Instance-Specific Algorithm Selection via Multi-Output Learning

Kai Chen, Yong Dou, Qi Lv, and Zhengfa Liang

Abstract: Instance-specific algorithm selection technologies have been successfully used in many research fields, such as constraint satisfaction and planning. Researchers have been increasingly trying to model the potential relations between different candidate algorithms for the algorithm selection. In this study, we propose an instance-specific algorithm selection method based on multi-output learning, which can manage these relations more directly. Three kinds of multi-output learning methods are used to predict the performances of the candidate algorithms: (1) multi-output regressor stacking; (2) multi-output extremely randomized trees; and (3) hybrid single-output and multi-output trees. The experimental results obtained using 11 SAT datasets and 5 MaxSAT datasets indicate that our proposed methods can obtain a better performance over the state-of-the-art algorithm selection methods.

Key words: algorithm selection; multi-output learning; extremely randomized trees; performance prediction; constraint satisfaction

1 Introduction

In the recent years, more and more algorithms have been proposed to solve different types of problems and tasks such as propositional satisfaction (SAT) and constraint satisfaction and planning. It becomes difficult to determine the algorithm that should be used to obtain the best performance in a certain task[1]. This leads to a challenging research area known as “algorithm selection problem”[2].

For algorithm selection, the simplest way is to select a single best algorithm for all instances. However, this strategy is inefficient to solve difficult tasks. For example, in hard SAT problems, a common case is that one algorithm is better than others in some instances but significantly worse in other instances.

An instance-specific algorithm selection method has recently been proposed to overcome this problem. Figure 1 shows the general procedure for instance-specific algorithm selection. For each instance, some features are extracted to characterize the instance, such as problem size features and clause learning features of the Conjunctive Normal Form (CNF) instance in the SAT problem[3]. These features are then fed into an algorithm selector to select an appropriate algorithm to solve the instance. The algorithm selector is generally built on the basis of machine learning methods. Based on the instance’s features and the algorithm’s performance history, a model is trained to predict the algorithm’s performance and the algorithm with the best predicted value is then selected for each individual instance. This method is successfully used in many fields. For example, SATZilla, a machine learning based algorithm selection method for the SAT problem, has won several annual SAT competitions[3,4].

For the algorithm selection methods, a growing trend is to model the underlying relations between the candidate algorithms. In fact, the algorithms are not independent of each other. For example, CryptoMiniSAT[5] and MiniSAT[6] are very similar. They belong to the complete SAT solvers category, and CryptoMiniSAT is an extension of MiniSAT. Thus,
in 2011 and 2012, a new version of SATZilla\cite{3} used the cost-sensitive random forest algorithm to model the preferable relation between each pair of SAT solvers.

To make better use of the relations between the algorithms, we propose an instance-specific algorithm selection method using multi-output learning. The multi-output learning algorithm can simultaneously predict multiple outputs using the same set of input features\cite{7}. It not only uses the relations between features and the targets directly, but also uses the relations between multiple output targets to model the preferences of the candidate algorithms. Hence, when any relation exists between output targets, multi-output learning algorithm is usually better than single-output learning algorithm\cite{8,9}. In our proposed method, three kinds of multi-output regression methods are used to predict the algorithm’s performance: (1) multi-output regressor stacking, (2) multi-output extremely randomized trees, and (3) hybrid single-output and multi-output trees. Experimental results indicate that our method can select the best algorithm very effectively.

The rest of this study is organized as follows. We review the algorithm selection methods in Section 2. In Section 3, we present three kinds of multi-output regression methods for algorithm selection. Section 4 demonstrates the experimental results of our proposed methods, and Section 5 presents our conclusion.

### 2 Related Work

In 1976, Rice\cite{2} proposed the algorithm selection problem. Since then, algorithm selection technologies have been used to select the best algorithm in different kinds of domains such as constraint satisfaction, planning, and machine learning. In this study, we mainly focus on the constraint satisfaction problem.

The algorithm selection methods proposed in the past did not use the relations between the candidate algorithms to develop the preference model of different algorithms. SATZilla 2007\cite{4} was considered an important algorithm selection method owing to its success in solving the SAT problem. Linear regression was used to predict the logged runtime of each SAT solver. The regressor is trained independently of each other in SATZilla 2007. CPHydra\cite{10} also used the runtime prediction method, based on case-based reasoning algorithm, to select algorithms. 3S\cite{11} and Snappy\cite{12} used k-Nearest Neighbor (kNN) method to perform algorithm selection. After selecting $k$ solvers using kNN, the best solver is chosen on the basis of some specific score schemes such as Penalized Average Runtime (PAR).

Then, some communities began to utilize the relations between the algorithms to perform algorithm selection more effectively. SATZilla 2012\cite{3} modeled the preference relations between each pair of SAT solvers. A cost-sensitive random forest model is trained for each pair of solvers and a voting strategy is used to select the best solver. RAS\cite{13} is a ranking-based algorithm selection method, which considers the ranking relation between the solvers. CSHC\cite{14} is an algorithm selection method based on cost-sensitive hierarchical clustering, where the calculation of cost requires the information of all candidate algorithms.

### 3 Multi-Output Regression for Algorithm Selection

To choose the best algorithm, we use multi-output learning technologies to predict the performance of the candidate algorithms. For a dataset with $N$ instances and $K$ algorithms, $D$ features are extracted from each instance and $K$ performances are collected for $K$
algorithms. Thus, the input is an \( N \times D \) matrix \( X = [x^{(1)}, \ldots, x^{(N)}]^{T} \), and the output is an \( N \times K \) matrix \( Y = [y^{(1)}, \ldots, y^{(N)}]^{T} \), where \( x^{(n)} \in \mathbb{R}^{D} \) and \( y^{(n)} \in \mathbb{R}^{K} \). The multi-output regression model is trained using \( \{X, Y\} \). In this study, we propose three kinds of multi-output regression methods for algorithm selection: (1) multi-output regressor stacking, (2) multi-output extremely randomized trees, and (3) hybrid single-output and multi-output trees.

### 3.1 Multi-output regressor stacking

Stacking is a transformation method that has been used in many machine learning problems such as the multi-label classification [15]. For multi-output learning, the stacking method transforms the multi-output learning problem to a single-output learning problem. In this study, we use extremely randomized trees (ExtraTrees) [16] method as the single-output regression algorithm. ExtraTrees is a variant of the random forest, which randomly chooses the threshold for each candidate feature at the node splitting stage when building trees. For the runtime prediction, random forest provides the best result compared with the other algorithms reported in the experiments of Ref. [17]. In our experiments, we find that ExtraTrees can obtain a better performance than random forest.

The training process of the stacking method can be divided into two stages.

**First stage:** The dataset \( \{X, Y\} \) is transformed into \( K \) datasets \( \{X_{1}, Y_{1}\}_{k=1}^{K} \). For the \( k \)-th dataset, a single-output regressor is trained and the output of the model \( \hat{Y}_{k} \) is calculated.

**Second stage:** The predictions of the first stage are added into \( X \) as the new features. So we can obtain \( X' = [x^{*}(1), \ldots, x^{*}(N)]^{T} \), where \( x^{*}(n) = [x^{(n)}, \hat{y}^{(n)}] \in \mathbb{R}^{D+K} \). Then, \( \{X', Y\} \) is also transformed into \( K \) datasets \( \{X', Y_{k}\}_{k=1}^{K} \) and \( K \) single-output regressors are trained.

It can be observed that the stacking method needs to train \( 2K \) single-output regressors. By taking the predictions of the first stage as the input features, the model of the second stage can use the information of all algorithms to correct the final predictions.

### 3.2 Multi-output extremely randomized trees

The multi-output ExtraTrees algorithm belongs to algorithmic adaptation method [18], which uses a single model to predict all the outputs. The multi-output ExtraTrees algorithm consists of multi-output decision trees that consider the information of all the outputs when splitting the nodes. The multi-output ExtraTrees can directly learn the dependencies of multiple outputs based on a single model, whereas the stacking method needs two models to learn the dependencies.

The learning process of the multi-output regression tree is similar to that of the single-output regression tree. For each node of the tree, the best variable and its value are chosen as the point to split the training samples into left and right subtrees. The splitting ends when all the outputs of the node are same or some stopping criteria is attained. A greedy approach is used to select the best split point using some score function. Suppose that the point \( s \) splits the node \( t \) into a left node \( t_{L} \) and a right node \( t_{R} \), the score function is defined as

\[
\text{Score}(s, t) = \text{var}\{y|t\} = \frac{N_{t_{L}}}{N_{t}} \text{var}\{y|t_{L}\} - \frac{N_{t_{R}}}{N_{t}} \text{var}\{y|t_{R}\}
\]

where \( N_{t}, N_{t_{L}}, \) and \( N_{t_{R}} \) represent the number of samples in the node \( t \), \( t_{L} \), and \( t_{R} \), respectively. The var is the variance of the node, which is calculated as

\[
\text{var}\{y|t\} = \sum_{i=1}^{N_{t}} \sum_{k=1}^{K} (y^{(i)}_{k} - \bar{y}_{k})^{2}
\]

where \( \bar{y}_{k} = \frac{1}{N_{t}} \sum_{i=1}^{N_{t}} y^{(i)}_{k} \) is the mean of the \( k \)-th outputs of the samples in the node \( t \). After building the regression tree, the mean values of the samples in the leaf are used as the predictions. The \( k \)-th output of the multiple predictions is

\[
\hat{y}_{k} = \frac{1}{N_{t}} \sum_{i=1}^{N_{t}} y^{(i)}_{k}
\]

where \( N_{t} \) is the number of samples in the leaf \( t \).

In the multi-output ExtraTrees algorithm, the split variable is chosen from a random subset of features, and the cut point is also randomly selected. The randomness increases the variance of each tree. Finally, the ensemble of the multi-output trees can obtain a low variance and a high accuracy model [16].

### 3.3 Hybrid single- and multi-output trees

Multi-output regression trees use the global information of all candidate algorithms. However, when the correlation between the candidate algorithms is small, the global information will reduce the accuracy of performance prediction for the single algorithm. Thus, we propose a hybrid model which combines the single-output and multi-output regression trees. For the dataset
\{X, Y\} \in \mathbb{R}^D \times \mathbb{R}^K$, we build a multi-output ExtraTrees model. Furthermore, for each output $Y_k$, we build a single-output ExtraTrees model. Then the $k$-th output of the hybrid model is defined as

$$M_k = \alpha M^m + (1 - \alpha) M^s_k$$

where $M^m$ is the multi-output ExtraTrees, $M^s_k$ is the $k$-th single-output ExtraTrees, and \( \alpha \) is the parameter that controls the trade-off between the global and the local information. The larger value of \( \alpha \) corresponds to an increased impact of multi-output learning on the entire model. A total of $K + 1$ ExtraTrees models need to be trained for the hybrid model. Table 1 provides a comparison of the number of ExtraTrees required in different methods for $K$ candidate algorithms.

### Table 1: Number of ExtraTrees (ET) in different methods for $K$ candidate algorithms.

| Method       | Number of ET | Method       | Number of ET |
|--------------|--------------|--------------|--------------|
| Single-ET    | $K$          | Multi-ET     | 1            |
| Stack-ET     | $2K$         | Hybrid-ET    | $K + 1$      |

### 4 Experiments

We implement our algorithm selection method based on scikitlearn (http://scikit-learn.org/stable/). The source code is available at: https://github.com/KaenChan/alg-sel. All experiments are conducted using a high-performance server with 32-core 2.0 GHz Intel Xeon CPU and 64 GB of memory.

#### 4.1 Datasets

To verify the performance of our proposed method, we conducted the experiments on 11 SAT datasets and 5 MaxSAT datasets from the Algorithm Portfolio Benchmark Set (APBS) (http://4c.ucc.ie/~ymalitsky/APBS.html), which is a collection of datasets to evaluate and compare algorithm selection techniques. Table 2 provides the statistics of all the datasets. The information regarding the extracted features and SAT/MaxSAT solvers can be obtained from the APBS and related resources. For example, HAND12S is provided by the University of British Columbia (UBC) group after SATzilla won the 2012 SAT Challenge (http://www.cs.ubc.ca/labs/beta/Projects/SATzilla/). The features and solvers of HAND12s are given as follows:

**125 Features:** The extracted features fall within 9 categories\(^3\), namely, problem size, variable graph, clause graph, variable-clause graph, balance, proximity to Horn formula, local search probing, clause learning, and survey propagation.

**31 Solvers:** The solvers include ebglucose, ebminisat, glucose2, glueminisat, lingeling, lrglshr, minisatpsm, mphaseSAT64, precosat, quatersat, rcl, restarting, cryptominisat2011, spear-sw, spear-hw, eagleup, sparrow, marchrw, mphaseSATm, satime11, tnm, mxc09, gnoveltyp2, sattime, sattimep, clasp2, clasp1, picosat, mphaseSAT, sapperlot, and sol.Time.

Figure 2 shows the heat-map of the pairwise output correlations for the HAND12S dataset. The deeper color represents that the pairwise outputs are more correlated. The relations between the outputs are very complex. For example, solver 8 (mphaseSAT64) is more related to solver 19 (mphaseSATm) and solver 29 (mphaseSAT). The solvers 17, 20, 21, 23, 24, and 25 are related to each other because they all belong to the incomplete solvers group. Figure 3 shows the box-plots of the distribution of all pairwise target correlations for all datasets. We can observe that the correlations of the outputs are significant for most of the datasets except WPMS Indu. RAND50S is the most correlated dataset, indicating that the 9 algorithms in RAND50S have many similar characteristics.

#### 4.2 Evaluation of different prediction targets

In SATZilla 2007, $\log_{10}$ runtime of the algorithm is chosen as the regression target. To select the best regression target for our proposed methods, we train the
Tables 3 Performance (average percentage of instances solved) comparison of different prediction targets using multi-output ExtraTrees.

| Dataset   | 1     | 2     | 3     | 4     | 5     | 6     |
|-----------|-------|-------|-------|-------|-------|-------|
| HAND12s   | 93.78 | 93.74 | 91.72 | 93.07 | 93.48 | 93.64 |
| INDU12s   | 97.37 | 97.00 | 97.02 | 97.04 | 97.52 | 97.44 |
| RAND12s   | 96.73 | 96.67 | 96.46 | 96.25 | 96.71 | 96.81 |
| HAND50s   | 90.72 | 90.07 | 86.68 | 87.97 | 87.87 | 89.06 |
| INDU50s   | 92.79 | 93.71 | 93.31 | 93.71 | 91.39 | 90.74 |
| RAND50s   | 97.52 | 97.03 | 90.64 | 95.73 | 97.32 | 97.32 |
| Mean      | 94.82 | 94.80 | 92.64 | 93.96 | 94.05 | 94.17 |

The performance metric is the average percentage of instances that are successfully solved by the selected algorithm. It is observed that the original runtime achieves the best results compared with the other prediction targets. Therefore, we use the original runtime as the regression target in the later experiments.

4.3 Comparison of random forest and ExtraTrees

We compare the mean values, variances, and training times of Random Forest (RF), and ExtraTrees (ET) on 11 SAT datasets in Table 4. For each model, the ensemble number of decision trees is 400. The models are trained using 32 threads based on sklearn. The single-output learning method needs to train $K$ models; therefore, its training time is larger than that required by the multi-output learning method. It can also be seen that the multi-output ExtraTrees method achieves the best result, and its training time is shorter than that of the RF.

4.4 Analyzing the hybrid factor

In the hybrid model of the single-output and multi-output learning, the hybrid factor $\alpha$ controls the tradeoff between the global and the local information. Figure 4 shows the effect of $\alpha$ on the performance of the algorithm selection. As seen in Fig. 4, the hybrid model achieves a better performance on the HAND50s and MaxSAT-WPMS-Indu datasets. On the SAT2011-splits...

Table 4 Comparison of RF and ExtraTrees: average and standard variance of percentage of instances solved, and the training time.

| Algorithm | Mean (%) | Std  | Training time (s) |
|-----------|----------|------|-------------------|
| Single-RF | 92.67    | 0.051| 170.98            |
| Single-ET | 93.38    | 0.052| 74.28             |
| Multi-RF  | 93.22    | 0.051| 30.94             |
| Multi-ET  | 93.43    | 0.049| 9.60              |

Fig. 2 Heat-map of the pairwise correlation coefficients of 31 solvers in the HAND12S dataset.

Fig. 3 Box-plots of the distribution of all pairwise output correlation coefficients for all datasets.
datasets, the multi-output model performs the best. For the mean result of all 16 datasets, using $\alpha = 0.8$ can afford the best result.

4.5 Comparison with state-of-the-art algorithm selection methods

In this section, we compare our proposed methods with state-of-the-art algorithm selection methods on 11 SAT datasets and 5 MaxSAT datasets.

4.5.1 Comparison on SAT datasets

Table 5 compares our methods with 3S and snappy which use kNN method to select the SAT solvers. For each method and each dataset, the average percentage of instances solved and average PAR-1 score in units of seconds are provided. The results of 3S and snappy come from Ref. [12]. VBS represents the virtual best solver, which provides the imaginary ideal result for algorithm selection. SBS represents the single best solver, where one single best solver is chosen for all instances. For the RF and ExtraTrees models, the number of trees is 400. The $\alpha$ in the hybrid model is set as 0.8. We can observe that the multi-output ExtraTrees and the hybrid model achieve the best results for two datasets. On the SAT-2012-comp-f2 dataset, the 3S method provides the best performance. Moreover, the hybrid model achieves the best average performance on all datasets.

Table 6 compares the average percentage of instances solved and average PAR-10 score in seconds of our methods with those of SATZilla, RAS, 3S, and CSHC on 6 SAT datasets. The results of SATZilla, 3S, and CSHC come from Ref. [14] and the results of RAS come from Ref. [13]. In our model, the number of trees is set as 400. The $\alpha$ in hybrid model is set as 0.8. It can be seen that the results of the stacking method are worse than those observed for the single-output RF and single-output ExtraTrees method. The multi-output ExtraTrees achieves similar results as those of the single-output learning methods. The hybrid model achieves the best results on two datasets and along with achieving the best average performance on all the datasets.

4.5.2 Comparison on MaxSAT datasets

Table 7 compares performance of our proposed methods with 3S, SATZilla, and CSHC on 5 MaxSAT datasets. The average percentage of instances solved and average PAR-10 score in seconds are reported for each method on each dataset. The results of 3S, SATZilla, and CSHC come from Ref. [14]. In our models, the number of trees is set as 400. The $\alpha$ in hybrid model is set as 0.8. It can be seen that the results of the multi-output learning are better than the single-output learning on most of the datasets. 3S achieves the best results on MS Crafted and WPMS Indu datasets. CSHC provides the best results on the WPMS Crafted dataset. The multi-output ExtraTrees method provides the best results on the PMC Crafted and PMS Indu datasets. Moreover, CSHC achieves the best performance on the average result of all the datasets, whereas Hybrid-ET can obtain
Table 5  Comparison with 3S and snappy on 5 SAT datasets: average percentage of instances solved (%) (average PAR-1 score in seconds). VBS: virtual best solver. SBS: single best solver.

| Benchmark | SAT2011-splits | SAT2012-10fold-f1 | SAT2012-comp-f1 | SAT2012-10fold-f2 | SAT2012-comp-f2 | Mean |
|-----------|----------------|--------------------|-----------------|--------------------|-----------------|------|
| VBS       | 100 (200)      | 99.83 (62)         | 94.40 (235)     | 99.83 (62)         | 94.40 (235)     | 97.69 |
| SBS       | 64.57 (1980.1) | 64.79 (809.7)      | 38.26 (1348.9)  | 64.79 (809.7)      | 38.26 (1348.9)  | 54.13 |
| 3S        | 91.23 (773)    | 96.59 (174)        | 83.05 (556)     | 97.23 (146)        | 85.42 (499)     | 90.70 |
| snappy    | 94.52 (513)    | 96.48 (162)        | 83.77 (561)     | 96.17 (168)        | 85.42 (526)     | 91.27 |

Single-RF 94.61 (508) 97.29 (135) 83.24 (504) 97.23 (139) 83.22 (475) 91.12
Single-ET 94.62 (506) 97.30 (135) 83.07 (556) 97.16 (140) 83.35 (476) 91.10
Stack-ET 93.91 (538) 96.99 (143) 82.77 (490) 97.03 (144) 82.64 (483) 90.67
Multi-ET 94.83 (493) 97.60 (130) 84.16 (450) 97.35 (135) 83.92 (459) 91.57
Hybrid-ET 94.64 (498) 97.61 (129) 84.47 (451) 97.33 (135) 84.07 (457) 91.62

Table 6  Comparison with SATZillaRAS and 3S on 6 SAT datasets: average percentage of instances solved (%) (average PAR-10 score in seconds). VBS: virtual best solver. SBS: single best solver.

| Benchmark | HAND12S | INDU12S | RAND12S | HAND50S | INDU50S | RAND50S | Mean |
|-----------|---------|---------|---------|---------|---------|---------|------|
| VBS       | 100 (108) | 100 (87) | 100 (47) | 100 (478) | 100 (339) | 100 (228) | 100 |
| SBS       | 68.38 (3888) | 89.82 (1341) | 85.28 (569) | 73.30 (18192) | 83.85 (8469) | 69.50 (15415) | 78.36 |
| SATZilla  | -       | -       | -       | 89.5 (5760) | 92.1 (4444) | 98.2 (1165) | -   |
| RAS       | 81.1    | 92.6    | 97.3    | 89      | 90.5    | 92.3    | 90.47 |
| 3S        | -       | -       | -       | 81.8 (10453) | 88.0 (6638) | 96.6 (2004) | -   |
| CSHC      | 91      | 96.6    | 97      | 90.9 (5169) | 93.1 (4093) | 99.0 (870) | 94.60 |

Single-RF 93.52 (885) 97.36 (415) 97.02 (421) 91.45 (4759) 92.15 (4231) 97.60 (1389) 94.57
Single-ET 93.30 (909) 97.50 (397) 97.00 (423) 92.27 (4368) 92.40 (4187) 97.40 (1489) 94.98
Stack-ET 92.70 (979) 96.96 (465) 96.46 (481) 90.73 (5154) 92.97 (3905) 96.25 (2033) 94.35
Multi-ET 94.19 (806) 97.56 (390) 96.67 (456) 91.00 (4999) 93.00 (3848) 97.56 (1413) 95.00
Hybrid-ET 94.08 (818) 97.76 (367) 96.71 (452) 92.00 (4500) 93.30 (3679) 97.56 (1412) 95.24

Table 7  Comparison with SATZillaRAS and 3S on 5 MaxSAT datasets: average percentage of instances solved (%) (average PAR-10 score in seconds). VBS: virtual best solver SBS: single best solver.

| Benchmark | MS Crafted | PMS Crafted | PMS Indu | WPMS Crafted | WPMS Indu | Mean |
|-----------|------------|-------------|----------|--------------|-----------|------|
| VBS       | 100 (111)  | 100 (40)    | 100 (56) | 100 (39)     | 100 (64)  | 100  |
| SBS       | 99.38 (224)| 83.55 (3083)| 92.64 (1388)| 70.91 (574)| 98.82 (324)| 89.06|
| 3S        | 99.4 (224) | 95.3 (1014) | 96.4 (728) | 91.4 (1683) | 100 (132) | 96.50|
| SATZilla2012 | 99.3 (228) | 99.3 (155)  | 98.1 (412) | 95.0 (948)  | 96.6 (718) | 97.66|
| CSHC      | 99.4 (256) | 99.3 (196)  | 98.3 (391) | 97.0 (609)  | 98.3 (421) | 98.46|

Single-RF 98.75 (336) 97.91 (402) 98.34 (364) 94.55 (1021) 97.92 (446) 97.49
Single-ET 98.75 (337) 97.97 (390) 98.53 (330) 94.55 (1022) 98.02 (431) 97.56
Stack-ET 99.38 (224) 98.41 (321) 98.67 (310) 94.67 (998) 98.59 (336) 97.94
Multi-ET 99.38 (235) 99.35 (152) 99.48 (159) 94.37 (1054) 98.38 (366) 98.19
Hybrid-ET 99.38 (235) 99.35 (152) 99.43 (168) 94.55 (1023) 98.49 (348) 98.24

better results on more datasets with a slightly lower average performance.

5 Conclusion

In this study, we propose an instance-specific algorithm selection method based on multi-output learning. To make better use of the relations between the algorithms, three kinds of multi-output regression algorithms are proposed to predict the algorithm’s performance: (1) multi-output regressor stacking which transforms the multi-output learning problem to single-output learning problem, (2) multi-output ExtraTrees which can predict multiple outputs using one model, and (3) hybrid model which combines single-output and multi-output trees to trade-off the global and local information of all outputs. The experimental results indicate that our proposed methods can achieve a better performance than the state-of-the-art algorithm selection methods.
Acknowledgment

This work was mainly supported by the National Natural Science Foundation of China (Nos. 61125201, 61303070, and U1435219).

References

[1] L. Kotthoff, Algorithm selection for combinatorial search problems: A survey, Ai Magazine, vol. 35, no. 3, pp. 48–60, 2012.
[2] J. R. Rice, The algorithm selection problem, Advances in Computers, vol. 15, pp. 65–118, 1976.
[3] L. Xu, F. Hutter, H. Hoos, and K. Leyton-Brown, Evaluating component solver contributions to portfolio-based algorithm selectors, in Proceedings of the 15th International Conference on Theory and Applications of Satisfiability Testing, 2012, pp. 228–241.
[4] L. Xu, F. Hutter, and H. H. Hoos, SATzilla: Portfolio-based algorithm selection for SAT, Journal of Artificial Intelligence Research, vol. 32, pp. 565–606, 2008.
[5] M. Soos, K. Nohl, and C. Castelluccia, Extending sat solvers to cryptographic problems, in International Conference on Theory and Applications of Satisfiability Testing, 2009, pp. 244–257.
[6] N. Eén and N. Sörensson, An extensible sat-solver, in Theory and Applications of Satisfiability Testing, 6th International Conference, Santa Margherita Ligure, Italy, 2003, pp. 502–518.
[7] G. Tsoumakas, E. Spyromitros-Xioufis, A. Vrekou, and I. Vlahavas, Multi-target regression via random linear target combinations, in ECML PKDD 2014, 2014, pp. 225–240.
[8] Z. Han, Y. Liu, J. Zhao, and W. Wang, Real time prediction for converter gas tank levels based on multi-output least square support vector regressor, Control Engineering Practice, vol. 20, no. 3, pp. 1400–1409, 2012.
[9] T. Aho, B. Enkö, S. eroski, and T. Elomaa, Multi-target regression with rule ensembles, Journal of Machine Learning Research, vol. 13, no. 1, pp. 2367–2407, 2012.
[10] E. O’Mahony, E. Hebrard, A. Holland, C. Nugent, and B. O’Sullivan, Using case-based reasoning in an algorithm portfolio for constraint solving, in Irish Conference on Artificial Intelligence & Cognitive Science, 2013.
[11] S. Kadioglu, Y. Malitsky, A. Sabharwal, H. Samulowitz, and M. Sellmann, Algorithm selection and scheduling, in Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming, 2011, pp. 454–469.
[12] H. Samulowitz, C. Reddy, A. Sabharwal, and M. Sellmann, Snappy: A simple algorithm portfolio, in Proceedings of the 16th International Conference on Theory and Applications of Satisfiability Testing, 2013, pp. 422–428.
[13] R. J. Oentaryo, S. D. Handoko, and H. C. Lau, Algorithm selection via ranking, in AAAI Conference on Artificial Intelligence, 2015.
[14] Y. Malitsky, A. Sabharwal, H. Samulowitz, and M. Sellmann, Algorithm portfolios based on cost-sensitive hierarchical clustering, in IJCAI’13 Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, 2013, pp. 608–614.
[15] S. Godbole and S. Sarawagi, Discriminative methods for multi-labeled classification, in Advances in Knowledge Discovery and Data Mining, Springer, 2004, pp. 22–30.
[16] P. Geurts, D. Ernst, and L. Wehenkel, Extremely randomized trees, Machine Learning, vol. 63, no. 1, pp. 3–42, 2006.
[17] F. Hutter, L. Xu, H. H. Hoos, and K. Leyton-Brown, Algorithm runtime prediction: Methods & evaluation, Artificial Intelligence, vol. 206, pp. 79–111, 2014.
[18] H. Borchani, G. Varando, C. Bielza, and P. Larraaga, A survey on multi-output regression, Wiley Interdisciplinary Reviews Data Mining & Knowledge Discovery, vol. 5, no. 5, pp. 216–233, 2015.