International Journal of Data Science and Analytics
https://doi.org/10.1007/s41060-020-00229-x

REGULAR PAPER

When algorithm selection meets Bi-linear Learning to Rank: accuracy and inference time trade off with candidates expansion

Jing Yuan[1] · Christian Geissler[1] · Weijia Shao[1] · Andreas Lommatzsch[1] · Brijnesh Jain[1]

Received: 20 January 2020 / Accepted: 16 July 2020

Abstract
Algorithm selection (AS) tasks are dedicated to find the optimal algorithm for an unseen problem instance. With the knowledge of problem instances' meta-features and algorithms' landmark performances, Machine Learning (ML) approaches are applied to solve AS problems. However, the standard training process of benchmark ML approaches in AS either needs to train the models specifically for every algorithm or relies on the sparse one-hot encoding as the algorithms' representation. To escape these intermediate steps and form the mapping function directly, we borrow the learning to rank framework from Recommender System (RS) and embed the bi-linear factorization to model the algorithms' performances in AS. This Bi-linear Learning to Rank (BLR) has proven to work with competence in some AS scenarios and thus is also proposed as a benchmark approach. Thinking from the evaluation perspective in the modern AS challenges, precisely predicting the performance is usually the measuring goal. Though approaches' inference time also needs to be counted for the running time cost calculation, it's always overlooked in the evaluation process. The multi-objective evaluation metric Adjusted Ratio of Root Ratios (A3R) is therefore advocated in this paper to balance the trade-off between the accuracy and inference time in AS. Concerning A3R, BLR outperforms other benchmarks when expanding the candidates range to $TOP3$. The better effect of this candidates expansion results from the cumulative optimum performance during the AS process. We take the further step in the experimentation to represent the advantage of such $TOPK$ expansion, and illustrate that such expansion can be considered as the supplement for the convention of $TOP1$ selection during the evaluation process.

Keywords Algorithm selection · Bi-linear Learning to Rank · Multi-object evaluation · Candidates expansion

1 Introduction

In the *Algorithm Selection* domain, for scenarios like computational complexity and machine learning, the number of problem instances can be infinite, while a bunch of new algorithms are created for solving problem instances every year. In a specific scenario, the performance of an algorithm on different problem instances varies a lot, and thus correctly foretell the performances of algorithms on problem instances is critical for finding the good algorithm. The research problem of how to effectively select a good algorithm given a specific problem instance has been raised since year 1975 by Rice [1]. A per-instance Algorithm Selection (AS) problem can be formulated as $\mathbf{A} \times I \rightarrow \mathbb{R}$, where set $\mathbf{A} = \{A_1, A_2, \ldots, A_n\}$ represents the set of all available algorithms in a scenario, and I denotes a specific problem instance in this scenario. Overall performances algorithms behave on a problem instance are embedded in the space $\mathbb{R}$. Using brute force to traverse all the algorithms tells the exact performance and helps select the best algorithm precisely, whereas it is often time consuming. In order to speed up the algorithm selection process in the formulated problem, AS approaches need to sacrifice the chance of only giving back the absolute perfect algorithm and yet strive to find as close

✉ Jing Yuan
jing.yuan@campus.tu-berlin.de

Christian Geissler
christian.geissler@dai-labor.de

Weijia Shao
weijia.shao@campus.tu-berlin.de

Andreas Lommatzsch
andreas.lommatzsch@dai-labor.de

Brijnesh Jain
brijnesh.jain@dai-labor.de

[1] Technische Universität Berlin, TEL-14, Ernst-Reuter-Platz 7, 10587 Berlin, Germany

Published online: 09 October 2020

as possible to that of the perfect algorithms on instance set I [2].

Single Best or *Average Rank* makes use of the landmark features, i.e., performance values of algorithms on the problem instances [3,4]. They pick only one well-performed algorithm as the suggestion for all the problem instances in a scenario. However, choosing a single algorithm for all the unseen problem instances is not always a good way; it's possible that one algorithm performs better on many problem instances, but dramatically worse on other minor instances [5]. For the sake of increasing the coverage of the well solved instances, the per-instance algorithm selection has been proposed. It creates the possibility that every problem instance is treated individually and obtains their own optimal algorithm, thereby increasing the selection effect. Take an example, Propositional Satisfiability Problem (SAT) has been commonly solved by Machine Learning (ML) approaches. It is one of the most fundamental problems in computer science, and many other NP-complete problems can be converted into SAT and be solved by SAT solvers [5]. Thus algorithm competitions toward SAT problems are held every year in the community[1]. The frequent winner $SATzilla$ in the competition uses ML to build an *empirical hardness model* to serve as the basis for an *algorithm portfolio*. The model forms a computationally inexpensive predictor based on the features of the instance and algorithm's past performances [5,6]. The strategy of running the *algorithm portfolio* can be either sequential or parallel or the combination of the two [5,7]. Though the running time is sacrificed especially in the sequential cases, the solved ratio has been increased.

When formulating AS problem from the view of ML, it can be abstracted in different models [8–10]. More specifically: $SATzilla^*$ applies pair-wise performance prediction from random forest classifiers [5,6], $LLAMA$ creates the multi-class classification model to attribute a problem instance with meta features to an algorithm class [11], $ISAC$ aggregates the similar training instances as a subset via clustering or k-NN and find the best algorithm on set basis for a new problem instance [12,13]. Facing multiple ML-based AS approaches, Auto-Folio realized the process of locating the best AS approach in the combined searching space [14,15]. Similarly, in the ML scenarios, automatically selecting proper algorithms and their hyper parameter configuration for algorithm on a specific dataset is the main purpose. *AutoML* tools like AutoWeka [16] and AutoSklearn [17] are quite popular for the algorithms and hyper-parameter space search.

A typical AS problem can be represented as Fig. 1. Meta features of problem instances are fully given as the full matrix on left-hand side, while performances of solvers (algorithms) applied on known problem instances form the performance matrix on the right hand side. The mapping function from meta features to the performances is expected to be learned. Given a new problem instance, the performance prediction fully relies on the meta-features vector. This full reliance makes the prediction task in AS similar as *cold start* condition in Recommender System (RS). When looking at the blocks with stars in Fig. 1, standing from the view of RS, the problem instance meta-feature input can be understood as usual user profiling features like age, working field, preference category etc. And the performance matrix can be associated with user rating or implicit feedback matrix RS. Therefore, the approaches used in RS are also applicable in AS problem. The terminologies used in AS, ML and RS occasionally overlap, and we distinguish these terminologies in Table 5 in "Appendix A.1" to avoid misunderstanding. The Examples inside the table mainly come from the definitions in the work with *TSP* Solvers by Bao et al. [18].

When applying RS approaches in the AS problems, we need to note that the recorded algorithms' performances on problem instances are usually in much smaller size. Thence the state-of-the-art deep learning and transaction embedding techniques in the large-scale session-based RS [19,20] are not suitable for AS scenarios. On the contrary, shallow ML approaches from RS are more adaptable. Since 2010, Stern et al. have applied *Bi-linear Matrix Factorization* (originally designed for RS) in AS scenarios and got some good results [21,22]. Thereafter, many researchers tried the approaches from RS to solve AS tasks. Misir and Sebag created Alors AS system, which utilized random forest to map meta-features of problem instances onto the latent feature space. Based on these latent features, their Collaborative Filtering (CF) is designed to make algorithm recommendation [9,23]. Yang et al. proposed *Principal Component Analysis* to decompose the performance matrix actively to solve the sparse performance entries problem for the new problem instances [24].

Learning to Rank (L2R) as a famous RS framework has been proposed to learn the prediction model from the ranking of the recommended list [25–27] and is also applicable in AS. As summarized in [28], L2R methods are usually divided into three groups: point-wise, pair-wise and list-wise. Point-wise L2R is designed for the labeled ranks, and thus multi-classes classification ML models can be used. Pair-wise L2R works well for the recommendation with large amount of candidate items. Owing to the pairs sampling from the lengthy candidates list, time cost can be saved during learning. List-wise L2R creates the loss function through the cross entropy between the ground truth list and the predicted list. In [29], authors utilized the sigmoid function as ranking surrogate to tell the algorithms' pair-wise performance order. The surrogate embeds the polynomial scoring model function to produce the probability. However, the pair-wise L2R costs extra during the pair-wise sampling phase and list-wise L2R is more preferable for the shorter candidates list. To model the

[1] http://www.satcompetition.org/

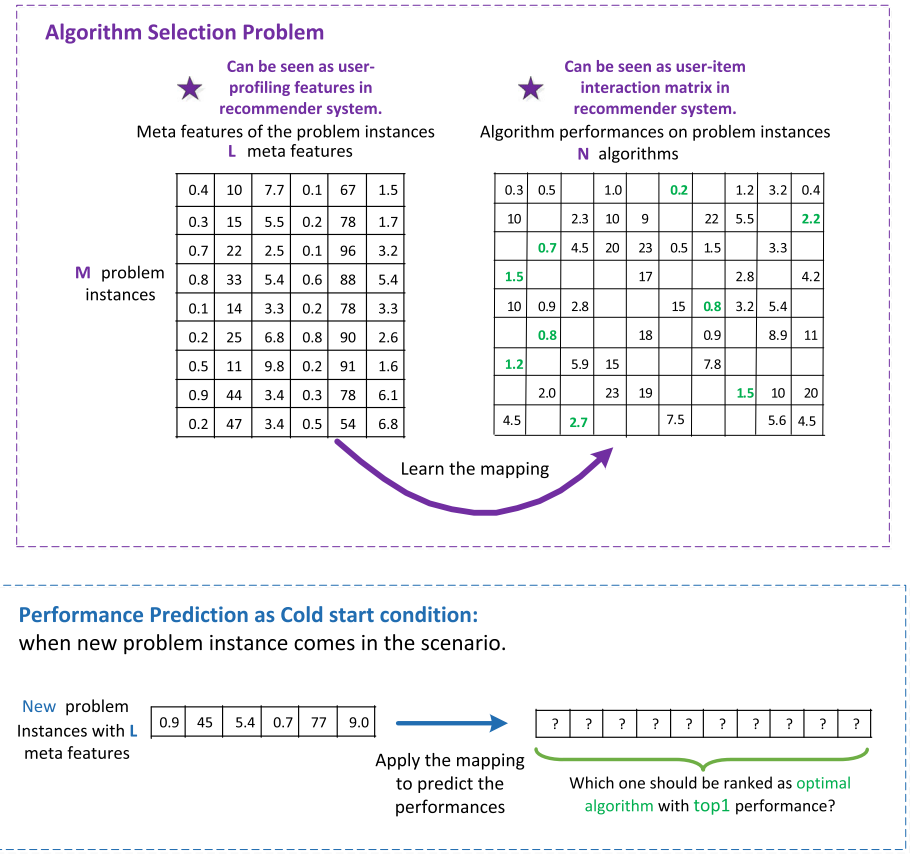

Fig. 1 Algorithm Selection as Recommendation System: we need to learn a model which maps the given meta features matrix (which size is $M \times L$) to the performance matrix (whose size is $M \times N$). Thus the inference of the newly coming problem instance in the same scenario is like making recommendation for a user in the cold-start phase in the Recommender System (RS).

uncertainty of the performance ranking, we apply list-wise L2R framework to the proposed model solving AS problems.

As the exchange for speeding up the algorithm selection process, AS approaches need to sacrifice the performance prediction accuracy to some extent. For every AS scenario, an *Oracle* or Virtual Best Solver (VBS) is assumed to know the best performed algorithm for all the instances. Reducing the gap between a proposed AS approach and the VBS is one of the evaluation goals while assessing a new AS approach. In this paper, we mainly deal with the AS problem in computational complexity scenarios like SAT, Maximum Satisfiability Problem (MAXSAT), Constraint Satisfaction Problems (CSP), Quantified Boolean Formula (QBF) and Answer Set Programming (ASP) [30–33]. In these scenarios, *run time* is the performance indicator for all candidate algorithms. The additional *runtime* cost and *solved ratio* of the predicted optimal algorithm are the main effect measurements for AS approaches [34–37]. Aside from the accuracy-oriented evaluation metrics, the inference time of AS approaches can span in many magnitudes thus also needs to be taken as a trade-off factor in the evaluation. Nevertheless, inference time is usually overlooked in the algorithm evaluation.

From the view of modeling, evaluation, and candidates selection while applying RS approaches in AS problems, there are still some open research questions: (1) if both problem meta-features and algorithms performance information are utilized for modeling, multi-models training or one-hot encoding is usually unavoidable in benchmark approaches, whether a model can skip these intermediate step and create the mapping directly? (2) During the evaluation process, the inference time from a specific AS approach is usually ignored. When both prediction accuracy and inference time are taken into account, how to balance the AS effect? (3) In most AS challenges [37,38], only the predicted optimal algorithm is chosen for the evaluation. It narrows the range of candidate set and reduces the chance of finding the actual optimal algorithm, whether a proper expansion on the candidates set can benefit the AS effect with the cumulative optimal algorithm? In order to address the research problems, we construct the following studies in this paper:

(1) We propose Bi-linear Learning to Rank (BLR) to include both problem instance meta-features and performance matrix in one L2R framework. The mapping matrix $\mathbf{W}$ and $\mathbf{V}$ in the model creates the mapping from meta-features to the performance matrix in a straightforward way. It avoids multi-models training or algorithms one-hot encoding as what other benchmark approaches do. And the probabilistic assumption on the ranking solves

the randomness modeling of the performance value in the algorithm-problem instance interaction matrix. We illustrate the good performance of BLR compared with other benchmark AS approaches in the experiments.

(2) Adjusted Ratio of Root Ratios (A3R) was proposed as a ranking measure for the algorithms in ML meta-learning; it incorporates both accuracy-oriented metric and time cost metric into one evaluation measurement. We apply A3R as the evaluation metric for the general AS tasks, in order to balance the accuracy and inference time for measuring AS approaches. Being measured with A3R, BLR outperforms other approaches in terms of this trade-off.

(3) While observing the cumulative optimal performance, we find that AS approaches usually converge to a good performance when K setting goes from 1 to 3 or 5. Though $TOP1$ candidate selection is still used in many AS challenges, we advocate expanding the this candidates selection spectrum from $TOP1$ to $TOPK$ (K depends on the concrete computational power). The error decrease effect detected in the experiment confirms the benefits of such expansion.

The rest of the paper is structured as follows: basic methodologies, benchmark approaches and concrete modeling steps of BLR are introduced in Sect. 2. In Sect. 3, we first list the evaluation metrics frequently used in AS tasks and then introduce A3R as the trade-off metric for accuracy and inference time. Section 4 presents the experiments design and the results. Finally, Sect. 5 draws the conclusion and gives an outlook to the future work.

2 Methodologies

In AS, regarding one problem instance, the predicting targets are the performances of multiple algorithms, instead of a single label or a numerical value. In order to solve the multi-targets prediction task, there are three ways to design AS approaches: (1) relying on statistics of algorithms' historical performances; (2) algorithm performances separation: building the predicting model for each algorithm individually, run the fitted models for all algorithms during the inference; (3) algorithm indicators' one-hot conversion: horizontally concatenate the problem instance meta-feature matrix and algorithm appearance one-hot matrix to form the input matrix as the input for the general prediction function. In this section, we first introduce the benchmark approaches which follow these three ways of design. Subsequently, we propose our own approach Bi-linear Learning to Rank (BLR), which doesn't need multi-models training and one-hot conversion to complete the AS model creation.

2.1 Benchmark approaches

Targeting diverse AS scenarios, some well-performed benchmark approaches have already been proposed.[2] We separate these benchmark approaches into three groups according to the data transformation ways mentioned above.

2.1.1 Performances' statistics

Virtual Best Selector and *Single Best* are two traditional benchmark approaches in AS. They don't rely on any Machine Learning (ML) model assumption of meta-features, but come from the performance statistics instead.

- *Virtual Best Selector* is the ground truth of the algorithms performances. The ranking of algorithms in VBS is the true rank used to compare with the predicted list. The evaluation of the VBS list is the upper bound for all other AS approaches.
- *Single Best* is the most classical algorithm selection baseline approach. It selects the algorithm whose mean performance is the best through all the problem instances in the training set.

2.1.2 Algorithm-based separated learning

The algorithm-based separated learning process is explained in Fig. 2. For each algorithm, a single prediction model is trained based on problem instances' meta-features and the algorithm's performances. When a new problem instance shows up, N prediction models are used to infer the performances for the N algorithms separately. The following AS approaches adopt the algorithm-based separated learning process. In spite of the model specialty for this group of approaches, long inference time is its main disadvantage.

- *Separated Linear Regressors* train linear regressors for candidate algorithms separately. When a new problem instance must be handled, the performance prediction on all algorithms depends on all the fitted linear models.
- *Separated Random Forest Regressors* fit Random Forest (RF) models for a designated algorithm. During the inference phase, N RF are called separately to generate predictions for N algorithm individually.
- *Separated Gradient Boosted Regression Trees (XGBoost)* uses gradient boosted trees to learn the performance predictor, every individual algorithm owns a XGBoost model and infer the new performance value based on its own XGBoost model.

[2] http://coseal.github.io/aslib-r/scenario-pages/

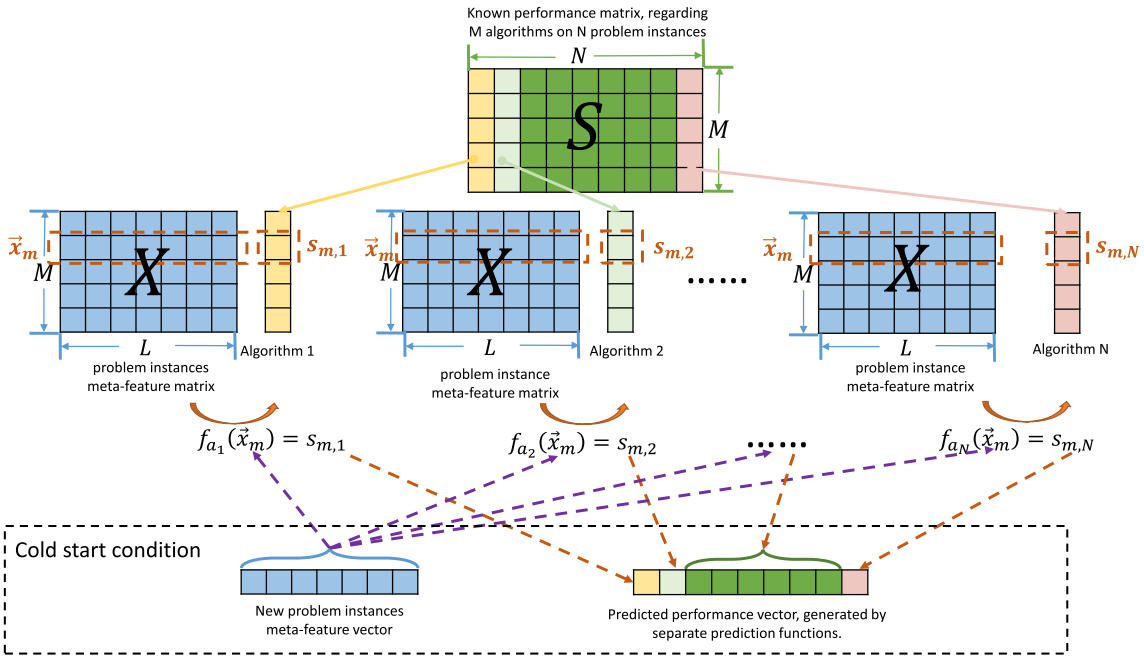

Fig. 2 Algorithm-Based Separated Learning: for each algorithm a_n, targeting on its performances $s_{:,n}$ (n_{th} column in performance matrix S), we learn the mapping function $f_{a_n}(\mathbf{x}_m) = s_{m,n}$ which infers the meta-feature vectors in X to $s_{:,n}$. N mapping functions are learned regarding N algorithms. Under cold start condition (described at the bottom dashed box), for new problem instance m, we need to apply N mapping functions $f_{a_n}(\mathbf{x}_m)$ on N algorithms separately. The recommended algorithm list is ordered according to the predicted scores.

2.1.3 Algorithms one-hot conversion

Another group of AS approaches apply the one-hot conversion of the algorithms appearance indicator to form the new AS input. Targeting on a new problem instance, concatenated vector of problem instance meta-feature and the algorithm indicator vector forms the input for the prediction model. Figure 3 represents the conversion process. Though the single model brings in the simplicity, one-hot conversion creates extra sparsity for the data. The AS approaches following this conversion rules include:

- *One-hot Linear Regressor* trains one linear predicting model with the flattened representation from the combination of problem instance meta-features and algorithms appearance indicators. Only one linear model is applied during the inference process for the new problem instances.
- *One-hot RF Regressor* has each entry in the performance matrix as the regression target, with the $L + N$ dimensional features, only RF is needed to fit the model. The model can infer any algorithm's performance with its one-hot encoded appearance indicator.
- *One-hot XGBoost* fit a single XGBoost model with $M \times N$ training samples, this XGBoost model is applicable for the performances inference for all the algorithms.

2.2 Bi-linear L2R

There are two matrices with known entries in AS scenarios. One is the problem instance meta-feature matrix X, and the other is the algorithm problem instance performance matrix S. The benchmark approaches mentioned in the above subsection solve the mapping from X to S via either multi-models training (time consuming) or algorithms' indicators' one-hot conversion (can sparsify the dataset). In order to avoid the multi-models training and features one-hot conversion, we propose Bi-linear Learning to Rank (BLR) to create the AS strategies. Given the bi-linear assumption, the factorization process of the mapping from X to S is represented in Fig. 4. The performance inference on new problem instances is depicted in Fig. 5. With the help of the two mapping latent matrices W and V, an entry in the performance matrix $s_{m,n}$ can be calculated through $\mathbf{X}_{m,:} \cdot W \cdot V_{:,n}$. Therefore, the model parameters to be learned are matrices W and V. There is no need to train specific models individually for different algorithms. Owing to the indices exact mapping, the latent dense matrix is enough to directly contribute to the entries in the performance matrix, thus the sparse one-hot encoding is not needed during the inference time.

In algorithms' performances, uncertainty always exists. For computational complexity problem, like SAT and Traveling Salesman Problem (TSP), the algorithm's *runtime*

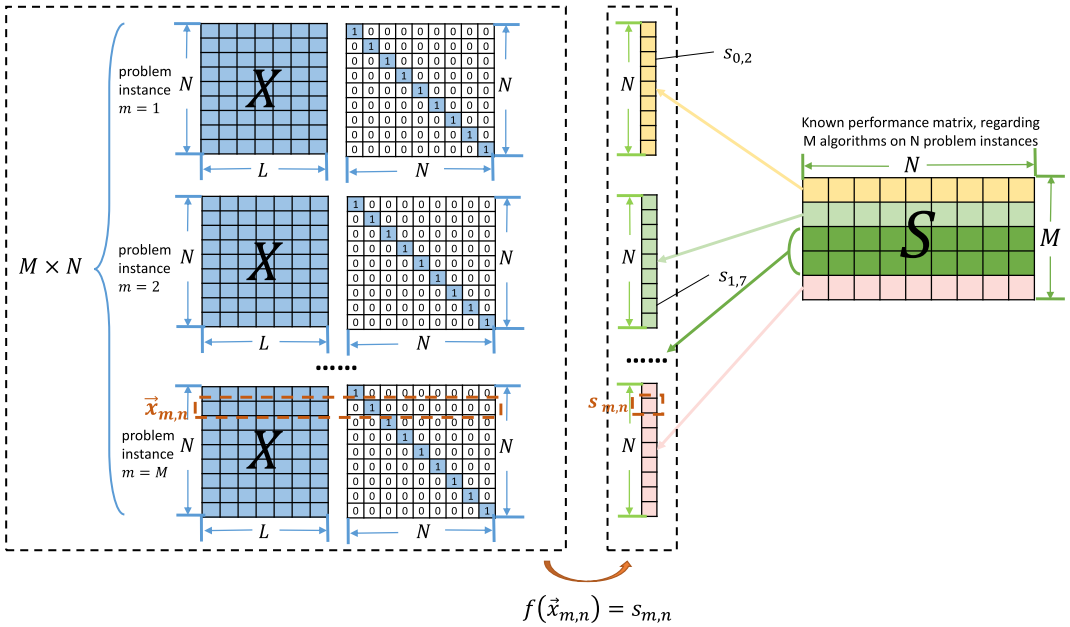

Fig. 3 Algorithm One-hot Conversion: convert the algorithm indicator as one-hot vector, combine it with the problem instance meta-feature vector to form the training input $\mathbf{x}_{m,n}$ (line within the dashed block in brown) to predict the performance of a problem instance - algorithm pair $s_{m,n}$. The general mapping function $f(\mathbf{x}_{m,n}) = s_{m,n}$ is learned from the stacked training input features with dimension $((M \times N), (L + N))$ (as shown in the dashed box on the left-hand side). The rows in the performance matrix are transposed and stacked to form the performance column as the predicting target (shown inside the dashed box in the middle).

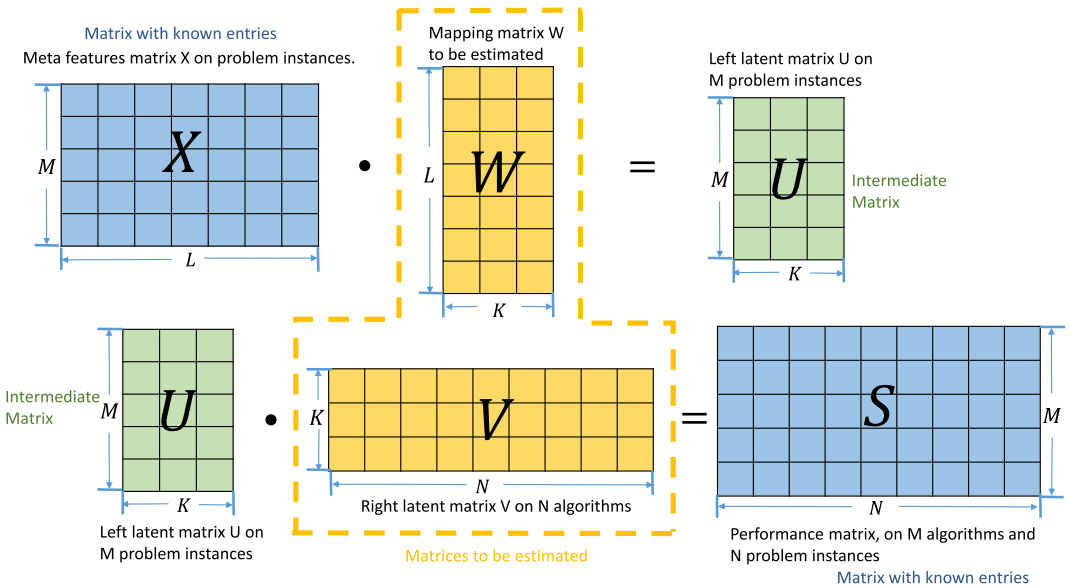

Fig. 4 Bi-linear factorization graph given the known matrices (problem instance meta-feature matrix X and performance matrix S in blue). W (in yellow) is supposed as the weighted mapping matrix for input X, to project X onto the intermediate left latent matrix U with K latent dimensions for M problem instances. The dot product of intermediate left latent matrix U and right latent matrix V (in yellow) yields the performance matrix S (in blue, known entries in the training set). Aside from the known matrices and intermediate matrices, the unknown matrices W and V in yellow are what to be estimated during the training process

performance can be different when altering the specific running environment. For ML-based problems, the accuracy measured can also be different when cross-validation setting changes. With the performance Bi-linear factorization assumption, we model the ranking of algorithms w.r.t. a specific problem instance in a probabilistic fashion. We

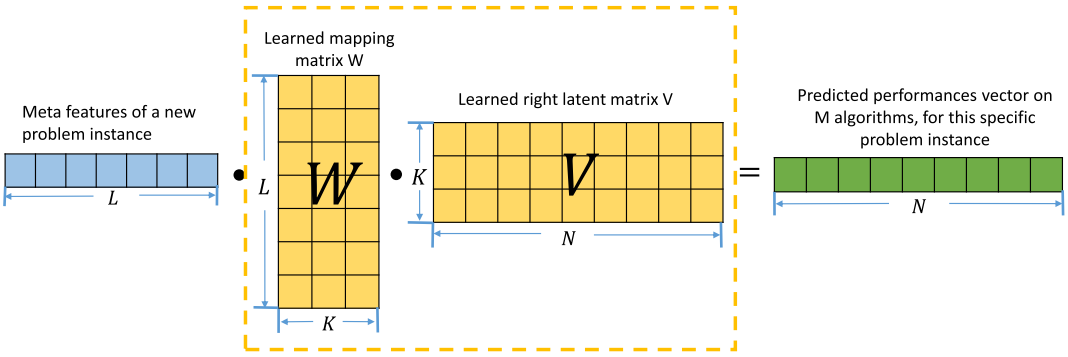

Fig. 5 Algorithm selection as a cold start problem under bi-linear Decomposition. W and V are the decomposed matrices after bi-linear factorization from problem instance meta-feature matrix and performance matrix. When a problem instance is introduced into a scenario with only its own meta-feature vector (on the left in blue), yet without any algorithm performance record. The continuous dot product on this meta-feature vector and the learned matrices W, V yields the full performance (on the right in green) vector regarding this new problem instance

assume the probability an algorithm ranked $TOP1$ for a problem instance is proportional to its performance (or predicted performance) among all the algorithms. The cross entropy between the ground truth $TOP1$ probability vector $P_{\mathbf{r}_m}(r_{m,n})$ and the predicted $TOP1$ probability vector $P_{\hat{\mathbf{r}}_m}(\hat{r}_{m,n})$ (where r is the converted value of a performance value s) defines the loss and influence the optimization strategy.

Embedding bi-linear factorization in L2R framework, this is the full idea of BLR. We refine the notations for BLR in Table 6 in "Appendix A.2". The modeling and learning of BLR is structured as four steps: (1) Performance scoring model function and corresponding rating converting function; (2) loss function considering the ranking loss; (3) gradient function for corresponding weights; and (4) updating rule of the weights according to the specific optimization approach. The first two steps are introduced as follows in this section, while gradient function and updating rules are explained in the "Appendix A.3.1 and A.3.2" separately.

2.3 Model function

In BLR, given the problem instance m and algorithm n, we predict the performance score as $\hat{s}_{m,n}$ in the Eq. (1). The preferred sorting order on performance values depends on the choice of target performance. For example, if *runtime* is performance metric, the lower value is better. However, if *accuracy* is the targeted performance metric, the higher performance is preferred. For the simplicity of calculating the list-wise ranking loss, we set a converting function $r = f(s)$ to make descending order preferable for all the rating values r. And the converted rating value r is the optimization unit in the ranking loss function. In this paper, we simply define $f(s)$ as Eq. (2).

$$\begin{aligned}\hat{s}_{m,n} &= \mathbf{u}_m \cdot \mathbf{v}_n^T \\ &= \mathbf{x}_m \times \mathbf{W} \cdot \mathbf{v}_n^T \\ &= \sum_{k=1}^{K} \left(v_{n,k} \cdot \sum_{l=1}^{L} x_{m,l} \cdot w_{l,k} \right)\end{aligned} \tag{1}$$

$$f(s) = \begin{cases} s & \text{higher performance value is preferred} \\ -s & \text{lower performance value is preferred} \end{cases} \tag{2}$$

2.4 List-wise loss function

Assuming that the performances scores of all algorithms on specific problem instance are with measuring noises, we model the probability that an algorithm being ranked top-one proportional to its normalized measured performance value. This normalized top-one probability representation has been proposed in L2R domain to model the list-wise ranking loss [28]. Regarding a single problem instance, the top-one probability for the same algorithm is different between the ground truth performances list and the predicted performances list. As defined in Eq. (3), for a problem instance m, with the rating vector $\mathbf{r}_m$ (converted version of the performance vector), the top-one probability for each algorithm n is normalized in the form of $P_{\mathbf{r}_m}$.

$$P_{\mathbf{r}_m}(r_{m,n}) = \frac{\varphi(r_{m,n})}{\sum_{n=1}^{N} \varphi(r_{m,n})} \tag{3}$$

For the sake of making probability distribution more gathered around the position of the largest input values, the exponential function is applied as the concrete form for monotonically increasing function φ in Eq. (3). Thus $P_{\mathbf{r}_m}$ can be represented as Eq. (4), which is in the same shape of

Softmax function representation.

$$P_{\mathbf{r}_m} = \frac{\exp(r_{m,n})}{\sum_{n=1}^{N} \exp(r_{m,n})} \tag{4}$$

To represent the list-wise ranking loss per problem instance, the cross entropy is calculated between the top-one probability from the predicted rating list $\hat{\mathbf{r}}_m$ and the ground truth rating value list $\mathbf{r}_m$. For each problem instance m, the point-wise loss for algorithm n is formulated as Eq. (5). Considering the probabilities normalization is calculated under the same scale for a problem instance m, the per instance list-wise loss L_m is defined as the summation of the point-wise loss inside this list, as shown in Eq. (6). Here L_m is the list-wise ranking loss between the ground truth list and the predicted list. The total loss on the whole m problem instances is defined in Eq. (7), in which L2 regularization is applied to avoid over-fitting.

$$L_{m,n} = -P_{\mathbf{r}_m}(r_{m,n}) \ln P_{\hat{\mathbf{r}}_m}(\hat{r}_{m,n}) \tag{5}$$

$$L_m = -\sum_{n=1}^{N} P_{\mathbf{r}_m}(r_{m,n}) \ln P_{\hat{\mathbf{r}}_m}(\hat{r}_{m,n}) \tag{6}$$

$$L = \sum_{m=1}^{M} L_m + \frac{\lambda}{2}\left(\|\mathbf{W}\|_F^2 + \|\mathbf{V}\|_F^2\right) \tag{7}$$

The concrete gradient calculation for the loss definition and the updating rule based on the gradient can be found in the "Appendix A.3.1 and A.3.2" separately.

3 Evaluation metrics

We measure the AS effect of different approaches with the evaluation metrics Success Rate (SUCC), Mis-Classification Penalty (MCP), Penalized Average Runtime Score (PAR10) and Mean Average Precision (MAP). In addition to these performance prediction accuracy-oriented metrics, A3R is also applied to solve the trade-off between prediction effect and inference time.

3.1 Accuracy-oriented evaluation metrics

SUCC, PAR10 and MCP are the standard evaluation metrics from AS community. SUCC cares only whether the selected algorithms are solvable. Yet for question, how close does the predicted best algorithm perform to the actual best algorithm? It is the most concern in PAR10 and MCP. Additionally, MAP is included as a representative of ranking measurement. Obeying the conventional candidate selection criteria, the selection range of algorithms is limited to $TOP1$ from the predicted list. The chance of finding the optimal algorithm is actually limited to this specific choice. In this paper, we propose expand the algorithm candidate selection range to $TOPK$ on the predicted list to gain the evaluation bonus. The four evaluation metrics with their $TOPK$ understanding are explained below.

SUCC stands for the average solved ratio of the selected algorithm per problem instances across the test set. For $TOP1$ selection criteria, the solved ratio is only calculated w.r.t. the algorithm with best predicted performance. Yet for the case of SUCC@K, the average calculation is applied over the best K algorithms.

PAR10 is the penalty version for the actual *runtime* of the selected algorithm. If the selected algorithm is actually timeout, its *runtime* will be penalized by multiplying 10 to the timeout runtime. Otherwise, the actual *runtime* is directly used. With $TOP1$ selection criteria, the penalty is only applied on the best ranked algorithm in the predicted list. For PAR10@K, the penalty will be applied on the algorithm with the shortest actual *runtime* in the $TOPK$ algorithms of the predicted list.

MCP compares the time cost difference between the actual *runtime* of the predicted best algorithm and the VBS. The algorithm with the lowest actual *runtime* in the $TOPK$ predicted list is chosen as the comparison with the *runtime* of VBS. The algorithm selected by VBS always has the MCP value as zero.

MAP measures the mean average precision of the $TOPK$ predicted algorithms vs. the $TOPK$ ranked algorithms with the ground truth performance. MAP for $TOPK$ algorithms in the predicted list is calculated in the same way as MAP@K (average of the precision rate which has a hit indicator).

Among the above evaluation metrics, accuracy-oriented ones SUCC and MAP comply with the rule *the higher the better*, while for the time cost-oriented metrics like MCP and MAP, *the lower the better*.

3.2 Multi-objective evaluation metrics

The standard AS evaluation metrics are aimed at the accuracy of the performance prediction. However, inference time on the unknown problem instances also deserves our attention. Multi-objective evaluation metric Adjusted Ratio of Root Ratios (A3R) involves both accuracy and inference time into the evaluation and brings in the trade-off between the two factors.

Abdulrahman, Salisu et al. introduced A3R in AutoML [39,40]. A3R is treated as the ranking basis for algorithms w.r.t. a dataset AutoML scenario. A3R balances the precision

and the runtime of the selected algorithm. As Eq. (8) shows, when applying algorithm a_p on dataset d_i, $SR_{a_p}^{d_i}$ stands for the success rate and $T_{a_p}^{d_i}$ represents the time cost. A reference algorithm a_q is chosen, to standardize the success rate across all the algorithms as ratio $SR_{a_p}^{d_i}/SR_{a_q}^{d_i}$. The equivalent ratio for time cost is represented as $T_{a_p}^{d_i}/T_{a_q}^{d_i}$. The combined metric takes success rate ratio as advantage, while the time ratio as disadvantage. Since the time cost ratio ranges across more magnitudes than the success rate does, N_{th} root on the denominator of Eg. 8 enables the re-scaling of the running time ratio and turns the $A3R$ to a reasonable value range. A3R is used to measure the comprehensive quality running an algorithm on the dataset.

$$A3R_{a_pa_q}^{d_i} = \frac{SR_{a_p}^{d_i}/SR_{a_q}^{d_i}}{\sqrt[N]{T_{a_p}^{d_i}/T_{a_q}^{d_i}}} \tag{8}$$

$$A3R(ACC)_{a_pa_q}^{s_i} = \frac{ACC_{a_p}^{s_i}/ACC_{a_q}^{s_i}}{\sqrt[N]{T_{a_p}^{s_i}/T_{a_q}^{s_i}}} \tag{9}$$

$$A3R(TC)_{a_pa_q}^{s_i} = \frac{\sqrt[M]{TC_{a_q}^{s_i}/TC_{a_p}^{s_i}}}{\sqrt[N]{T_{a_p}^{s_i}/T_{a_q}^{s_i}}} \tag{10}$$

In this paper, we borrow the idea of A3R from AutoML and apply it as the ranking basis for the approaches in AS scenario. We replace d_i with s_i (the i_{th} scenario), keep a but note as approach in Eq. (11). For accuracy-based metrics like SUCC and MAP, we apply their values ACC to substitute SR in the Eq. (8). While for run time cost based metrics TC, lower values denote higher accuracy, the inverse ratio $TC_{a_q}^{s_i}/TC_{a_p}^{s_i}$ is instead used in the numerator. Since the run time cost spans several magnitudes, M_{th} root is used on the numerator for re-scaling. In the following experiments, we utilize Eqs. 11 and 12 to evaluate the combined AS effect.

4 Experiments

We design the experiments to study: (1) The algorithm selection effect of the proposed BLR approach compared with other benchmark approaches; (2) AS effect when taking both *accuracy* and *inference time* into consideration; (3) the benefits of expanding the candidates set selection range.

4.1 Datasets

In this paper, we focus on typical AS problems in computational complexity domain. The Algorithm Selection Library (ASLib) released by COnfiguration and SElection of ALgorithms (COSEAL) [3] research group provides the most complete and standardized dataset over such tasks. In our experiments, we fetch the following scenarios from ASLib: *ASP-POTASSCO*, *BNSL-2016*, *CPMP-2015*, *CSP-2010*, *CSP-MZN-2013*, *CSP-Minizinc-Obj-2016*, *GRAPHS-2015*, *MAXSAT12-PMS*, *MAXSAT15-PMS-INDU*, *PROTEUS-2014*, *QBF-2011*, *QBF-2014*, *SAT11-HAND*, *SAT11-INDU*, *SAT11-RAND*, *SAT12-ALL*, *SAT12-HAND*, *SAT12-INDU*, *SAT12-RAND*, *SAT15-INDU* and *TSP-LION 2015*. In all of these computational complex AS scenarios, *runtime* is the main performance metric. In each scenario, the dataset comprises algorithms' performances on problem instances, problem instances meta-features run status, and feature values. The standardized datasets make the experiments evaluation results among many scenarios comparable.

In each AS scenario from the ASLib, we split the dataset into 10 folds and apply cross-validation on the 9 folds to find the best hyper-parameter setting for each approach. With the best selected hyper parameters, all approaches are trained again on the whole 9-fold dataset and the fitted models are acquired. These models are used to do the inference on the last fold (test set) to be evaluated.

4.2 Performance of Bi-linear L2R approach

We compare the AS effect of BLR with other benchmark approaches under the four evaluation metrics introduced in the last section. For BLR model, latent dimension K, learning rate η, regularizer λ are the hyper parameters to be tuned during cross validation. Since the optimization target of BLR decomposition is not convex, the trained model is sensitive to the initialization of the entries in the latent matrices. Thus the best initialization state is also determined in the cross-validation phase. To speed up the convergence of the BLR, we use *Stochastic Gradient Descent* instead of *Gradient Descent* as optimization method. Given the vibrated loss value on *Stochastic Gradient Descent*, we tell the convergence of BLR model with at least 5 successive increases on the loss detected during the optimization.

4.2.1 BLR performance with TOP1 candidates selection

First we apply the conventional $TOP1$ candidates selection in the evaluation and observe under what circumstances BLR performs better. In Table 1, AS scenario and evaluation metric combination are listed per row. These are the cases BLR is ranked among the best 3 compared with other benchmark approaches. More specifically, in *CSP-Mininzic-Obj-2016* and *SAT15-INDU* regarding *success rate*, in *PROTEUS-2014* concerning MCP and PAR10, in *TSP-LION2015* in terms of MAP, BLR is ranked as top1. These competitive

[3] https://www.coseal.net/

Table 1 Scenarios and evaluation metric under which Bi-linear L2R is measured as top 3 among all the benchmark approaches, when the candidate set size is set to ONE

Scenario name	Evaluation metric	Rank
CSP-Mininzic-Obj-2016	SUCC	1
GRAPHS-2015	SUCC	3
PROTEUS-2014	MCP	1
PROTEUS-2014	PAR10	1
SAT11-INDU	SUCC	2
SAT12-RAND	MAP	2
SAT15-INDU	SUCC	1
SAT15-INDU	MCP	2
SAT15-INDU	PAR10	3
TSP-LION2015	MAP	1

performances verify that BLR can also be considered as a benchmark approach in some AS scenarios.

4.2.2 Cumulative performance in TOPK expansion

If parallel processing on the candidate algorithms is considered, we can broaden the range of candidates selection to increase the chance of finding the best algorithm without extra time consumption. Thus if the cumulative best performances of approaches decrease drastically at first several predicted positions, it's proper to consider $TOPK$ expansion for the predicted list. We first observe the cumulative best performance along the $TOPK$ position elapse in some scenarios. For *SAT11-HAND*, *PROTEUS-2014* and *MAXSAT12-PMS*, we visualize the cumulative minimum mean runtime for all approaches' predicting lists in Fig. 6. On the left hand side, in scenario *SAT11-HAND*, though BLR (plotted with bold green yellow line) gives the worst recommendation at the top1 position, it reaches the optimal performance as *one hot random forest* does at position 4. Conversely, in scenario *PROTEUS-2014*, as plotted in the middle subplot, BLR finds the algorithm with shortest runtime at position 4 and beats all other approaches, while loses its dominant role gradually from position 3. Approaches like *single best*, *separated xgboost* and *separated random forest* take over the dominant positions from position 3. In scenario *MAXSAT12-PMS*, similar as in scenario *SAT11-HAND*, the recommendation from *blr* reaches best at top position 3, in spite of the worst average run time of its predicted algorithms list at position 1.

4.2.3 BLR performance with expanded candidates selection

The cumulative best performance varies a lot even considering single AS approach, thus the rank of approaches also changes when considering different expansion degrees. For BLR, aside from the conventional $TOP1$ candidates selection criteria, we observe its rankings under $TOP3$ selection. In Table 2, we list the conditions (combinations of scenario and evaluation metric) where BLR is evaluated as competitive (ranked in top 3). BLR can still perform well in some specific scenarios. When being compared with Table 2, only in scenarios *GRAPHS-2015* and *TSP-LION2015*, BLR shows competitive role in both $TOP1$ and $TOP3$. The advancing performances of BLR doesn't hold consistent between $TOP1$ and $TOP3$ candidates selection in most scenarios.

4.3 Accuracy and inference time trade-off

With evaluation metrics SUCC, MAP, MCP and PAR10, the accuracy of AS approaches can be assessed. Nevertheless, shorter inference time is also preferred for an AS approach. As introduced in Sect. 3, A3R is a good metric for measuring the combining effect of accuracy and time. We take this

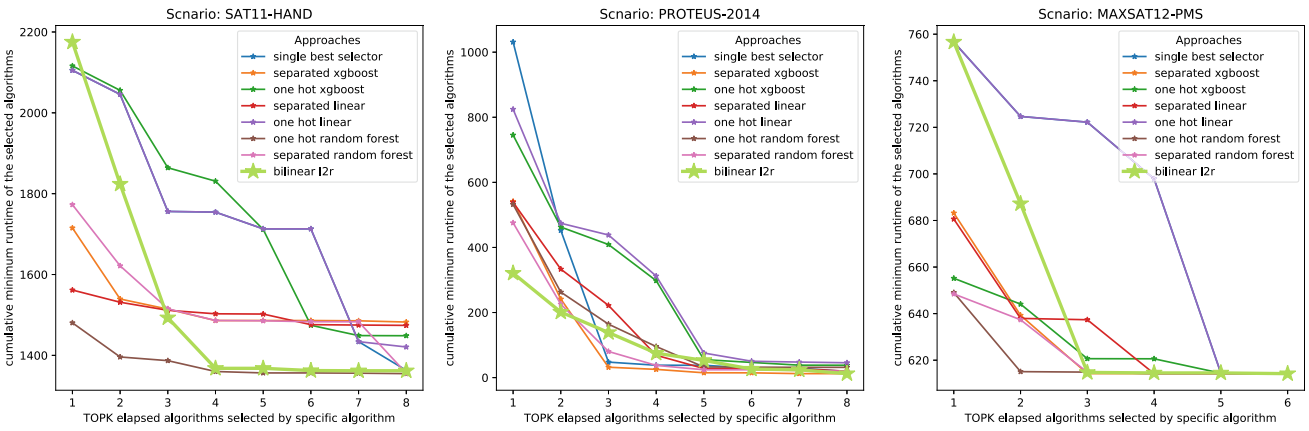

Fig. 6 Cumulative minimum runtime (average across all the predicted problem instances) for scenarios *SAT11-HAND*, *PROTEUS-2014* and *MAXSAT12-PMS*. In every scenario, for all the problem instances in the test set, the algorithms are sorted by their predicted performances. The average cumulative minimum of their actual performance in the sorted list is drawn at each TopK elapsed step

Table 2 Scenarios and evaluation metric under which Bi-linear L2R is measured as top 3 among all the benchmark approaches, when the candidate set size is set to THREE

Scenario name	Evaluation metric	Rank
CPMP-2015	MAP	3
CSP-2010	SUCC	2
CSP-2010	MAP	2
GRAPHS-2015	MCP	3
MAXSAT12-PMS	MCP	3
MAXSAT12-PMS	PAR10	2
PROTEUS-2014	MAP	3
SAT11-HAND	MCP	3
SAT11-HAND	PAR10	3
TSP-LION2015	MAP	1

metric to make the combining effect evaluation for the AS approaches in this experiment. To make the accuracy/time ratio comparable across all scenarios, *one hot random forest regressor* (the approach wins in most scenarios) is taken as reference approach (a_q) in the evaluation equation. It's drawn as the pink bar in the following figures, and the A3R value of this referred algorithm is always 1. All the accuracy metric values are from $TOP3$ candidates setting.

$$A3R(ACC)^{s_i}_{a_p a_q} = \frac{ACC^{s_i}_{a_p}/ACC^{s_i}_{a_q}}{\sqrt[N]{T^{s_i}_{a_p}/T^{s_i}_{a_q}}} \tag{11}$$

As to precision oriented accuracy metrics ((SUCC and MAP), the accuracy ratio is proportional to the metric value of the selected approach. Thus ACC value of a_q (referenced approach) is set as the denominator in the ratio formula $ACC^{s_i}_{a_p}/ACC^{s_i}_{a_q}$ in Eq. (11). Considering that inference time of different AS approaches span in 3 to 4 magnitudes, parameter for root N is set as 30 in the experiment to limit A3R in a reasonable range. As Fig. 7 shows, when evaluating the approaches regarding both MAP and inference time using A3R, BLR (in light blue bar) outperforms all other benchmark approaches. Thus BLR reaches the balance of model complexity and inference simplicity.

$$A3R(TC)^{s_i}_{a_p a_q} = \frac{\sqrt[M]{TC^{s_i}_{a_q}/TC^{s_i}_{a_p}}}{\sqrt[N]{T^{s_i}_{a_p}/T^{s_i}_{a_q}}} \tag{12}$$

For time cost-oriented accuracy metrics (MCP and PAR10), their values are negatively correlated with prediction accuracy. The accuracy ratio $TC^{s_i}_{a_q}/TC^{s_i}_{a_p}$ therefore takes the metric value $TC^{s_i}_{a_p}$ as the denominator. In addition, since the MCP and PAR10 metric value among approaches varies a lot

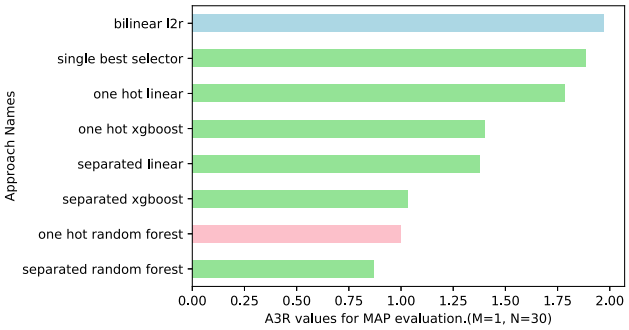

Fig. 7 Approaches' Average A3R score across all the scenarios. A3R in terms of MAP and inference time

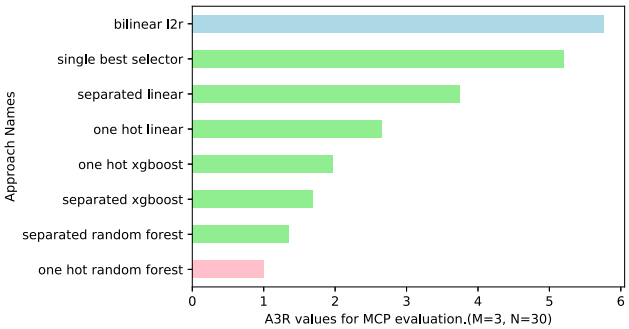

Fig. 8 Approaches' average A3R score across all the scenarios. A3R in terms of MCP and inference time

even concerning magnitude, root parameter M is involved for this accuracy ratio as well to transform the ratio to a readable range. As Fig. 8 shows, with the setting of $M = 3$ and $N = 30$, BLR (represented as light blue bar) again wins other benchmark approaches.

The excellent performance on A3R which cares both precision oriented and time cost-oriented accuracy metrics verifies that BLR can be a good option when the balance between accuracy and inference time needs to be taken into account.

4.4 Benefit of expanding the candidate selection range from Top1 to TopK

As discussed in the former subsections, if we enlarge the algorithm candidates range from $TOP1$ to $TOPK$, we can expect the algorithm selected from the wider spectrum yield better optimal selected algorithm. In this experiment, we tentatively set $K = 3$, and observe the difference on the cumulative evaluation result difference between the conditions $K = 1$ and $K = 3$. For every AS scenario, we list the approach with the largest performance difference caused by $TOP1$ and $TOP3$ selection criteria and thus illustrate the benefit of the $TOPK$ expansion. We choose time cost-oriented metrics MCP and PAR10 to represent the performance difference, considering their straightforward

Table 3 Improvement on MCP evaluation caused by $Top3$ expansion from $Top1$ selection , the decrease percentage over 90% have been highlighted in the bold boxes.

Scenario name	Approach name	MCP@1	MCP@3	MCP@3 - MCP@1	Difference in %
ASP-POTASSCO	One hot random forest	22.06	1.51	−20.55	**−93.16**
BNSL-2016	Separated xgboost	217.94	27.88	−190.06	−87.21
CPMP-2015	One hot random forest	120.11	0.41	−119.69	**−99.66**
CSP-2010	Bilinear l2r	535.44	0.00	−535.44	**−100.−0**
CSP-MZN-2013	Separated random forest	68.78	8.36	−60.42	−87.84
CSP-Minizinc-Obj-2016	Separated xgboost	216.13	95.57	−120.55	−55.78
GRAPHS-2015	One hot xgboost	3,774,156.34	125,048.31	−3,649,108.03	**−96.69**
MAXSAT12-PMS	Separated random forest	34.24	0.04	−34.20	**−99.88**
MAXSAT15-PMS-INDU	One hot random forest	77.08	5.47	−71.61	**−92.9**
PROTEUS-2014	Single best selector	1023.12	39.61	−983.51	**−96.13**
QBF-2011	One hot random forest	72.31	0.00	−72.31	**−100.0**
QBF-2014	Separated xgboost	65.52	9.49	−56.03	−85.52
SAT11-HAND	Bilinear l2r	1206.79	235.06	−971.74	−80.52
SAT11-INDU	Separated random forest	582.31	75.73	−506.58	−86.99
SAT11-RAND	Separated linear	456.10	16.44	−439.66	**−96.4**
SAT12-ALL	Separated xgboost	213.24	59.09	−154.15	−72.29
SAT12-INDU	One hot random forest	91.93	22.33	−69.59	−75.7
SAT12-RAND	One hot random forest	69.13	11.11	−58.02	−83.93
SAT15-INDU	Separated linear	689.23	148.96	−540.28	−78.39
TSP-LION2015	Separated linear	76.95	4.24	−72.71	**−94.5**

cumulative performance decrease along the $TOPK$ positions.

Mis-Classification Penalty (MCP) calculates the time cost difference between the selected algorithm and the actual best algorithm. The lower the MCP value, the better effect the AS approach possesses. Seen from Table 3, $TOP3$ selection criteria leads to the decreasing effect on MCP significantly. We highlight the decrease percentage higher than 90.00% in bold boxes in the table. The decrease percentage ranges from 55.78% to 100%. It demonstrates that enlarging the $TOPK$ candidates selection range can boost finding the algorithm *runtime* closer to the ground truth best.

The evaluation metric PAR10 gives 10 times penalty on the most recommended algorithm which is actually timeout. We list the decrease percentage caused by $TOP3$ candidates expansion in Table 4. This decrease percentage falls in the interval 19.47% to 95.72%. The cases, in that the percentage values are higher than 90.00%, have been highlighted in the bold boxes. This decrease percentage indicates the reduction of the possibility that selected algorithm runs in a timeout.

Expanding the $TOP1$ candidate set to the case $TOP3$, the observation of the significant decrease on the time cost metrics MCP and PAR10 confirms the benefit of the expansion. In AS, under the parallel testing environment, the test on the $TOPK$ candidates stops at the *runtime* of the optimal algorithm in the candidates set. Thus the test time is also saved owing to the expansion. The selection of K depends on the computational power and environmental limit. Though $TOP1$ setting is required in most AS challenges, we would like suggest the expansion of this candidates selection range.

4.5 Discussion

The experiments in this section unveil several interesting points: (1) BLR possesses the chance outperforming other benchmark AS approaches in some scenarios; (2) on evaluation metric A3R, BLR shows the power of balancing prediction accuracy and inference time; (3) $TOPK$ expansion on candidates set brings benefit for finding the optimal algorithm.

5 Conclusion and future work

In this paper, we propose Bi-linear Learning to Rank (BLR) to solve AS problem. BLR is inspired from the collaborative filtering in RS. With the list-wise $TOP1$ probability assumption, it models the uncertainty in the algorithm performance. The learning process of BLR averts the problems like multi-models training and algorithms' one-hot conversion in

Table 4 Improvement on PAR10 evaluation caused by $Top3$ expansion from $Top1$ selection, the decrease percentage over 90% has been highlighted in the bold boxes.

Scenario name	Approach name	PAR10@1	PAR10@3	PAR10@3 PAR10@1	Difference in %
ASP-POTASSCO	Bilinear l2r	1526.22	863.64	−662.58	−43.41
BNSL-2016	Separated xgboost	2089.12	251.60	−1837.52	−87.96
CPMP-2015	Separated random forest	5920.68	346.61	−5574.07	−94.15
CSP-2010	Bilinear l2r	11032.73	6285.47	−4747.26	−43.03
CSP-MZN-2013	Single best selector	9029.44	5538.58	−3490.86	−38.66
CSP-Minizinc-Obj-2016	One hot random forest	2520.38	360.37	−2160.01	−85.7
GRAPHS-2015	Separated linear	73,415,561.38	24,868,328.87	−48,547,232.51	−66.13
MAXSAT12-PMS	Bilinear l2r	7056.63	5611.30	−1445.33	−20.48
MAXSAT15-PMS-INDU	Bilinear l2r	4846.02	3018.59	−1827.43	−37.71
PROTEUS-2014	Single best selector	9851.49	421.98	−9429.51	**−95.72**
QBF-2011	Single best selector	15,699.44	10,371.48	−5327.96	−33.94
QBF-2014	One hot xgboost	3696.26	2536.14	−1160.12	−31.39
SAT11-HAND	Bilinear l2r	29,375.15	19,093.06	−10,282.08	−35.0
SAT11-INDU	One hot xgboost	17,012.79	13,700.37	−3312.42	−19.47
SAT11-RAND	One hot linear	23,445.32	9225.72	−14,219.60	−60.65
SAT12-ALL	Bilinear l2r	6318.53	2676.89	−3641.64	−57.63
SAT12-INDU	One hot xgboost	3698.78	2656.96	−1041.82	−28.17
SAT12-RAND	Separated xgboost	3843.91	3058.80	−785.11	−20.42
SAT15-INDU	Separated linear	3143.63	443.36	−2700.28	−85.9
TSP-LION2015	Separated random forest	255.34	25.21	−230.13	**−90.13**

traditional AS benchmark approaches. Being compared with the benchmark approaches, selection effects of BLR have proven to perform well in some AS scenarios. Considering the balance of the trade-off between the accuracy and inference time in the evaluation, we propose using A3R as the evaluation's protocol. BLR performs especially well on this new trade-off metric A3R. Finally, we affirm the benefit of expanding the selection range of candidate approaches from $TOP1$ to $TOPK$ regarding the cumulative optimal demand of AS evaluation.

Given the work so far, there is much to do for the future. For BLR, since it's a model with non-convex loss definition, the convergence criteria can be adjusted to tune better parameters setting. In the current experimental settings, we only investigate 21 AS scenarios. Extending the experiments to additional scenarios can give a stronger confidence on the experimental results. Though we set K in $TOPK$ expansion as 3, and illustrate the expansion benefit, more thorough study can be done on how to choose K to meet the balance of performance gain and computational power.

Acknowledgements This work is supported in part by the German Federal Ministry of Education and Research (BMBF) under the grant number 01IS16046.

Funding Open Access funding provided by Projekt DEAL.

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

A Appendix

A.1 Terminologies declaration

See Table 5.

Table 5 Terminologies and explanations

Term	Meaning
Scenario	To solve one specific type of problem (e.g., TSP in computational complexity domain), some problem instances (e.g., several complete graphs in a TSP problem), their descriptive meta-features(e.g., number of edges, number of nodes, etc.) and some algorithms (e.g., generic algorithm, the nearest neighbor algorithm) performances on these problem instances comprise a *scenario*
Problem Instance	A concrete instance to be solved in a scenario w.r.t. the problem in this scenario, e.g., a complete graph in TSP problem
Meta-features	Some descriptive features of a specific problem instance, like the number of edges, the number of nodes, etc., in a complete graph for TSP problem
Algorithm	Algorithm or heuristic (e.g., generic algorithm in TSP) which can successfully solve some of the problem instances in the designate scenario
Solver	The alias for algorithm in some problems like SAT
Solution	The solving result settings by an algorithm (or solver) on a problem instance
Approach	The method used to select the potential optimal algorithms candidates set for problem instances in a specific scenario
Predictor	The method that predicts the performance of algorithms on a problem instance
Selector	The method used to select the potential optimal algorithms based on their predicted performances
Algorithm Candidate Set	A set of algorithms selected as the most possible optimal algorithms for a specific problem instance inferred from an approach/selector
Performance	The measurement representing how an algorithm solves a problem instance, e.g., *runtime*
Evaluation Metric	The evaluation criteria to measure the selection effect of an approach in a scenario, e.g., SUCC, MCP

A.2 Notations of Bi-linear L2R

See Table6.

Table 6 Notations of Model Bi-linear Learning to Rank

M	Number of problem instances in the training set
N	Number of algorithms in the training set
L	Number of meta-features calculated for each problem instance
K	Dimension number of the latent factor
$\mathbf{S}_{M\times N}$	The performance matrix for N algorithms on M problem instances
$s_{m,n}$	The performance value of algorithm n on problem instance m
$\hat{\mathbf{S}}_{M\times N}$	The predicted performance matrix for N algorithms on M problem instances
$\hat{s}_{m,n}$	The predicted performance value of algorithm n on problem instance m
$\mathbf{X}_{M\times L}$	Values of L meta-features on M problem instances
$x_{m,l}$	The l_{th} meta-feature of problem instance m
$\mathbf{W}_{L\times K}$	Bi-linear weight matrix which maps from L problem meta-features to k-dimensional latent feature space
$w_{l,k}$	The mapping factor for the l_{th} meta-feature on the k_{th} latent factor
$\mathbf{U}_{M\times K}$	Matrix of K-dimensional latent vector for M problem instances
$u_{m,k}$	The k_{th} latent factor of problem instance m
$\mathbf{V}_{N\times K}$	Matrix of K-dimensional latent vector for N algorithms
$v_{n,k}$	The k_{th} latent factor of algorithm n
$\mathbf{R}_{M\times N}$	Matrix of performance ratings of N algorithms on M problem instances (A converted representation of $\mathbf{S}_{M\times N}$, which assign better performed algorithms a higher value)
$r_{m,n}$	The converted rating value of algorithm n on problem instance m
$\hat{\mathbf{R}}_{M\times N}$	Matrix of estimated performance ratings of N algorithms on M problem instances
$\hat{r}_{m,n}$	The estimated converted rating value of algorithm n on problem instance m
$P_{\mathbf{r}_m}(r_{m,n})$	Given the actual performance rating vector $\mathbf{r_m}$, the probability that algorithm n is ranked at top 1 regarding the m_{th} problem instance
$P_{\hat{\mathbf{r}}_m}(\hat{r}_{m,n})$	Given the estimated rating vector $\hat{\mathbf{r}}_m$, the probability that algorithm n is ranked at top 1 regarding the m_{th} problem instance

A.3 Gradient and updating rules in Bi-linear L2R

In BLR model, for approximating the weighting matrix $\mathbf{W}$ and latent matrix $\mathbf{V}$ to minimize the loss function defined in the Sect. 2, we calculate the gradient for loss function and use the updating rule as described in the following subsections.

A.3.1 Gradient calculation

Having known the loss function L, to use gradient descent as the optimizer, the gradient concerning meta-features mapping weight matrix $\mathbf{W}$ and algorithm latent vectors matrix $\mathbf{V}$ should be provided accordingly. Since the loss function is defined layer by layer through model function, converter function, top-one probability function and cross entropy function, we use chain rule to calculate the gradient correspondingly. For L, its partial differential over $w_{l,k}$ and $v_{n,k}$ can be factorized in the similar way as Eqs. (13) and (14) separately.

$$\frac{\partial L}{\partial w_{l,k}} = \sum_{m=1}^{M}\sum_{n=1}^{N} \frac{\partial L_{m,n}}{\partial P_{\hat{\mathbf{r}}_m}(\hat{r}_{m,n})} \frac{\partial P_{\hat{\mathbf{r}}_m}(\hat{r}_{m,n})}{\partial \hat{r}_{m,n}} \frac{\partial \hat{r}_{m,n}}{\partial \hat{s}_{m,n}} \frac{\hat{s}_{m,n}}{\partial w_{l,k}} + \lambda w_{l,k} \tag{13}$$

$$\frac{\partial L}{\partial v_{n,k}} = \sum_{m=1}^{M}\sum_{n=1}^{N} \frac{\partial L_{m,n}}{\partial P_{\hat{\mathbf{r}}_m}(\hat{r}_{m,n})} \frac{\partial P_{\hat{\mathbf{r}}_m}(\hat{r}_{m,n})}{\partial \hat{r}_{m,n}} \frac{\partial \hat{r}_{m,n}}{\partial \hat{s}_{m,n}} \frac{\hat{s}_{m,n}}{\partial v_{n,k}} + \lambda v_{n,k} \tag{14}$$

For each $L_{m,n}$, the intermediate calculation steps for deviation according to chain rule can be derived as following for each $L_{m,n}$:

$$\begin{aligned}
\frac{\partial L_{m,n}}{\partial P_{\hat{\mathbf{r}}_m}(\hat{r}_{m,n})} &= -P_{\mathbf{r}_m}(r_{m,n}) \frac{1}{P_{\hat{\mathbf{r}}_m}(\hat{r}_{m,n})} \\
\frac{\partial P_{\hat{\mathbf{r}}_m}(\hat{r}_{m,n})}{\hat{r}_{m,n}} &= \frac{\partial}{\partial \hat{r}_{m,n}} \frac{\exp \hat{r}_{m,n}}{\sum_{n=1}^{N} \exp \hat{r}_{m,n}} \\
&= \frac{\exp(\hat{r}_{m,n})}{\sum_{n=1}^{N} \exp(\hat{r}_{m,n})} - \frac{\exp(\hat{r}_{m,n})^2}{(\sum_{n=1}^{N} \exp(\hat{r}_{m,n}))^2} \\
\frac{\partial \hat{r}_{m,n}}{\partial \hat{s}_{m,n}} &= -1 \qquad (15) \\
\frac{\partial \hat{s}_{m,n}}{\partial w_{l,k}} &= x_{m,l} v_{n,k} \qquad (16) \\
\frac{\partial \hat{s}_{m,n}}{\partial v_{n,k}} &= \sum_{l=1}^{L} x_{m,l} w_{l,k} \qquad (17)
\end{aligned}$$

For the last step, which returns the partial differential $\hat{s}_{m,n}$ over $w_{l,k}$ and $v_{n,k}$, we can broadcast it in the vectorized way like Eqs. (18) and (19):

$$\frac{\partial \hat{s}_{m,n}}{\partial \mathbf{W}} = \mathbf{x}_m \otimes \mathbf{v}_n \tag{18}$$

$$\frac{\partial \hat{s}_{m,n}}{\partial \mathbf{v}_n} = \mathbf{x}_m \times \mathbf{W} \tag{19}$$

A.3.2 Updating rule

Having known the partial differential of the chain rule, we can update the weight matrix $\mathbf{W}$ and algorithm latent matrix $\mathbf{V}$ by the following updating rule Eqs. (20) and (21) , where η is the learning rate.

$$\mathbf{W}^t = \mathbf{W}^{t-1} - \eta \frac{\partial L}{\partial \mathbf{W}^{t-1}} \tag{20}$$

$$\mathbf{v}_n = \mathbf{v}_n^{t-1} - \eta \frac{\partial L}{\partial \mathbf{v}_n^{t-1}} \tag{21}$$

Since the loss is list-wise, which means for each problem instance, there is a loss schema based on the corresponding top one probability. If we would like to update the weights in a stochastic way, the updating unit should be a list based on problem instance m, rather than each rating point. Therefore, the stochastic updating rule is like Eqs. (22) and (23):

$$\begin{aligned}
\mathbf{W}^t &= \mathbf{W}^{t-1} - \eta \frac{\partial L_m}{\partial \mathbf{W}^{t-1}} \\
&= \mathbf{W}^{t-1} - \sum_{n=1}^{N} \left(\eta \frac{\partial L_{m,n}}{\partial \mathbf{W}^{t-1}} \right) \qquad (22)
\end{aligned}$$

$$\mathbf{v}_n = \mathbf{v}_n^{t-1} - \eta \frac{\partial L_m}{\partial \mathbf{v}_n^{t-1}} \tag{23}$$

References

1. Rice, J.R.: The Algorithm Selection Problem. Volume 15 of Advances in Computers, pp. 65–118. Elsevier, Amsterdam (1976)
2. Kerschke, P., Hoos, H.H., Neumann, F., Trautmann, H.: Automated algorithm selection: survey and perspectives. Evolut. Comput. **27**, 3–45 (2019)
3. Brazdil, P.B., Soares, C.: A comparison of ranking methods for classification algorithm selection. In: Machine Learning: ECML, pp. 63–75, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg (2000)
4. Abdulrahman, S.M., Brazdil, P., van Rijn, J.N., Vanschoren, J.: Speeding up algorithm selection using average ranking and active testing by introducing runtime. Mach. Learn. **107**, 79–108 (2018)
5. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Satzilla: Portfolio-based algorithm selection for SAT. Computing Research Repository (CoRR), arXiv:1111.2249 (2011)
6. Lin, X., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Satzilla: Portfolio-based algorithm selection for sat. J. Artif. Intell. Res. **32**, 565–606 (2008)
7. Gonard, F., Schoenauer, M., Sebag, M.: Asap.v2 and asap.v3: sequential optimization of an algorithm selector and a scheduler. In: Lindauer, M., van Rijn, J.N., Kotthoff, L. (eds.), Proceedings of the

Open Algorithm Selection Challenge, volume 79 of Proceedings of Machine Learning Research, pages 8–11, Brussels, Belgium, (2017). PMLR
8. Doan, T., Kalita, J.: Algorithm selection using performance and run time behavior. In: Dichev, C., Agre, G. (eds) International Conference on Artificial Intelligence: Methodology, Systems, and Applications. AIMSA 2016:Artificial Intelligence: Methodology, Systems, and Applications, pages 3–13, Cham, Switzerland, (2016). Springer International Publishing
9. Misir, M., Sebag, M.: Algorithm selection as a collaborative filtering problem. Research report, December (2013)
10. Liu, J.-H., Zhou, T., Zhang, Z.-K., Yang, Z., Liu, C., Li, W.-M.: Promoting cold-start items in recommender systems. PLoS ONE **9**, 1–13 (2014)
11. Kotthoff, L.: LLAMA: leveraging learning to automatically manage algorithms. Computing Research Repository (CoRR) arXiv:1306.1031 (2013)
12. Kadioglu, S., Malitsky, Y., Sellmann, M., Tierney, K.: Isac—instance-specific algorithm configuration. In: Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence, pp. 751–756, Amsterdam, Netherland (2010). IOS Press
13. Kadioglu, S., Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M.: Algorithm selection and scheduling. In: Lee, J. (ed), Principles and Practice of Constraint Programming—CP 2011, pages 454–469, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg
14. Lindauer, M.T., Hoos, H.H., Hutter, F., Schaub, T.: Autofolio: an automatically configured algorithm selector. J. Artif. Intell. Res. **53**, 745–778 (2015)
15. Lindauer, M., Hutter, F., Hoos, H.H., Schaub, T.: Autofolio: an automatically configured algorithm selector (extended abstract). In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017, pp. 5025–5029 (2017)
16. Kotthoff, L., Thornton, C., Hoos, H.H., Hutter, F., Leyton-Brown, K.K.: Auto-weka 2.0: automatic model selection and hyperparameter optimization in weka. J. Mach. Learn. Res. (JMLR) **18**, 826–830 (2017)
17. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.), Advances in Neural Information Processing Systems 28, pages 2962–2970. Curran Associates, Inc., (2015)
18. Lin, B.L., Sun, X., Salous, S.: Solving travelling salesman problem with an improved hybrid genetic algorithm. J. Comput. Commun. **4**, 98–106 (2016)
19. Wang, S., Hu, L., Cao, L.: Perceiving the next choice with comprehensive transaction embeddings for online recommendation. In: Ceci, M., Hollmén, J., Todorovski, L., Vens, C., Džeroski, S. (eds.) Machine Learning and Knowledge Discovery in Databases, pp. 285–302, Cham, Switzerland (2017). Springer International Publishing
20. Wang, S., Hu, L., Wang, Y., Sheng, Q.Z, Orgun, M., Cao, L.: Modeling multi-purpose sessions for next-item recommendations via mixture-channel purpose routing networks. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, pp. 3771–3777. International Joint Conferences on Artificial Intelligence Organization, July (2019)
21. Stern, D.H., Herbrich, R., Graepel, T.: Matchbox: Large scale online bayesian recommendations. In: Proceedings of the 18th International Conference on World Wide Web, WWW '09, pages 111–120, New York, NY, USA, (2009). ACM
22. Stern, D., Herbrich, R., Graepel, T., Samulowitz, H., Pulina, L., Tacchella, A.: Collaborative expert portfolio management. In: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI'10, pages 179–184. AAAI Press, (2010)
23. Misir, M., Sebag, M.: Alors: an algorithm recommender system. Artif. Intell. **244**, 291–314 (2017)
24. Yang, C., Akimoto, Y., Kim, D.W., Udell, M.: Oboe: collaborative filtering for automl model selection. In: Proceedings of the Twenty-Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1173–1183. Association for Computing Machinery, (2019)
25. Wang, X., Bendersky, M., Metzler, D., Najork, M.: Learning to rank with selection bias in personal search. In: Proc. of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 115–124 (2016)
26. Joachims, T., Swaminathan, A., Schnabel, T.: Unbiased learning-to-rank with biased feedback. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM '17, pages 781–789, New York, NY, USA, (2017). ACM
27. Abdollahpouri, H., Burke, R., Mobasher, B.: Controlling popularity bias in learning-to-rank recommendation. In: Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys '17, pp. 42–46, New York, NY, USA, 2017. ACM
28. Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., Li, H.: Learning to rank: From pairwise approach to listwise approach. In: Proceedings of the 24th International Conference on Machine Learning, ICML '07, pages 129–136, New York, NY, USA, (2007). ACM
29. Oentaryo, R.J., Handoko, S.D., Lau, H.C.: Algorithm selection via ranking. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI'15, pp. 1826–1832. AAAI Press, (2015)
30. Luo, M., Li, C.-M., Xiao, F., Manya, F., Zhipeng, L.: An effective learnt clause minimization approach for CDCL sat solvers. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, pp. 703–711, (2017)
31. Lee, J.N.-Z., Wang, Y.S., Jiang, J.H.R.: Solving stochastic boolean satisfiability under random-exist quantification. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, pp. 688–694, (2017)
32. Mordido, A., Caleiro, C., Casal, F.: Classical generalized probabilistic satisfiability. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, pp. 908–914 (2017)
33. Bacchus, F., Hyttinen, A., Matti, S.: Paul: Reduced cost fixing in maxsat. In: Principles and Practice of Constraint Programming—23rd International Conference, CP 2017, Melbourne, VIC, Australia, pp. 641–651 (2017)
34. Gomes, C.P., Selman, B.: Algorithm portfolios. Artif. Intell. **126**, 43–62 (2001)
35. Gomes, C.P., Selman, B.: Algorithm portfolio design: theory vs. practice. Computing Research Repository (CoRR), arXiv:1302.1541 (2013)
36. Leyton-Brown, K., Nudelman, E., Andrew, G., McFadden, J., Shoham, Y.: A portfolio approach to algorithm select. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI'03, pp. 1542–1543, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc (2003)
37. Kotthoff, L., Hurley, B., O'Sullivan, B.: The ICON challenge on algorithm selection. AI Mag. **38**, 91–93 (2017)
38. Bischl, B., Kerschke, P., Kotthoff, L., Lindauer, M.T., Malitsky, Y., Fréchette, A., Hoos, H.H., Hutter, F., Leyton-Brown, K., Tierney, K., Vanschoren, J.: Aslib: A benchmark library for algorithm selection. Computing Research Repository (CoRR), arXiv:1506.02465 (2015)
39. Abdulrahman, S., Brazdil, P.: Measures for combining accuracy and time for meta-learning. CEUR Workshop Proc. **1201**, 49–50 (2014)

40. Abdulrahman, S.M., Pavel, B., Wan, M.N., Wan, Z., Alhassan, A.: Simplifying the algorithm selection using reduction of ranking of classification algorithms. In: Proceedings of the 2019 8th International Conference on Software and Computer Application, pp. 140–148. ACM (2019)