Meta-Learning-Based System for Solving Logistic Optimization Problems

Alan Dávila de León[1(✉)], Eduardo Lalla-Ruiz[2], Belén Melián-Batista[1], and J. Marcos Moreno-Vega[1]

[1] Departamento de Ingeniería Informática y de Sistemas, Universidad de La Laguna, La Laguna, Spain
{adavilal,mbmelian,jmmoreno}@ull.es
[2] Institute of Information Systems, University of Hamburg, Hamburg, Germany
eduardo.lalla-ruiz@uni-hamburg.de

Abstract. The Algorithm Selection Problem seeks to select the most suitable algorithm for a given problem. For solving it, the algorithm selection systems have to face the so-called cold start. It concerns the disadvantage that arises in those cases where the system involved in the selection of the algorithm has not enough information to give an appropriate recommendation. Bearing that in mind, the main goal of this work is two-fold. On the one hand, a novel meta-learning-based approach that allows selecting a suitable algorithm for solving a given logistic problem is proposed. On the other hand, the proposed approach is enabled to work within cold start situations where a tree-structured hierarchy that enables to compare different metric dataset to identify a particular problem or variation is presented.

Keywords: Meta-learning · Logistic problems · Meta-heuristics

1 Introduction

The Algorithm Selection Problem (ASP) is introduced in [1] seeking to answer the research question *"Which algorithm is the best option to solve my problem?"* under those cases where the decision-maker or solver counts with more than one algorithm for a given problem. The importance of tackling the ASP is provided by: (*i*) No Free Lunch ([2], NFL) theorem, (*ii*) the big number of available algorithms, and (*iii*) the need of trying to obtain the best possible solution, and not only a correct one.

In the related literature, some systems have been proposed to solve the ASP. The machine learning toolbox project [3], continued in Statlog [4] and METAL [5], aims to select the best algorithm for a given dataset. Furthermore, in [5] a helping system for aiding the selection of machine learning algorithms, according to the dataset is proposed. They obtain meta-features that allow to compare different datasets and, by means of that, obtain a reduced group of datasets

R. Moreno-Díaz et al. (Eds.): EUROCAST 2017, Part I, LNCS 10671, pp. 339–346, 2018.
https://doi.org/10.1007/978-3-319-74718-7_41

similar to the one at hand. Those reduced groups are later used to give a recommendation. In [6], a multilayer perceptron network is used to select the best optimization algorithm to solve the quadratic assignment problem. The above-mentioned contributions and systems are focused on recommending or choosing algorithms for a particular problem. That is why a meta-learning [7] based system may be appropriate for those scenarios where a ranking of algorithms sorted according to a provided criterion for any supported input problem is necessary.

In the context of algorithm selection systems, an usual appearing drawback is the so-called *cold start* (see [8]). It defines the disadvantage that arises in those cases where the system involved in the selection of the algorithm for providing a solution has not enough information to give an appropriate recommendation or selection. An extreme case of this problem happens when the system has no previous information for comparing the input stream. Therefore, in this work, we aim to propose, on the one hand, a meta-learning-based approach that allows selecting suitable algorithms for solving given logistic problems, and on the other hand, a tree-structured hierarchy that enables to address cold start situations.

The structure of this paper is as follows. Section 2 describes the meta-learning based system and its main features. The proposed approach for solving the cold start shortcoming appearing in algorithm selection is described in Sect. 3. The results obtained by our proposed system are discussed in Sect. 4. Finally, Sect. 5 presents the conclusions together with future research lines.

2 Meta-Learning

The selection of the best algorithm for a given problem is studied by a subfield of Machine Learning known as *meta-learning* [7,9]. It aims to improve the recommendation of algorithms for a given problem by applying machine learning methods that take into account collected data from past problems with similar features.

2.1 Meta-Learning-Based System

The algorithm selection problem (ASP) can be formally defined as follows: having a problem instance $x \in P$, with given features $f(x) \in F$, the objective is to perform a selection mapping $S(f(x))$ into the algorithms space A, with the goal of selecting the algorithm $\alpha \in A$ that maximizes the performance mapping $y(\alpha, x) \in Y$ such that $y(\alpha, x) \geq y(a, x)\,, \forall a \in A$.

In order to start the algorithm selection process by means of a meta-learning-based system (MLS), firstly is necessary to train the system. This is conducted by running the set of available algorithms $A = \{a_1, ..., a_m\}$ on a set of training instances $P_t = \{x_1, ..., x_n\}$. Each algorithm $a \in A$ is executed a certain number of iterations on each training instance in order to obtain the data related to $Y = \{y(a_1, x_1), ..., (a_m, x_n)\}$. In this work, $y(\alpha, x)$ corresponds to the objective function value obtained by running the algorithm α on the instance x. On the other hand, each instance $x \in P_t$ is analyzed to extract the features

$F = \{f(x_1), ..., f(x_n)\}$. Once the system is trained, a table whose rows correspond to each possible combination instance-algorithm is generated. This table associates to each row the features extracted from each instance $f(x)$ as well as the algorithm employed a_j. Moreover, each row contains the metrics obtained by $y(a, x)$ upon the execution of the experiment as, for example, the objective value. A machine learning algorithm whose input is the above-mentioned table is then applied. Its output provides a knowledge base used for providing the ranking of preferred algorithms based on the input instance. Once the system is prepared, when a new instance x' is provided, it firstly extracts its features $f(x')$ in order to start the algorithm selection process. The features that define the instance are subsequently included within a vector to be compared with the collected data using a machine learning algorithm. Based on that, a ranking $\tau_{x'}$ of recommended algorithms for the instance x' is provided.

The performance space Y is used to determine the preference of selecting an algorithm a instead of another for a particular instance x. This preference is expressed through a ranking of algorithms τ_x for each instance $x \in P_t$, where each algorithm gets a position according to their performance. This knowledge base is built offline from the set of training instances allowing to make recommendations with the premise that if two instances have similar features, the algorithms should exhibit a similar performance in both.

2.2 Problem and Algorithm Space

In this subsection, the problem and algorithm space are defined. The set P of problem instances is composed by 3 problems: (*i*) Travelling Salesman Problem (TSP, [10]), (*ii*) Capacitated Vehicle Routing Problem (CVRP, [11]), and (*iii*) Capacitated Vehicle Routing Problem with Time Windows (CVRPTW, [12]). Moreover, the algorithm space A used in the computational experiment is composed of 6 algorithms:

- Greedy Heuristic (GH): GH selects an unused vehicle v. Then, assign feasible nodes to vehicle v while its capacity is not exceeded. This process is repeated until a feasible solution has been created.
- Greedy Randomized Heuristic (GRH): GH but using a restricted list of candidates.
- Local Search (LS): swap environment, exhaustive sampling, and GRH.
- Greedy Randomized Adaptative Search Procedure ([13], GRASP): constructive phase = GRH, and improvement phase = LS.
- Simulated Annealing ([14], SA): GRH, initial temperature = 5, final temperature = 0.01, cooling rate = 0.01, and 100 iterations.
- Large Neighbourhood Search ([15], LNS): GRH, random destruction, degree of destruction = 0.1, and 10000 iterations.

2.3 KNN Supervised Classification

When the set of features F is available, it is possible to compare the features of the instances with each other to determine the most appropriate ranking, thus the use of a supervised classification algorithm such as k-nearest neighbors algorithm (KNN, [16]) is proposed. Because it works with rankings instead of labels, it is necessary to use a ranking aggregation methods in order to make a suitable prediction. In this work, the Borda's method [17] is used, where a score is assigned to each candidate in function of the position in which it appears in the ranking of each voter and is ordered by the total score obtained.

During the training phase, the system obtains the rankings for each problem instance. This ranking is categorized as ideal because it has all the information to properly build it. In the validation phase, it is necessary to determine the degree of success of the recommendations made. After the recommendation has been conducted, the ideal ranking is determined and compared to the recommended one in order to measure the degree of success of the system. This degree of success is measured by the distance between the rankings, for which we use the Spearman's rank correlation coefficient, where $\tau_j(i)$ is the position occupied by the candidate i in the ranking τ_j. It measures the correlation of the rankings and its value ranges between -1 and 1. The closer to 1 the higher the similarity between the rankings, while a value of -1 indicates that the rankings are inverse.

$$\rho(\tau_1, \tau_2) = 1 - \frac{6\sum_{i=1}^{|A|}(\tau_1(i) - \tau_2(i))^2}{|A| \cdot (|A|^2 - 1)} \tag{1}$$

3 Cold Start

The cold start commonly arises when tackling real-world problems because it is very difficult to train the system to solve the ASP for any optimization problem and its variants. This problem worsens in real environments where each problem has its own peculiarities and features. To address this issue, we propose a tree-structured hierarchy approach which enables to work within cold start situations, where although the system does not have previous information of an introduced logistic problem, it may count with information from a similar problem or from a generalization of it. It allows, thus, to compare different metric dataset to identify a particular problem or variation.

In Fig. 1, the tree-structured hierarchy is depicted for the considered routing problems (i.e. TSP, VRP, CVRP, and CVRPTW). Each node corresponds to a set of features for a given problem or variation. Besides, each problem is defined by its features and those of its parents. The leaf nodes correspond to more specific problems, and as it rises in the hierarchy, the problems become more abstract or general. This means a loss of information about the specific given problem, but it allows to start the algorithm selection process based on generalizations of this problem. That is, if there are no data on the input problem, instead of not recommending a ranking, the MLS ascends in the hierarchy until it detects a problem for which it has enough information. Then, the features of that problem are extracted and the recommendation is made.

For example, suppose that the MLS is trained with TSP instances. Then, the MLS receives a VRP problem in order to start the algorithm selection process.

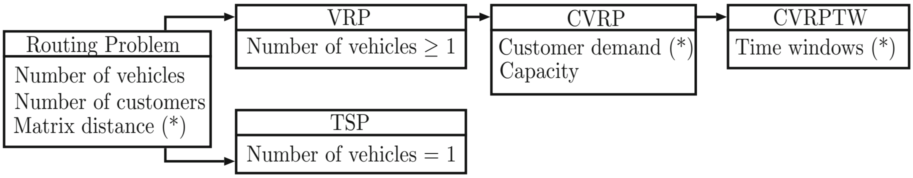

Fig. 1. Tree hierarchy of features for Rounting Problems. The features marked with an asterisk correspond to the statistics: minimum, average, maximum, standard deviation, and median

At this stage, the MLS is not able to provide a recommendation, since the TSP features do not match those extracted for the VRP. To solve it, MLS ascends in the hierarchy until detecting a problem for which it is available, (i.e. Routing Problem) and makes the recommendation based on its features.

4 Computational Results

In this experiment, we proceed to evaluate our solution approach to solve the cold start problem appearing when we proceed to start the algorithm selection process using input instance problems of the CVRPTW, but varying the problem with which the system is trained: (*i*) 37 TSP instances [18], (*ii*) 37 CVRP instances [19], and (*iii*) 56 CVRPTW instances [20].

In Tables 1 and 2 the Spearman's rank correlation coefficient ρ for three trained MLSs is depicted. The column 'Instance' show the CVRPTW instance identifier, while columns CVRPTW, CVRP, and TSP reflect the ρ value when MLS is trained with each problem. Thus, in columns CVRPTW there is no cold start problem. In this regard, the computational results report that the system can accurately predict rankings of metaheuristics, although the system has not been trained with instances of the same problem as the input instance. This information is summarized in Fig. 2, where three sets of columns are depicted. Each set of columns corresponds to a trained MLS for a single problem. For example, the 2nd set of columns is a trained MLS with CVRP instances. On the y-axis, the Spearman's rank correlation coefficient ρ is represented.

In the first set of columns, there is no cold start problem, since the system has been trained with CVRPTW instances. Therefore, this is our reference set. Note that the system is able to build the ideal ranking according to the previous data collected for similar problems, but it is not always possible although the system has been trained for the same problem. On the average case, ρ equals 0.91 which indicates a high correlation between the recommended and the ideal ranking. Finally, in the worst case, a relatively high correlation greater than zero (no correlation) is obtained, ρ equals 0.6.

In the second and third set, the system has been trained with CVRP and TSP instances, respectively. These two sets provide similar results, so they are set out together. Firstly, note that it is possible to obtain the ideal ranking even though

Table 1. Spearman's rank correlation coefficient ρ for three trained MLS and instances for CVRPTW with 25 and 50 customers.

25 customers				50 customers			
Instance	MLS_{CVRPTW}	MLS_{CVRP}	MLS_{TSP}	Instance	MLS_{CVRPTW}	MLS_{CVRP}	MLS_{TSP}
C101	0.94	0.60	0.66	C103	0.89	0.94	0.94
C105	1.00	0.66	0.77	C107	0.94	0.77	0.77
C109	0.77	0.66	0.26	C202	0.77	0.26	0.22
C204	0.66	0.83	0.83	C206	0.94	0.77	0.77
C208	0.77	0.37	0.37	R102	0.94	0.94	0.89
R104	0.83	0.60	0.37	R106	0.94	0.83	0.77
R108	0.94	0.83	0.66	R110	1.00	0.77	0.66
R112	0.94	0.77	0.60	R202	1.00	0.66	0.66
R204	0.94	0.94	0.94	R206	0.83	0.89	0.89
R208	0.94	0.94	0.94	R210	1.00	0.89	0.89
RC101	0.69	0.89	0.31	RC103	0.94	0.89	0.49
RC106	1.00	0.54	0.66	RC107	1.00	0.60	0.49
RC202	0.77	1.00	0.83	RC204	0.60	0.49	0.60
RC206	0.66	0.54	0.49	RC208	0.94	0.66	0.49
Average	0.85	0.73	0.62	Average	0.90	0.74	0.68

Table 2. Spearman's rank correlation coefficient ρ for three trained MLS and instances for CVRPTW with 100 and 200 customers.

100 customers				200 customers			
Instance	MLS_{CVRPTW}	MLS_{CVRP}	MLS_{TSP}	Instance	MLS_{CVRPTW}	MLS_{CVRP}	MLS_{TSP}
C103	0.94	0.94	0.94	C1_210	0.89	0.54	0.66
C107	0.94	0.60	0.66	C1_2_6	0.94	0.60	0.77
C202	0.94	0.89	0.89	C1_2_8	0.89	0.60	0.77
C206	0.94	0.77	0.77	C2_2_2	0.94	0.71	0.89
R102	0.94	0.89	0.77	C2_2_6	0.89	0.60	0.66
R106	0.94	0.83	0.77	R1_210	1.00	0.60	0.66
R110	0.94	0.83	0.77	R1_2_5	1.00	0.60	0.66
R202	0.89	0.89	0.89	R1_2_9	1.00	0.60	0.66
R206	0.89	0.66	0.66	R2_2_3	1.00	0.77	0.94
R210	0.94	0.77	0.77	R2_2_7	0.94	0.83	0.83
RC103	0.94	0.83	0.94	RC1_2_1	0.94	0.60	0.66
RC107	0.94	0.60	0.77	RC1_2_5	1.00	0.54	0.60
RC204	0.89	1.00	1.00	RC1_2_9	1.00	0.54	0.60
RC208	0.94	0.77	0.77	RC2_2_3	1.00	0.83	0.83
Average	0.93	0.80	0.81	Average	0.96	0.64	0.72

the system has not been explicitly trained to solve such problems, see Maximum columns. In addition, on the average case a high correlation is obtained, 0.73 and 0.71 when the system has been trained with CVRP and TSP instances, respectively, compared to the 0.91 obtained in the first set. This indicates a

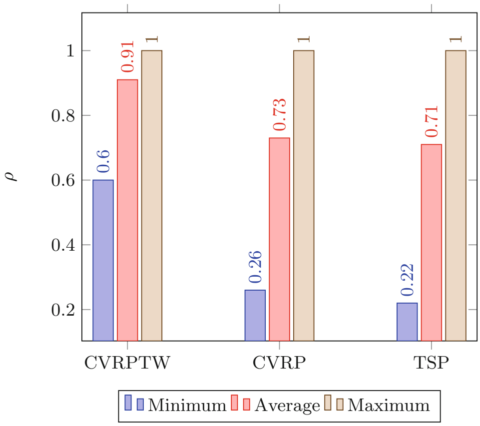

Fig. 2. Spearman's rank correlation coefficient ρ for three trained MLS.

strong relationship between the performance of the algorithms for problems of the same family where, as can be observed, there is a considerable decrease in the ρ value. Note how the values of the average and worst case decrease as the training problem differs from the input problem.

5 Conclusions

A novel way to select, from a pool of algorithms, a suitable algorithm for solving a given logistic problem has been proposed. Moreover, a new way for addressing the cold start is provided. It allows training the system with general problems and making recommendations for specific problems. We have investigated meta-features for TSP, CVRP, CVRPTW to be applied to the meta-learning process. Finally, we have shown that meta-learning models can accurately predict rankings of metaheuristics.

Additional future work includes extensively studying our approach on other well-known combinatorial problems such as those coming from the family of knapsack problems.

Acknowledgements. This work has been partially funded by the European Regional Development Fund, the Spanish Ministry of Economy and Competitiveness (project TIN2015-70226-R).

References

1. Rice, J.R.: The algorithm selection problem. Adv. Comput. **15**, 65–118 (1976)
2. Wolpert, D.H., Macready, W.G., et al.: No free lunch theorems for search. Technical report SFI-TR-95-02-010, Santa Fe Institute (1995)

3. Kodratoff, Y., Sleeman, D., Uszynski, M., Causse, K., Craw, S.: Building a machine learning toolbox. In: Steels, L., Lepape, B. (eds.) Enhancing the Knowledge-Engineering Process - Contributions from Esprit, pp. 81–108. Elsevier publishing company (1992)
4. Michie, D., Spiegelhalter, D.J., Taylor, C.C.: Machine Learning, Neural and Statistical Classification. Ellis Horwood, Chichester (1994)
5. Brazdil, P.B., Soares, C., Da Costa, J.P.: Ranking learning algorithms: using IBL and meta-learning on accuracy and time results. Mach. Learn. **50**(3), 251–277 (2003)
6. Smith-Miles, K.: Towards insightful algorithm selection for optimisation using meta-learning concepts. In: WCCI 2008: IEEE World Congress on Computational Intelligence, pp. 4118–4124. IEEE (2008)
7. Pappa, G.L., Ochoa, G., Hyde, M.R., Freitas, A.A., Woodward, J., Swan, J.: Contrasting meta-learning and hyper-heuristic research: the role of evolutionary algorithms. Genet. Program. Evolvable Mach. **15**(1), 3–35 (2014)
8. Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M.: Methods and metrics for cold-start recommendations. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 253–260. ACM (2002)
9. Brazdil, P., Carrier, C.G., Soares, C., Vilalta, R.: Metalearning: Applications to Data Mining. Springer Science & Business Media, Heidelberg (2008). https://doi.org/10.1007/978-3-540-73263-1
10. Gutin, G., Punnen, A.P. (eds.): The Traveling Salesman Problem and its Variations, vol. 12. Springer Science & Business Media, Heidelberg (2006). https://doi.org/10.1007/b101971
11. Ralphs, T.K., Kopman, L., Pulleyblank, W.R., Trotter, L.E.: On the capacitated vehicle routing problem. Math. Program. **94**(2), 343–359 (2003)
12. Cordeau, J.-F., Groupe d'études et de recherche en analyse des décisions (Montréal, Québec): The VRP with Time Windows. Groupe d'études et de recherche en analyse des décisions, Montréal (2000)
13. Feo, T., Resende, M.: Greedy randomized adaptive search procedures. J. Glob. Optim. **6**(2), 109–133 (1995)
14. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., et al.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
15. Pisinger, D., Ropke, S.: Large neighborhood search. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics. International Series in Operations Research & Management Science, vol. 146, pp. 399–419. Springer, Boston (2010). https://doi.org/10.1007/978-1-4419-1665-5_13
16. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(1), 21–27 (1967)
17. Dwork, C., Kumar, R., Naor, M., Sivakumar, D.: Rank aggregation methods for the web. In: Proceedings of the 10th International Conference on World Wide Web, pp. 613–622. ACM (2001)
18. Reinelt, G.: TSPLIB - a traveling salesman problem library. ORSA J. Comput. **3**(4), 376–384 (1991)
19. Augerat, P., Belenguer, J.M., Benavent, E., Corberán, A., Naddef, D., Rinaldi, G.: Computational results with a branch-and-cut code for the capacitated vehicle routing problem (1998)
20. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper. Res. **35**(2), 254–265 (1987)