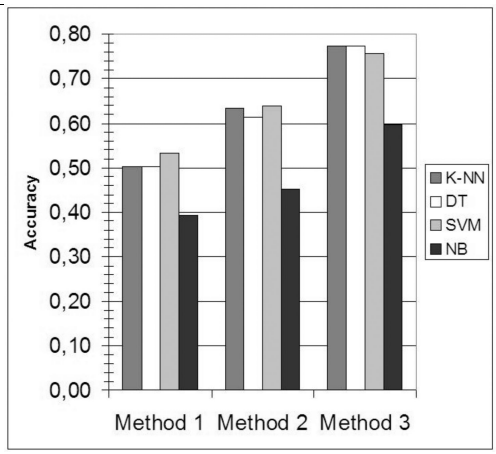


Fig. 9. Summary of the accuracies for each multi-label classification method – real data.

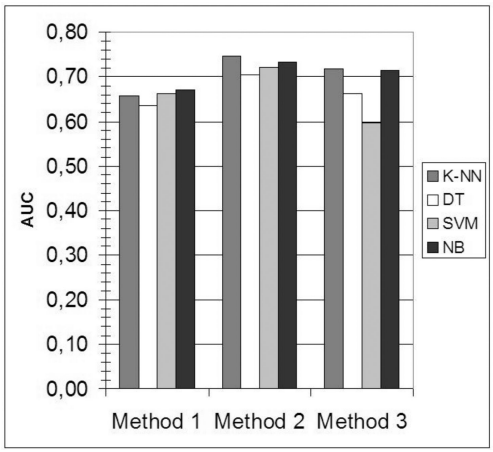


Fig. 10. Summary of the AUCs for each multi-label classification method – real data.

Table 10
Accuracies for each dataset (number of cities) – Method 3 (real data)

Dataset	K-NN	DT	SVM	NB
60 cities	0.76±0.13	**0.77**±0.12	**0.77**±0.12	0.60±0.21
70 cities	0.81±0.04	**0.82**±0.03	**0.82**±0.04	0.53±0.14
80 cities	**0.81**±0.07	**0.81**±0.07	**0.81**±0.07	0.46±0.11
90 cities	0.74±0.18	**0.75**±0.17	0.73±0.17	0.59±0.15
100 cities	0.76±0.14	0.75±0.16	**0.77**±0.14	0.73±0.13

estimating a classifier predictive performance, we cannot expect their results to show similar trends. However, further analysis is necessary in order to identify these divergent trends.

Given that Method 3 provided better accuracies than methods 1 and 2, let us now consider only this method and focus our attention on the performances of each classifier for instances with the same number of cities. Table 10 reports the achieved average accuracies. DT and SVM classifiers have shown better results for most of the datasets. Similarly, since Method 2 presented the best AUC results, we will analyze the AUC values for the datasets produced by this method, which are shown in Table 11. Although K-NN was superior in 60% of the datasets, the values are similar, making difficult to choose one of the classifiers.

Table 11
AUC results for each dataset (number of cities) – Method 2 (real data)

Dataset	K-NN	DT	SVM	NB
60 cities	**0.69**±0.11	0.63±0.12	0.65±0.11	0.68±0.09
70 cities	0.73±0.17	0.71±0.17	0.72±0.20	**0.78**±0.17
80 cities	**0.72**±0.16	0.67±0.16	0.69±0.14	**0.72**±0.12
90 cities	0.58±0.15	**0.60**±0.14	0.59±0.14	0.56±0.10
100 cities	**0.69**±0.16	0.66±0.12	0.65±0.09	0.64±0.14

Table 12
Classifier rankings for accuracy – real data

Dataset from method	K-NN	DT	SVM	NB
Method 1	2.5	2.5	1	4
Method 2	2	3	1	4
Method 3 – Class TS	3	1.5	1.5	4
Method 3 – Class GRASP	1	2	3	4
Method 3 – Class SA	1.5	1.5	3	4
Method 3 – Class GA	1	3	2	4
Average Ranking	1.83	2.25	1.92	4.00

Table 13
Classifier rankings for AUC – real data

Dataset from method	K-NN	DT	SVM	NB
Method 1	3	4	2	1
Method 2	1	4	3	2
Method 3 – Class TS	2	4	3	1
Method 3 – Class GRASP	1	2	4	3
Method 3 – Class SA	1	3	4	2
Method 3 – Class GA	1	3	4	2
Average Ranking	1.50	3.33	3.33	1.83

Tables 12 and 13 report the ranks used by the adopted statistical procedure. Considering the accuracy, a p-value of 0.0083 was obtained from the Friedman Test. In pair-wise comparisons, significant differences ($\alpha = 5\%$) were only observed between (K-NN, NB) and (SVM, NB). For the AUC measure, a p-value of 0.0169 was obtained from the Friedman Test, but the Nemenyi test failed to detect statistical differences in pair-wise comparisons ($\alpha = 5\%$). From this standpoint, a more conservative conclusion would be that further studies are necessary in order to conclude if some particular classifier is better suited for this kind of problem.

In spite of the DT classifier did not present the best performance in all datasets is the first option for future

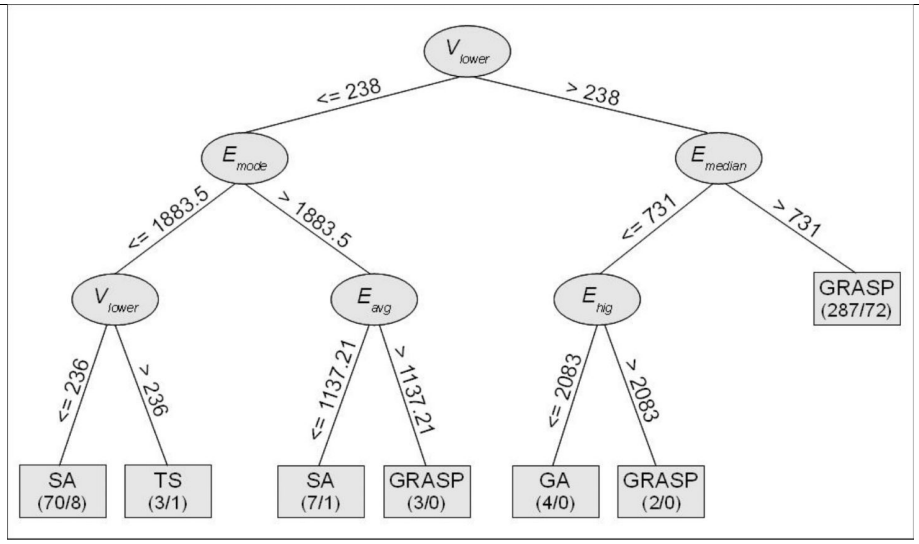


Fig. 11. Decision tree for dataset of 70 cities – Method 2 (real data).

experiments. It is computationally efficient and has a clear knowledge representation, which allows the identification of the most relevant attributes for the meta-learning process. Figure 11 illustrates a decision tree induced for the classification of TSP instances with 70 cities generated from Method 2. The meta-features V_{lower}, E_{mode}, E_{median}, E_{avg} and E_{hig} produced the highest information gain and were the only attributes selected by the learning algorithm for the induction of the decision tree. These meta-features are represented by non-leaf nodes, while classes are identified by leaf nodes. Each leaf node shows the quantity of instances assigned to the class and the quantity of misclassification.

8. Conclusions

There are several optimization techniques that can be successfully used to find good solutions for known TSP instances. However, it is very difficult to identify the technique that will provide the best solution for unseen TSP instances. In this article, we proposed a meta-learning approach to support the selection of the most promising optimization technique for a new TSP instance. Essentially, a learning model is trained with TSP instances for which the performance of a set of optimization algorithms is known a priori. Then, the learned model is used to predict the best algorithm for a new TSP instance. Each instance is described by meta-features that capture characteristics of the TSP that affect the performance of the optimization algorithms. Given that the best solution for a given TSP instance can be obtained by several algorithms, the meta-learning problem is tackled as a multi-label classification problem.

The success of a meta-learning approach directly depends on the correct identification of the meta-features that best relate the main aspects of a problem to the performances of the used algorithms. In this work, we identified a set of aspects from the TSP that are based on its graphical representation. These aspects originated the meta-features used as input parameters in the meta-learning system. Several illustrative experiments with both synthetic and real data, using four well-known optimization techniques, were performed to evaluate the proposed approach. Our experimental results suggest that the use of meta-learning for the selection of the best optimization techniques for a TSP is promising.

In spite of the good performance obtained by the meta-learning system, we are currently investigating other TSP aspects that can expand the set of meta-features. As stated by Brazdil et al. [**?**] "it is necessary to define which properties are important to characterize datasets and to develop meta-features that represent these properties". In this sense, we believe that the performance of the selection model can be further improved with the use of better meta-features.

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). We also thank the anonymous referees for their useful comments.

References

[1] D.L. Applegate, R.E. Bixby, V. Chvátal and W.J. Cook, *The Traveling Salesman Problem: A Computational Study*, Princeton University Press, New Jersey, 2006.

[2] M.S. Bazaraa, J.J. Jarvis and H.D. Sherali, *Linear Programming and Networks Flows*, 4th edition, John Wiley & Sons, New Jersey, 2010.

[3] M.R. Boutell, J. Luo, X. Shen and C.M. Brown, Learning multi-label scene classification, *Pattern Recognition* **37**(9) (2004), 1757–1771.

[4] R.R. Bouckaert, E. Frank, M. Hall, R. Kirkby, P. Reutemann, A. Seewald and D. Scuse, *WEKA Manual for Version 3-6-0*, University of Waikato, Hamilton, New Zealand, December 2008.

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

[6] P.B. Brazdil, C. Soares and J.P. Costa, Ranking Learning Algorithms: Using IBL and Meta-Learning on accuracy and Time Results, *Machine Learning*, Kluwer Academic Publishers, **50**(3) (2003), 251–277.

[7] R.J.G.B. Campello and E.R. Hruschka, On comparing two sequences of numbers and its applications to clustering analysis, *Information Sciences*, Elsevier Science, **179**(8) (2009), 1025–1039.

[8] A.C.P.F.L. Carvalho and A.A. Freitas, A Tutorial on Multi-Label Classification Techniques, *Studies in Computational Intelligence*, Springer-Verlag, Berlin Heidelberg, **205** (2009), 177–195.

[9] V. Cerny, Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm, *Journal of Optimization Theory and Applications*, Plenun Publishing Corporation, **45**(1) (1985), 41–51.

[10] J. Demsar, Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research* **7** (2006), 1–30.

[11] T.A. Feo and M.G.C. Resende, A probabilistic heuristic for a computationally difficult set covering problem, *Operations Research Letters*, Elsevier Science Publishers B.V., **8**(2) (1989), 67–71.

[12] M.R. Garey and D.S. Johnson, *Computer and Intractability: a Guide to the Theory of NP-Completeness*, W.H.Freeman & Co, New York, USA, 1990.

[13] L.A. Goodman and W.H. Kruskal, Measures of Association for Cross Classifications, *Journal of the American Statistical Association*, **49**(268) (1954), 732–764.

[14] F. Glover, Future paths for integer programming and links to artificial intelligence, *Computers and Operations Research*, Elsevier Science Ltd, **13**(5) (1986), 533–549.

[15] G. Gutin and A.P. Punnen (Eds.), *The Traveling Salesman Problem and Its Variations*, Kluwer Academic Publishers, Dordrechet, The Netherlands, 2002.

[16] J.H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, MIT Press, Cambridge MA, 1992.

[17] D.S. Johnson and L.A. McGeoch, *Experimental analysis of heuristics for the STSP* In: The Traveling Salesman Problem and it Variations, Chapter 9, Kluwer Academics Publishers, Netherlands, (2002), 369–443.

[18] J. Kanda, A. Carvalho, E. Hruschka and C. Soares, Using Meta-learning to Classify Traveling Salesman Problem, *2010 Eleventh Brazilian Symposium on Neural Networks*, IEEE Computer Society, (2010), 73–78.

[19] S. Kirkpatrick, C.D. Gelatt Jr. and M.P. Vecchi, Optimization by Simulated Annealing, *Science*, American Association for the Advancement of Science, **220**(4598) (1983), 671–680.

[20] T. Öncan, I. Altinel and G. Laport, A comparative analysis of several asymmetric traveling salesman problem formulations, *Computers & Operations Research*, Elsevier Ltd., **36**(3) (2009), 637–654.

[21] G. Reinelt, TSPLIB – A traveling salesman problem library, *ORSA Journal on Computing* **3**(4) (1991), 376–384.

[22] K.A. Smith-Miles, Towards Insightful Algorithm Selection For Optimisation Using Meta-Learning Concepts, *2008 International Joint Conference on Neural Networks*, IEEE Computer Society, (2008), 4118–4124.

[23] K. Smith-Miles, J. Hemert and X.Y. Lim, Understanding TSP Difficulty by Learning from Evolved Instances, *Lecture Notes in Computer Science*, Springer-Verlag, **6073** (2010), 266–280.

[24] T. Stützle, A. Grün, S. Linke and M. Rüttger, A Comparison of Nature Inspired Heuristics on the Traveling Salesman Problem, *Lecture Notes in Computer Science*, Springer-Verlag, **1917** (2000), 661–670.

[25] P-N Tan, M. Steinbach and V. Kumar, *Introduction to Data Mining*, Pearson Education, Inc., Boston, 2006.

[26] G. Tsoumakas and I. Katakis, Multi-Label Classification: An Overview, *International Journal of Data Warehousing and Mining* **3**(3) (2007), 1–13.

[27] D. Whitley, T. Starkweather and D. Shaner, The Traveling Salesman and Sequence Scheduling: Quality Solutions using Genetic Edge Recombination, in: *In Handbook of Genetic Algorithms*, L. Davis, ed., Van Nostrand Reinhold, New York, 1991, pp. 350–372.

[28] D.H. Wolpert and W.G. Macready, No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation* **1**(1) (1997), 67–82.