Alors: An algorithm recommender system

Mustafa Mısır, Michèle Sebag

ALORS: an Algorithm Recommender System

Preprint of *Artificial Intelligence*, 244:291-314, 2017, ISSN 0004-3702

Mustafa Mısır[1]*, Michèle Sebag[2]

[1] *LARC, School of Information Systems, Singapore Management University, 680 Stamford Road, 178902, Singapore*

[2] *LRI, CNRS–INRIA–Université Paris-Sud, Université Paris-Saclay Shannon Bdg 660, 91405 Orsay Cedex, France*

Abstract

Algorithm selection (AS), selecting the algorithm best suited for a particular problem instance, is acknowledged to be a key issue to make the best out of algorithm portfolios. While most AS approaches proceed by learning the performance model, this paper presents a collaborative filtering approach to AS.

Collaborative filtering, popularized by the Netflix challenge, aims to recommend the items that a user will most probably like, based on the previous items she liked, and the items that have been liked by other users. As first noted by Stern et al. (2010), algorithm selection can be formalized as a collaborative filtering problem, by considering that a problem instance "likes better" the algorithms that achieve better performance on this particular instance.

Two merits of collaborative filtering (CF) compared to the mainstream algorithm selection (AS) approaches are the following. Firstly, AS requires extensive and computationally expensive experiments to learn the performance model, with all algorithms launched on all problem instances, whereas CF can exploit a sparse matrix, with a few algorithms launched on each problem instance. Secondly, AS learns a performance model as a function of the initial instance representation, whereas CF builds latent factors to describe algorithms and instances, and uses the associated latent metrics to recommend algorithms for a specific problem instance. A main contribution of the proposed algorithm recommender ALORS system is to handle the *cold start* problem − emitting recommendations for a new problem instance − through the non-linear modelling of the latent factors based on the initial instance representation, extending the linear approach proposed by Stern et al. (2010).

The merits and generality of ALORS are empirically demonstrated on the ASLib (Bischl et al., 2015) and OpenML (van Rijn et al., 2013) benchmark datasets.

*Work was achieved while at LRI, funded by an ERCIM grant

1 Introduction

In many fields, such as propositional satisfiability (SAT), constraint satisfaction (CSP), machine learning (ML) or data mining (DM), a variety of algorithms and heuristics have been developed to address the specifics of problem instances. In order to get peak performance on any particular problem instance, one must select the algorithm and hyper-parameter setting best suited to this problem instance. This selection is known under various names depending on the field: algorithm selection and configuration in SAT and CSP (Rice, 1976; Leyton-Brown et al., 2003; Epstein et al., 2006; Samulowitz and Memisevic, 2007; Xu et al., 2008),(O´Mahony et al., 2008; Hutter et al., 2009; Pulina and Tacchella, 2010; Kadioglu et al., 2010; Xu et al., 2012a; Malitsky et al., 2013; Malitsky and O´Sullivan, 2014; Lindauer et al., 2015), meta-learning and hyper-parameter tuning in ML (Brazdil and Soares, 2000; Bergstra et al., 2011; Thornton et al., 2013; Bardenet et al., 2013; Sun and Pfahringer, 2013), or meta-mining in DM (Nguyen et al., 2014).

This paper focuses on algorithm selection in the SAT, CSP and ML domains. In this domain, extensive algorithm portfolios have been proposed following Gomes and Selman (2001) and significant advances regarding algorithm selection have been made in the last decade (see Kotthoff (2014) for a survey; section 2). Most algorithm selection approaches proceed by estimating the performance model, applying supervised machine learning algorithms onto datasets that record the performance of each algorithm in the portfolio on each problem instance (described by a vector of feature values) in a benchmark suite.

Another approach, based on collaborative filtering, is investigated in this paper. Collaborative filtering (Su and Khoshgoftaar (2009); Bobadilla et al. (2013); section 3), popularized by the Netflix challenge (Bennett and Lanning, 2007), exploits the user data (the items she liked/disliked in the past) and the community data (recording which user liked which items), to recommend new items that the user is most likely to like. As first noted by Stern et al. (2010), algorithm selection can be viewed as a collaborative filtering problem, by considering that a problem instance "likes better" the algorithms that achieve better performance on this problem instance.

Collaborative filtering (CF) offers two main advantages compared to supervised ML-based algorithm selection (Brazdil et al., 2008; Xu et al., 2012a). Firstly, it does not require all portfolio algorithms to be run on all problem instances, which entails significant computational savings. Secondly, it enables to independently analyze: i) the quality and representativity of the benchmark suite with regard to the algorithm portfolio; ii) the quality of the features used to describe the problem instances in the benchmark suite. CF tackles the first issue by extracting a new description of the problem instances and the portfolio algorithm, referred to as *latent representation*, and checking whether this representation provides an adequate performance model on the benchmark suite. The second issue is tackled by studying whether the initial features can be used to estimate the latent features, and thus provide a performance model for new problem instances. When it is the case, CF successfully handles the so-called

cold-start task (Ahn, 2008; Park and Chu, 2009; Liu et al., 2014): emitting recommendations for a brand new user or selecting algorithms for a brand new problem instance. A main novelty of the presented ALORS algorithm recommender system is to address the cold-start problem by learning *non-linear latent representations*, extending the linear approaches proposed by Stern et al. (2009, 2010).

The main contributions of the present paper are the following:

- The publicly available collaborative algorithm recommender ALORS[1] can accommodate sparse data (as opposed to the extensive experiment campaigns required by supervised ML-based algorithm selection);
- The extensive experimental validation of ALORS on the ASlib benchmark (Bischl et al., 2015) empirically shows the merits of ALORS, with an overall runtime less than one minute on a standard PC. Experiments on artificial data show the robustness of ALORS compared to the MATCHBOX system (Stern et al., 2009, 2010), with respect to the initial representation of the problem instances;
- The latent representation extracted by ALORS supports the visual and quantitative analysis of the initial representation; this is particularly useful when the initial representation is insufficient, as will be illustrated on the OpenML dataset (van Rijn et al., 2013).

The paper is organized as follows. Section 2 discusses the state of the art in algorithm selection. The basics of collaborative filtering are introduced in Section 3. Section 4 gives an overview of the proposed ALORS system, detailing how it tackles the cold-start issue. Section 5 discusses the goal of experiments and introduces the experimental setting used to conduct the experimental validation of ALORS. Section 6 reports on the experimental comparative validation of the approach and the general lessons learned. Section 7 concludes the paper with a discussion and some perspectives for further work.

2 Related work

In several fields related to combinatorial optimization, and specifically in constraint satisfaction and machine learning, it was early recognized that there exists no such thing as a universal algorithm dominating all other algorithms on all problem instances (Wolpert and Macready, 1997). Abandoning the search for universal algorithms, the scientific community therefore aimed at comprehensive algorithm portfolios, such that a problem instance would be properly handled by at least one algorithm in the portfolio. Algorithm portfolios thus give rise to the algorithm selection issue, aimed at selecting the algorithm best suited to a particular problem instance.

[1] available at `http://www.lri.fr/~sebag/Alors`

2.1 Formalization

Algorithm selection was first formalized by Rice (1976) to our best knowledge, based on i) a problem space $\mathcal{P}$; ii) an algorithm space $\mathcal{A}$; iii) a mapping from $\mathcal{P} \times \mathcal{A}$ onto $\mathbb{R}$, referred to as *performance model*, estimating the performance of any algorithm on any problem instance. The performance model thus naturally supports algorithm selection, by selecting the algorithm with best estimated performance on the current problem instance[2].

2.2 Learning a performance model

The advances of algorithm selection in SAT and CP in the last decade rely on two facts (Kotthoff, 2014). Firstly, extensive sets of problem instances have been gathered to benchmark the algorithms. Secondly and most importantly, a comprehensive set of features with moderate computational cost was proposed to describe problem instances and the algorithm state at any point in time, respectively referred to as static and dynamic features (Kadioglu et al., 2011; Xu et al., 2012b).

The set of problem instances, their descriptive features and the associated algorithm performances together define a supervised machine learning problem, where each benchmark problem instance is represented as a d-dimensional feature vector $\mathbf{x}$ ($\mathbf{x} \in \mathbb{R}^d$), labeled with the actual performance $F(\mathbf{x}, a) \in \mathbb{R}$ of any given algorithm a in the algorithm portfolio $\mathcal{A}$ on this problem instance. Note that the algorithm performance is domain- and application-specific (e.g., time-to-solution for satisfiable instances or number of solutions found in a given amount of time). From the training set

$$\mathcal{E} = \{(\mathbf{x}_i, F(\mathbf{x}_i, a)), \mathbf{x}_i \in \mathbb{R}^d, a \in \mathcal{A}, F(\mathbf{x}_i, a) \in \mathbb{R}, i = 1 \dots n\}$$

supervised machine learning algorithms, specifically regression algorithms[3], derive an estimate of the computational cost of any algorithm a on any problem

[2] Note that if the algorithm space also involves the algorithm hyper-parameters, the above setting also encompasses the search for the algorithm hyper-parameter values that yield an optimal performance on the problem instance, referred to as *Algorithm Configuration* (Hutter et al., 2011; Thornton et al., 2013). As noted by Thornton et al. (2013), algorithm selection and algorithm configuration can be *reformulated as a single combined hierarchical hyper-parameter optimization problem*, where the choice of the first hyper-parameter – corresponding to the algorithm index – governs the other parameters. However, algorithm selection and algorithm configuration are usually tackled using quite different approaches in the literature (with the exception of (Lindauer et al., 2015)) and this paper restricts its scope to algorithm selection.

[3] An alternative is to formulate algorithm selection as a classification problem, where each problem instance is labeled with the best algorithm for this instance. The classification formulation is however more brittle (as there might be ties, with several algorithms reaching the best performance for a problem instance), and less scalable w.r.t. the number of algorithms in the portfolio. Hybrid approaches have been developed with promising results (Kotthoff, 2012), using stacking ML, and exploiting the predicted runtimes of the portfolio algorithms as features for the classification problem.
Another possibility is to learn one classification model for each pair of algorithms (a, b), predicting whether a will outperform b on a problem instance. These pairwise models are then aggregated to support algorithm selection.

instance described by its feature vector $\mathbf{x}$,

$$\widehat{F} : \mathbf{x} \in \mathbb{R}^d, a \in \mathcal{A} \mapsto \widehat{F}(\mathbf{x}, a) \in \mathbb{R}$$

Algorithm selection proceeds by selecting the algorithm with optimal estimated performance on the current problem instance $\mathbf{x}$:

$$\text{Select } a^*(\mathbf{x}) = \underset{a \in \mathcal{A}}{\text{argmin}}\ \widehat{F}(\mathbf{x}, a) \tag{1}$$

Among the many algorithm selection approaches designed for SAT and CSP (Epstein et al., 2006; Samulowitz and Memisevic, 2007; Xu et al., 2008; O´Mahony et al., 2008; Hutter et al., 2009; Kotthoff, 2012), SATzilla is considered to be the most prominent one after Kotthoff (2014), as it has dominated the SAT competition for years, continuously extending its approach and improving its performances (Xu et al., 2008, 2012a).

SATzilla is a multi-stage process. It first runs pre-solvers for a short amount of time, aimed at solving easy instances. For other instances, their representation (the values of the descriptive features) needs to be computed to achieve algorithm selection. However, as it might require a long time to compute these descriptive features, a dedicated model is learned to predict the time required to compute the descriptive feature values. In cases where the predicted computational time is greater than a threshold (suggesting that the problem instance is a hard one), a backup solver is launched. Otherwise, the description of the problem instance is exploited by the performance model to select the best algorithm in the portfolio (Eq. 1). The performance model itself was learned using ridge regression (Xu et al., 2008); currently random forests are used to perform pairwise selection (Xu et al., 2012c). Note that the problem instances are commonly partitioned into categories (e.g. random, crafted and industrial instances), with one specific performance model learned for each category.

Other approaches such as CPHydra (O´Mahony et al., 2008), ISAC (Kadioglu et al., 2010) or ArgoSmArT (Nikoli et al., 2013) rely on the (semi) metric on the problem instance space defined from the descriptive features. In CPHydra a case-based approach is used, akin a nearest neighbor approach, to determine the solver schedule expectedly most appropriate to a new problem instance. In ISAC, the problem instances are first clustered; the cluster associated with the new problem instance $\mathbf{x}$ is determined, and if sufficiently close from the cluster center, $\mathbf{x}$ is processed using the algorithm that achieves the best average performance over this cluster, called *single best* algorithm for the cluster; otherwise, it is processed using a best default algorithm (single best for the whole dataset). In ArgoSmArT, the nearest neighbors of the current instance are computed and the selected algorithm is the one with best performance on the set of nearest neighbors.

Various AS approaches use various supervised machine learning algorithms, ranging from linear regression to random forests, with the exception of (Gagliolo and Schmidhuber, 2011), who use reinforcement learning. The specifics of the AS problem are handled by means of the loss function, used in supervised ML to train the performance model. Recent AS advances exploit specific

loss functions. A natural loss function in SAT is the extra computational cost incurred in case of AS mistake. Cost-sensitive loss functions are involved in the last version of SATzilla and in the *Cost Sensitive Hierarchical Clustering* (CSCH), which won the open track of the 2013 SAT competition (Malitsky et al., 2013). Another possibility, first considered in an early work (Mısır and Sebag, 2013), is to consider order-based loss functions, where the loss measures how many pairs of algorithms are swapped for a given problem instance (such that $\left(\widehat{F}(\mathbf{x},a) < \widehat{F}(\mathbf{x},b)\right) \neq (F(\mathbf{x},a) < F(\mathbf{x},b))$). A rank-based loss function is used in RAS (Oentaryo et al., 2015) to learn a probabilistic polynomial model with a stochastic gradient descent method. This model, mapping the problem instance space onto the algorithm ranks, is used as performance model.

2.3 Discussion

It is commonplace to say that the quality of a learned model depends on the quality of the descriptive features. While the quality of descriptive features can be assessed *a posteriori* from the quality of the performance model, it is more difficult to assess them *a priori*. Intuitively, good descriptive features should induce a topology on the space of problem instances, and/or on the space of algorithms, such that similar problem instances should be recommended the same algorithm. How to build this topology and these similarity functions is at the core of recommender systems and collaborative filtering, described below.

3 Collaborative Filtering

Recommender systems help the user to face the overwhelming diversity of the items available online (e.g., items for Amazon or movies for Netflix), and retrieve the items she will possibly like. Referring the interested reader to Su and Khoshgoftaar (2009); Agarwal et al. (2013) for a comprehensive survey, let us describe recommender systems, focusing on the collaborative filtering (CF) approach popularized by the Netflix challenge (Bennett and Lanning, 2007), before introducing some algorithm selection approaches based on CF.

3.1 Formalization

Formally, CF exploits the matrix $\mathcal{M}$ which stores the purchases and feedback of the whole community of users: what they bought and what they liked/disliked. Let n (respectively m) denote the number of users (resp. items). The (n, m) matrix $\mathcal{M}$ is a high-dimensional matrix – typically the Netflix challenge involved circa 480,000 users and 18,000 movies; and the set E of the user-item pairs for which $\mathcal{M}_{i,j}$ is known is less than 1% of all user-item pairs, as a user actually sees very few movies on average.

Memory-based approaches rely on metrics or similarity functions, ranging from cosine similarity and Pearson correlation to more ad hoc measures (e.g., mixing *proximity, impact* and *popularity* (Ahn, 2008)) on the user and item

spaces. These metrics or similarities support the recommendation of items most similar to those items the target user liked in the past, or the recommendation of items that users similar to the target user liked in the past.

Model-based approaches learn a low-rank approximation of the CF matrix, taking inspiration from Singular Value Decomposition (SVD) (Strang, 1980). SVD proceeds by decomposing matrix $\mathcal{M}$ as

$$\mathcal{M} = \mathrm{U}\Sigma\ \mathrm{V}^t$$

with U an (n, n) matrix, V an (m, m) matrix and V^t its transpose, and Σ a matrix with same dimensions as $\mathcal{M}$, with non-negative decreasing coefficients on its first diagonal. A low-rank approximation of $\mathcal{M}$ is obtained by cancelling out all coefficients in Σ except the top-k ones. Finally, incorporating the eigenvalues in U and V, the CF matrix $\mathcal{M}$ is expressed as:

$$\mathcal{M} \approx_E \mathrm{U}_k.\ \mathrm{V}_k^t$$

where U_k and V_k, respectively (n, k) and (m, k) matrices, are solutions of the optimization problem:

$$\mathrm{U}_k,\ \mathrm{V}_k = \underset{U,V}{\operatorname{argmin}} \left\{ \mathcal{L}(\mathcal{M},\ \mathrm{U}.\ \mathrm{V}^t) + \frac{\lambda}{2}(tr(\ \mathrm{U}.\ \mathrm{U}^t) + tr(\ \mathrm{V}.\ \mathrm{V}^t)) \right\} \qquad (2)$$

where i) $\mathcal{L}$ is the loss function enforcing the approximation of $\mathcal{M}$ (see below); ii) regularization terms $tr(\ \mathrm{U}.\ \mathrm{U}^t)$ and $tr(\ \mathrm{V}.\ \mathrm{V}^t)$ are meant to prevent overfitting given the sparsity of $\mathcal{M}$; and iii) the rank k and the regularization weight λ are determined by cross-validation. For notational simplicity, the k index is omitted in the following. The loss function measures how well $\mathcal{M}_{i,j}$ is approximated by the scalar product of the i-th row in U (noted U_i) and the j-th row in V (noted V_j), for all user-item pairs (i, j) such that $\mathcal{M}_{i,j}$ is known.

Matrices U and V are referred to as the *latent representation* of users and items respectively. For instance in the Netflix challenge, each one of the k coordinates is interpreted as representing a "pure" movie type, e.g., action, romance, comedy, fantasy or gore categories. Along this interpretation, U_i represents the i-th user as a weighted combination of amateur of action, romance, comedy, etc., movies. Likewise, V_j represents the j-th movie as a weighted combination of action, romance, etc. movie.

Under the low rank assumption, the model-based approach enables to reconstruct the whole matrix $\mathcal{M}$, approximating $\mathcal{M}_{i,j}$ by the scalar product $\langle\ \mathrm{U}_i,\ \mathrm{V}_j \rangle$ for all pairs (i, j). It thereby supports recommendation by selecting for each known i-th user, the known j-th item maximizing $\mathcal{M}_{i,j}$. However, this does not work when considering a new user or a new item, for which no like/dislike is available in the CF $\mathcal{M}$ matrix, as the associated latent representation is unavailable. We shall return to this issue, referred to as the *cold start* problem (Schein et al., 2002; Gunawardana and Meek, 2008) in section 3.3.

Model-based CF tackles three interdependent issues: i) setting the CF model space; ii) defining the optimization criterion; iii) solving the optimization problem.

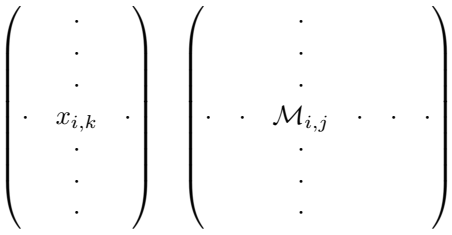


Figure 1: Collaborative filtering matrix $\mathcal{M}$, where $\mathcal{M}_{i,j}$ gives the rating of the j-th item by the i-th user, and $x_{i,k}$ denotes the k-th context information on the i-th user.

Regarding the CF model space, probabilistic models are often preferred as they are more robust w.r.t. preference noise on the one hand, and because the structure of the probabilistic model enables to take advantage of prior knowledge about the problem domain (Salakhutdinov and Mnih, 2008; Zhou et al., 2010). As depicted in Fig. 3.1, the CF matrix is often accompanied with some user context (e.g., age and gender); this context is exploited to build richer CF models, and to provide user-dependent recommendations in the cold-start phase.

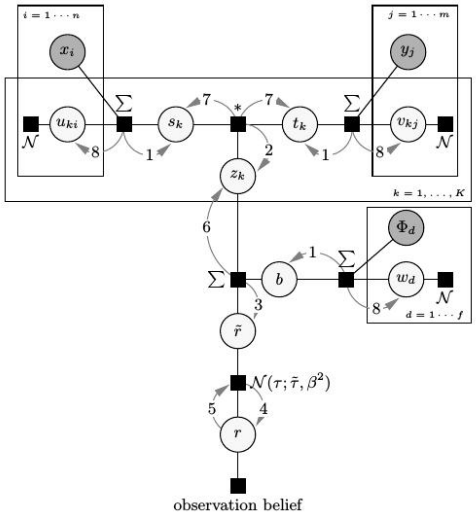
The loss criterion (Eq. 2) can be adapted to reflect the fact that ranks might matter more than ratings in a recommendation context, as noted by Weimer et al. (2007). A possibility is to replace the rating $\mathcal{M}_{i,j}$ of the j-th item by its rank among all items rated by the i-th user. Another possibility is to consider an order-dependent loss criterion, the Normalized Discounted Cumulative Gain (NDCG), which puts the stress on correctly ordering the top-ranked items. NDCG is used in ALORS and will be detailed in section 4.2.

3.2 The MATCHBOX Approach

Let us describe the MATCHBOX collaborative filtering approach, proposed by Stern et al. (2009), which is at the root of the first attempt to apply collaborative filtering to algorithm selection (Stern et al., 2010).

The probabilistic model learned by MATCHBOX (Fig. 2) estimates the linear mapping U (resp. V) from the initial representation $\mathbf{x}_i$ of the i-th user (resp. $\mathbf{y}_j$ of the j-th item) onto a k-dimensional trait vector $s_i = \mathrm{U}\mathbf{x}_i$ (resp. $t_j = \mathrm{V}\mathbf{y}_j$), where each component in U and V is modelled as a Gaussian variable. The latent rating $r = \mathcal{N}(\langle s_i, t_j \rangle + b_{i,j}, \beta)$, with $b_{i,j}$ the bias associated with the i-th user and the j-th item, and β the noise amplitude, is confronted to the observed ratings, using approximate message passing to iteratively infer the distribution of the U and V components and of the biases.

The flexibility of MATCHBOX partly relies on the learned decoding mechanism, from the latent rating onto the observed (rank-based or binary) $\mathcal{M}_{i,j}$. In the rank-based case for instance, a cumulative threshold model is built (Chu and Ghahramani, 2005), maintaining for each user the $L-1$ thresholds used to segment the continuous latent ratings into L intervals of varying length, respectively mapped onto ranks 1 to L. This decoding mechanism is extended in



A message passing algorithm is used to learn the model (matrices U, V, biases b and noise β), by propagation from the initial representations $\mathbf{x}$ of users and $\mathbf{y}$ of items to the latent trait space $s = \mathrm{U}\mathbf{x}$ and $t = \mathrm{V}\mathbf{y}$ (messages (1)), from the latent space of users and items to their product z (messages (2)), from z to the latent ratings $\bar{r}$ (messages (3)), from latent ratings to the noisy observations r (messages (4)); and the reverse messages propagate back to update the model (messages (5)-(8)).

Figure 2: The Matchbox Factor Graph (from Stern et al. (2009))

(Stern et al., 2010) to handle the scheduler problem (Streeter and Smith, 2008), through inferring a distribution probability on the runtime of an algorithm for a given problem instance, conditionally to the predicted rank of the algorithm.

Another key aspect in MATCHBOX is the approximate message passing algorithm, combining expectation propagation and variational message passing. A primary motivation for this approach is to support incremental learning and thus accommodate the preference drift of the users, whose tastes generally evolve along time; any new information can be exploited to update the model. The second motivation is to support fast approximate inference, through a single pass over the massive collaborative data.

Independently of Stern et al. (2010), Malitsky and O´Sullivan (2014) also achieve algorithm selection by decomposing the collaborative filtering matrix using standard SVD.

3.3 Discussion: The Cold-Start Issue

While algorithm selection can be viewed as a collaborative filtering problem (Stern et al., 2010), the two problems differ in several respects. Firstly, AS is a small or medium-size problem, usually involving a few thousand problems and a few hundred algorithms (or less), thus smaller by two orders of magnitude than CF. Secondly, AS considers stationary data; the performance of an algorithm on a problem instance is defined once for all. Quite the contrary, recommender systems must deal with the fact that user preferences evolve along time, e.g., in the domain of news or music recommendation. These two remarks suggest that the issue of handling massive and non-stationary data is not a key issue for AS.

Thirdly, collaborative filtering mostly considers known users while algorithm selection is mostly if not exclusively concerned with selecting an algorithm for a brand new problem instance − the so-called cold start problem[4]. In the case where no context information about the user is available, AS can only fall back on the best default recommendation. Otherwise, the context information can be used to provide an informed cold start recommendation. As seen above, MATCHBOX determines the latent factors through linear combinations of the initial features. Malitsky and O´Sullivan (2014) use random forests to predict the latent factors from the initial features.

In (Schein et al., 2002), latent classes z are learned and used to decompose the recommendation into two independent probabilities: the probability $P(z|p)$ to select latent class z for user p, and the probability $P(m|z)$ of selecting item m given latent class z. This approach is taken one step further by Gunawardana and Meek (2008), using tied Boltzman machines to model the joint distribution of user ratings depending on the item representation.

Weston et al. (2012) consider the more complex setting of collaborative information retrieval, aimed at retrieving the top items for a user and a query. The collaborative filtering matrix here is a tensor, involving users $\times$ queries $\times$ items. Latent factors are found through decomposing this tensor. The cold-start issue − faced when an item has very few or no users associated with it − is handled by learning a linear mapping from the item features onto the latent factor space.

4 The ALORS System

This section presents an overview of the ALORS system, detailing its two modes: the matrix completion mode achieves algorithm selection for known problem instances; the cold-start mode achieves algorithm selection for new problem instances. The differences between ALORS and MATCHBOX are thereafter discussed.

4.1 Input

ALORS exploits the (n, m) collaborative filtering matrix $\mathcal{M}$ reporting the performance $\mathcal{M}_{i,j}$ of the j-th algorithm for the i-th problem instance for a fraction of the (i, j) pairs. As said, the performance is domain- and application-specific; in the following, it is assumed that the best performance corresponds to the lowest value, associated to rank 1. Besides the performance matrix, ALORS will also exploit the rank matrix M, where $M_{i,j}$ reports the rank of the j-th algorithm on the i-th problem instance (Table 4.1).

ALORS also exploits − for its cold-start functionality only − the (n, d) matrix $\mathbf{X}$, yielding the initial d-dimensional representation of the problem instances.

[4] In all generality, the cold-start problem faces three issues: recommending known items for new users; recommending new items for known users, and recommending new items for new users. Only the first issue is considered in the following.

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_1	s_2	s_3	s_4	s_5	s_6	s_7
i_1	0.21	–	–	0.18	–	0.21	–	2	–	–	1	–	2	–
i_2	–	–	0.056	1.2	–	–	0.069	–	–	1	3	–	–	2
i_3	0.55	0.42	–	–	1	–	–	2	1	–	–	3	–	–
i_4	–	–	–	–	–	0.061	0.18	–	–	–	–	–	1	2
i_5	–	–	0.55	–	0.59	0.29	–	–	–	2	–	3	1	–
	Matrix $\mathcal{M}$							Matrix M						

Table 1: Collaborative filtering matrices. Left: the performance matrix $\mathcal{M}$ reports the performance value of the algorithms (column) on the instances (rows). Right: the rank matrix M, reporting the rank of the algorithm performance for each instance. The lower the performance value, the better.

4.2 The Matrix Completion Functionality

The matrix completion functionality fills in the missing values in the collaborative matrix, using memory- or model-based approaches (section 3.1). For each problem instance, the recommended algorithm is the one with best (initial or filled-in) performance.

4.2.1 Memory-based CF

The memory-based approach computes the similarity of any two problem instances, exploiting either the initial collaborative matrix $\mathcal{M}$ or the rank-based M. Letting $I(i,\ell)$ denote the set of indices j such that both $\mathcal{M}_{i,j}$ and $\mathcal{M}_{\ell,j}$ are available, if $I(i,\ell)$ is not empty the value-based similarity $sim(i,\ell)$ of the i-th and ℓ-th instances is the cosine of the i-th and ℓ-th rows in $\mathcal{M}$, restricted to columns j in $I(i,\ell)$:

$$sim(i,\ell) = \frac{\sum_{k\in I_{i,\ell}} \mathcal{M}_{i,k} \,.\, \mathcal{M}_{\ell,k}}{\sqrt{\sum_{k\in I_{i,\ell}} \mathcal{M}_{i,k}^2} \cdot \sqrt{\sum_{k\in I_{i,\ell}} \mathcal{M}_{\ell,k}^2}}$$

The rank-based similarity $sim_r(i,\ell)$ is likewise defined as:

$$sim_r(i,\ell) = \frac{\sum_{k\in I_{i,\ell}} M_{i,k} \,.\, M_{\ell,k}}{\sqrt{\sum_{k\in I_{i,\ell}} M_{i,k}^2} \cdot \sqrt{\sum_{k\in I_{i,\ell}} M_{\ell,k}^2}}$$

with $sim(i,\ell)$ and $sim_r(i,\ell)$ set to 0 by convention if $I(i,\ell)$ is empty.
The similarities among problem instances is used to estimate the missing performance values using neighbor-based regression. Letting I_ℓ denote the set of indices for which $\mathcal{M}_{\ell,j}$ is known, the value-based estimate noted $\widehat{\mathcal{M}_{i,j}}$ is defined as:

$$\widehat{\mathcal{M}_{i,j}} = \frac{\sum_{\ell \ s.t. \ j \ \in I_\ell} sim(i,\ell)\mathcal{M}_{\ell,j}}{\sum_{\ell \ s.t. \ j \ \in I_\ell} sim(i,\ell)} \tag{3}$$

Likewise, the rank-based estimate noted $\widehat{M_{i,j}}$ is defined as:

$$\widehat{M_{i,j}} = \frac{\sum_{\ell \ s.t. \ j \ \in I_\ell} sim_r(i,\ell)M_{\ell,j}}{\sum_{\ell \ s.t. \ j \ \in I_\ell} sim_r(i,\ell)} \tag{4}$$

4.2.2 Model-based CF

The model-based approach extracts a k-dimensional latent representation of the instances and the algorithms, borrowing Cofirank its optimization method (Weimer et al., 2007). Formally, each i-th problem instance (resp. j-th algorithm) is mapped onto the k-dimensional vector U_i (resp. V_j), such that matrices U and V maximize the Normalized Discounted Cumulative gain (NDCG, (Järvelin and Kekäläinen, 2000)), defined as follows. Let π_i denote the performance order related to the i-th problem instance, with $\pi_i(j)$ being the true rank of the j-th best algorithm after $\langle \mathrm{U}_i, \mathrm{V}_j \rangle$. The discounted cumulative gain (DCG) criterion and its normalized version NDCG enforce the correct ordering of the top L ranked algorithms for each i-th problem instance, by maximizing:

$$NDCG(L) = \frac{1}{Z} \sum_{i=1}^{m} \sum_{\ell=1}^{L} \frac{2^{-\pi_i(\ell)}-1}{log(\ell+1)} \tag{5}$$

with Z a normalization factor. As NDCG is not a convex criterion, a linear upper-bound thereof is used and its optimization is tackled by alternate minimization of U and V (Teo et al., 2007). The CF functionality is thereafter achieved by setting $\widehat{\mathcal{M}_{i,j}}$ to the scalar product of U_i and V_j.

4.3 The Cold Start Functionality

The cold start (CS) functionality achieves algorithm selection for a new problem instance, for which the latent representation cannot be determined from the CF matrix by construction. Alors exploits the known problem instances to learn the latent representation from the initial representation as follows. Each problem instance defines a training example $(\mathbf{x}_i, \mathrm{U}_i)$. From the training set

$$\mathcal{E} = \{((\mathbf{x}_i, \mathrm{U}_i), \mathbf{x}_i \in \mathbb{R}^d, \mathrm{U}_i \in \mathbb{R}^k, i \in 1 \ldots n\}$$

is learned a mapping $\Phi : \mathbb{R}^d \mapsto \mathbb{R}^k$, using random forests[5]. For each new problem instance with initial representation $\mathbf{x}$, the associated latent representation $\mathrm{U}_\mathbf{x}$ is approximated as $\Phi(\mathbf{x})$. The cold-start problem is then brought back to the matrix completion one, by estimating the performance of the j-th algorithm (or the rank thereof) on the new problem instance as $\langle \Phi(\mathbf{x}), \mathrm{V}_j \rangle$ and algorithm selection then follows (Alg. 1).

[5] The open source library scikit-learn is used with default values (Pedregosa et al., 2011).

Algorithm 1 ALORS: Cold-Start Functionality

Input
Collaborative filtering matrix $\mathcal{M} \in \mathbb{R}^{n \times m}$
Initial representation $\mathbf{X} \in \mathbb{R}^{n \times d}$ of the problem instances
Description $\mathbf{x}$ of the current problem instance
Parameter L of the NDCG criterion (Eq. 5)

Collaborative filtering (Section 4.2.2)
Build matrices U and V by solving Eq 2.

Learning latent factor model Φ
Build $\mathcal{E} = \{(\mathbf{x}_i, \mathrm{U}_i), i = 1 \dots n\}$.
Learn $\Phi : \mathbb{R}^d \mapsto \mathbb{R}^k$ from $\mathcal{E}$

Algorithm selection
Compute $\mathrm{U}_{\mathbf{x}} = \Phi(\mathbf{x})$
Return $\underset{j=1 \dots m}{\operatorname{argmin}} \langle \mathrm{U}_{\mathbf{x}}, \mathrm{V}_j \rangle$

4.4 Discussion

The main difference between the two collaborative filtering-based approaches, ALORS and MATCHBOX, lies in the extraction of the latent representation. MATCHBOX determines the latent representation $\mathrm{U}_{\mathbf{x}}$ associated to some initial representation $\mathbf{x}$ by looking for the matrix U such that $\mathrm{U}_{\mathbf{x}} = \mathrm{U}\mathbf{x}$, where all components in U are independent Gaussian scalar variables. ALORS builds the latent representation $\mathrm{U}_{\mathbf{x}}$ only from the collaborative filtering matrix. The initial representation is only used in a second phase, to learn the mapping from the initial representation $\mathbf{x}$ onto the latent representation $\mathrm{U}_{\mathbf{x}}$.

Decoupling the extraction of the latent representation, and its characterization from the initial representation enables to independently assess the representativity of the benchmark problems, and the quality of the initial representation:

- The sample representativity of the problem instances is assessed from the ability to recover the full collaborative filtering matrix from an excerpt thereof, measured by the matrix completion performance based on the latent factors;
- The quality of the initial representation of the problem instances is assessed from the ability to estimate the latent factors. Assuming that the matrix completion performance is good, the latent representation can be viewed as an oracle representation, in the sense that it captures the whole information of the collaborative filtering matrix. Therefore, the initial representation is good iff it enables to estimate the latent representation; the latent factors need not be restricted to linear combinations of the initial features.

The main difference between ALORS and the CF-based approach proposed by Malitsky and O´Sullivan (2014) lies in the decomposition of the CF matrix. The latter authors use a standard singular value decomposition aimed at recovering

the full matrix, while ALORS focuses on preserving the order of the top-ranked algorithms for each problem instance (using the NDCG criterion, Eq. 5). The rationale for using an order-based criterion is to increase the generality of the approach with respect to the measure of performance: the solution only depends on the algorithm ranking with respect to a given problem instance.

5 Experiment Goals and Setting

This section presents the experimental setting used for the comparative empirical validation of ALORS. The Matrix Completion performance, noted CF, measures the ability to reconstruct matrix $\mathcal{M}$ from an excerpt thereof. After the above discussion, the CF performance reflects the quality of the set of problem instances, called benchmark in the following. Secondly, the Cold Start performance, noted CS, measures the quality of algorithm selection for a new problem instance; it reflects the quality of the initial representation of the problem instances, and whether they enable to learn the latent factors. The sensitivity of both performances with respect to the incompleteness rate p of the $\mathcal{M}$ matrix and the number k of latent factors is studied on two real-world benchmarks. Additionally, an artificial problem is defined to investigate and compare the cold-start performances of ALORS and MATCHBOX.

5.1 Real world benchmarks

The first benchmark is the *Algorithm Selection Benchmark Library* (ASlib)[6], centered on propositional satisfiability and constraint satisfaction, summarized in Table 2. The interested reader is referred to (Bischl et al., 2015) for a detailed presentation. All unsolved instances are preliminarily removed. The ALORS performance is assessed using three indicators: i) *Rank* is the true rank of the selected algorithm, averaged on all problem instances; ii) *Par10* is the runtime it requires to solve a problem instance, averaged over all problem instances where the penalty for each unsolved instance is 10 times the runtime budget per instance; iii) *SolvedRatio* is the fraction of problem instances solved by the selected algorithm.

The second benchmark gathers the results of the OpenML[7] platform (van Rijn et al., 2013), reporting the performance of 292 algorithm-configuration pairs on 76 problem instances. Each algorithm-configuration pair involves one out of 42 supervised machine learning algorithms, with a distinct configuration (values of the algorithm discrete or continuous hyper-parameters); they are handled as 292 distinct algorithms in the following. Performance $\mathcal{M}_{i,j}$ is the test classification error of the j-th algorithm on the i-th problem instance. The number d of descriptive features is 11. The ALORS performance is assessed using two indicators. The first one, *Rank*, is the true rank of the selected algorithm,

[6] https://github.com/coseal/aslib_data/releases

[7] http://openml.org/. The authors thank Joaquin Vanschoren for making these data publicly available.

Dataset	#Instances	#Algorithms	#Features
ASP-POTASSCO	1294	11	138
CSP-2010	2024	2	86
MAXSAT12-PMS	876	6	37
PREMARSHALLING-ASTAR-2015	527	4	22
PROTEUS-2014	4021	22	198
QBF-2011	1368	5	46
SAT11-HAND	296	15	115
SAT11-INDU	300	18	115
SAT11-RAND	600	9	115
SAT12-ALL	1614	31	115
SAT12-HAND	767	31	115
SAT12-INDU	1167	31	115
SAT12-RAND	1362	31	115

Table 2: The Aslib benchmark (Bischl et al., 2015).

averaged on all problem instances. However, the statistical noise of the test error might induce non statistically significant differences among the $\mathcal{M}_{i,j}$ associated to a same problem instance, adversely affecting the stability of the ranks. A second indicator is thus considered: the *Regret* is defined as the test error of the selected algorithm minus the test error of the true best algorithm, averaged over all problem instances.

The ALORS performance is compared to three baselines: i) the Oracle, selecting the best algorithm for each problem instance; ii) the SingleBest (best algorithm on average on all problem instances in the dataset); iii) the average performance of all algorithms (legend Random). The lack of comparison with the most recent AS algorithms, such as the descendant of SATzilla (section 2.2) algorithm dominating the 2015 ICON competition[8], is due to the fact that current AS algorithms involve a pre-scheduler component (launching a fixed sequence of algorithms for a fixed budget each for each problem instance, and achieving algorithm selection for problem instances which are not solved by the pre-scheduler). Their solution space is thus significantly more complex than that of an AS system like ALORS, rendering the comparison meaningless. Nevertheless, the comparison with the oracle, single best and average performances indicates how close the AS algorithm is from the optimal performance.

5.2 Experimental setting

The number k of latent factors is set to 10, or to m if the number m of algorithms is less than 10 ($k = min(10, m)$). For every p, the matrix completion performance $\text{CF}(p)$ and cold start performance $\text{CS}(p)$, computed as follows, are

[8] http://challenge.icon-fet.eu/challengeas

graphically depicted using a boxplot. The comparison of the latent and initial representations of a problem domain also offers some visual and quantitative insights into the quality of the initial representation (section 6.4).

ALORS parameters are summarized in Table 3.

Phase	**Parameter**	**Range**
	Incompleteness rate p	$\{.1, \ldots .9\}$
Matrix Completion		
Model-based	number of latent factors k	$\min(10, m)$
	NDCG parameter L	10
	Regularization parameter λ	10
Cold start		
Random Forest	Number of trees	10
	Split criterion	mse
	Minimum number of samples to split	2

Table 3: The ALORS parameters. The Matrix Completion parameters are determined from few preliminary experiments; the number k of latent factors is set to 10, or m if $m <$ 10. The Cold Start parameters are set to the default values, using mean square error for the split criterion in the Random Forest.

5.2.1 Matrix Completion

For each incompleteness rate p ranging in $\{10, \ldots, 90\}$, the CF(p) performance is computed by:
i) uniformly selecting and removing p% of the entries from matrix $\mathcal{M}$, subject to keeping at least one entry on each line and column, thus forming the sparsified matrix $\mathcal{M}_p$;
ii) building k-dimensional latent matrices U and V from $\mathcal{M}_p$;
iii) using the latent factors to fill-in matrix $\mathcal{M}_p$ and determining the selected algorithm for each i-th problem instance;
iv) recording the true rank $r(i)$ of the selected algorithm, its penalized runtime and ratio of solved instances (or its regret for the OpenML benchmark), and taking their average over all n problem instances;
v) reporting the average performances over 10 independent drawings of sparsified matrix $\mathcal{M}_p$.

5.2.2 Cold Start

For each incompleteness rate p ranging in $\{10, \ldots, 90\}$, the CS(p) performance is measured using a ten-fold cross validation, using an equi-partition of the set of problem instances into 10 subsets. Thereafter:
i) Let matrix $\mathcal{M}_{-\ell}$ denote the union of all rows in $\mathcal{M}$ except those in the ℓ-th

subset;
ii) A fraction $p\%$ of the entries in $\mathcal{M}_{-\ell}$ is removed as above, defining matrix $\mathcal{M}_{-\ell,p}$; $\mathcal{M}_{-\ell,p}$ is used to build k-dimensional latent matrices U and V;
iii) Mapping Φ is learned from training set $\mathcal{E}_{-\ell,p}$, with

$$\mathcal{E}_{-\ell,p} = \{(\mathbf{x}_i,\ \mathrm{U}_i) \text{ for } i \text{ ranging in the rows of } \mathcal{M}_{-\ell}\}$$

iv) Φ is used to estimate $\mathcal{M}_{i,j}$ for every i-th problem instance in the ℓ-th subset and select the algorithm with optimal estimated $\mathcal{M}_{i,j}$;
v) the true rank of the selected algorithm, its penalized runtime or regret are averaged over all problem instances in the ℓ-th subset;
vi) these performances are averaged over 10 independent drawings of sparsified matrix $\mathcal{M}_{-\ell,p}$, and over all 10 folds ($\ell = 1 \ldots 10$).

5.3 Artificial Cold Start Problem

An artificial setting is considered to compare MATCHBOX and ALORS. If the initial features provide a good linear model of the CF matrix, MATCHBOX should dominate ALORS as it explores a simpler search space. If on the contrary the latent factors are too complex to be captured by linear combinations of the initial features, ALORS should dominate MATCHBOX. In the general case where the quality of the initial features is unknown, the question is whether one should consider a rich set of features and search for linear latent factors, or build latent factors from the collaborative filtering matrix and model them as (non necessarily linear) functions of the features.

This question is empirically investigated by defining an artificial AS problem, involving 200 problem instances and 30 algorithms. For each i-th problem instance (respectively, j-th algorithm), an initial representation noted $\mathbf{x}_i$ (resp. $\mathbf{y}_j$) is uniformly generated in $[-10, 10]^{10}$, and the performance $\mathcal{M}_{i,j}$ is set to the Euclidean distance of $\mathbf{x}_i$ and $\mathbf{y}_j$ restricted to their first 3 coordinates, plus a Gaussian noise $\mathcal{N}(0, \epsilon)$ with ϵ ranging in $\{.1, .25, .5, 1\}$. By construction, $\mathcal{M}_{i,j}$ cannot be modelled as a linear function of the initial representation. Therefore, the initial representation of the problem instances in $\mathbb{R}^{10}$ is enriched by another 55 features, the squares and the cross-products of the initial features. The true performance model now belongs to both MATCHBOX and ALORS search space. MATCHBOX has to find a linear model depending on 6 out of 65 features, while ALORS has to find i) latent features; ii) an approximation of these features using the available 65 features.

6 Empirical validation

This section presents the validation of ALORS, distinguishing the Matrix Completion (section 6.1) and the Cold Start (section 6.2) performances. Section 6.3 reports on the experimental comparison of MATCHBOX and ALORS, and section 6.4 discusses the lessons learned from comparing the latent and the initial

representations of the domain. All experiments are performed on an Intel Core i5-4690 3.50GHz PC with Ubuntu 14.04.

6.1 The Matrix Completion Performance

The Matrix Completion performance measures whether the information contained in part of the collaborative filtering matrix $\mathcal{M}$ is sufficient to recover the rest of the information. To account for the varying difficulty of the ASlib datasets, three indicators are reported:

- The rank of the single best algorithm for a given dataset reflects the heterogeneity of the problem instances of this dataset;
- the average rank of ALORS for an incompleteness rate of 50% indicates how far ALORS can go with only half of the initial information;
- finally, the maximum incompleteness rate such that ALORS significantly outperforms the single best baseline (with 95% confidence after Wilcoxon Rank Sum Test) reflects the representativity of the dataset.

Dataset	**Rank**		**SolvedRatio**		**Max p**
	Single-Best	ALORS **p=50%**	**Single-Best**	ALORS **p=50%**	
ASP-POTASSCO	4.71	1.89	0.92	0.99	80%
CSP-2010	1.22	1.22	0.98	0.98	40%
MAXSAT12-PMS	2.5	1.79	0.9	0.98	70%
PREMARSHALLING-ASTAR-2015	2.07	1.62	0.81	0.94	60%
PROTEUS-2014	9.8	2.17	0.71	0.95	90%
QBF-2011	2.39	1.7	0.75	0.91	60%
SAT11-HAND	6.37	2.24	0.68	0.93	80%
SAT11-INDU	7.21	2	0.85	0.99	90%
SAT11-RAND	3.73	1.87	0.74	0.97	80%
SAT12-ALL	11.97	2.67	0.76	0.95	90%
SAT12-HAND	12.11	3.18	0.68	0.93	90%
SAT12-INDU	8.68	2.21	0.9	0.99	90%
SAT12-RAND	3.73	2.06	0.96	0.99	80%

Table 4: Matrix Completion performances on the ASlib benchmark: Rank and SolvedRatio for the single best baseline and for ALORS with incompleteness rate $p = 50\%$; Maximum p such that ALORS statistically significantly outperforms the single-best (with 95% confidence after Wilcoxon Rank Sum Test).

The reported ALORS results are obtained with the parameter-less memory-based approach (section 4.2.1), which outperforms the model-based approach

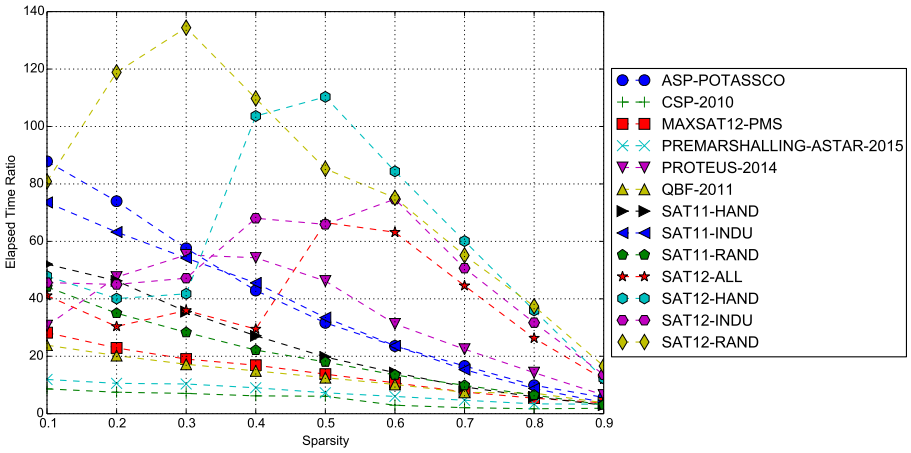


Figure 3: Matrix Completion Runtimes: ratio of the model based and memory based approach runtimes. The memory-based approach, with quadratic complexity in the number of problem instances and linear complexity in the number of algorithms, runs in a fraction of second.

(except for very high p values) and is significantly less computationally demanding (Fig. 3).

After Table 4, four different categories of datasets are found in the ASlib benchmark. One of them includes the only CSP-2010 dataset (Fig. 4), with m=2 algorithms, and where the single best algorithm with an average rank of 1.22 is hard to beat. For incompleteness rate above 40%, ALORS does no better than the single best baseline.

A second category includes the PREMARSHALLING-ASTAR-2015 (Fig. 5) and the MAXSAT-12-PMS datasets (Fig. 6), with a low average rank of the single best baseline (respectively 2.07 and 2.5). On these datasets, the information is sufficient for ALORS to significantly improve on the single-best baseline for incompleteness rate respectively $p = 60\%$ and 70%.

The third category includes 6 datasets (ASP-POTASSCO, QBF-2011, SAT11-HAND, SAT-11-INDU, SAT11-RAND, SAT12-RAND), with a higher rank of the single best algorithm (in $[3.73, 6.37]$, except for QBF-2011 where it is 2.39), which suggests that these datasets involve more diverse problem instances. The ALORS performances are displayed on the ASP-POTASSCO dataset (Fig. 7), which is representative of the third category; they gracefully degrade as p increases and ALORS still significantly outperforms the single best baseline for $p = 80\%$ (except for QBF-2011 where $p = 70\%$).

The fourth category includes 4 datasets (PROTEUS-2014, SAT12-ALL, SAT12-HAND and SAT12-INDU), with a very high rank of the single best algorithm (in $[8.68, 12.11]$), which suggests that the problem instances are very heterogeneous.

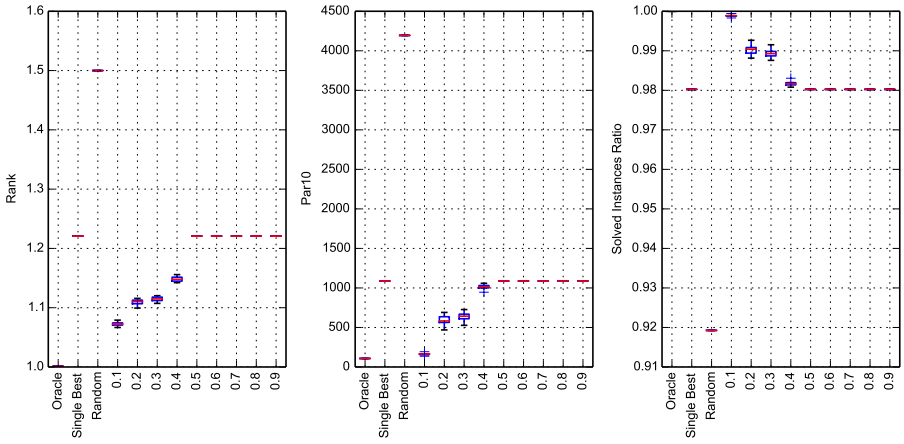


Figure 4: Matrix Completion performances (left: Rank; middle: Penalized Runtime; right: Ratio Solved). On each plot, from left to right: Oracle, Single best and Average baselines, and ALORS (for incompleteness rate in 10% ... 90%) on CSP-2010.

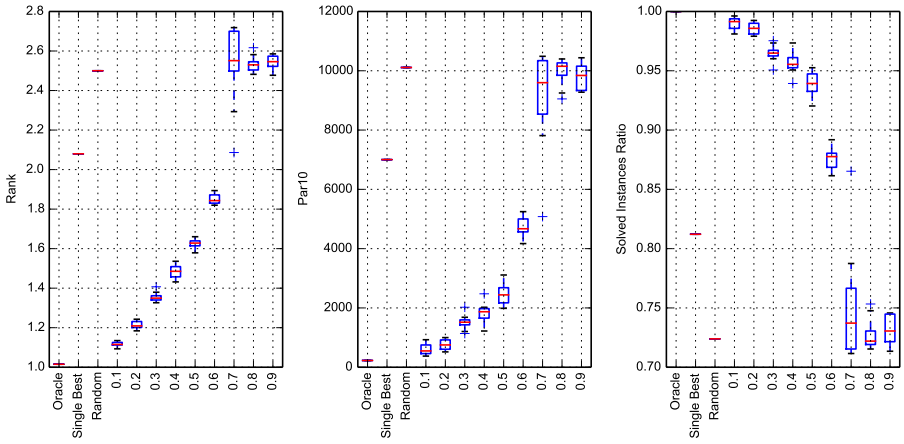


Figure 5: Matrix Completion performances (left: Rank; middle: Penalized Runtime; right: Ratio Solved). On each plot, from left to right: Oracle, Single best and Average baselines, and ALORS (for incompleteness rate in 10% ... 90%) on PREMARSHALLING-ASTAR-2015.

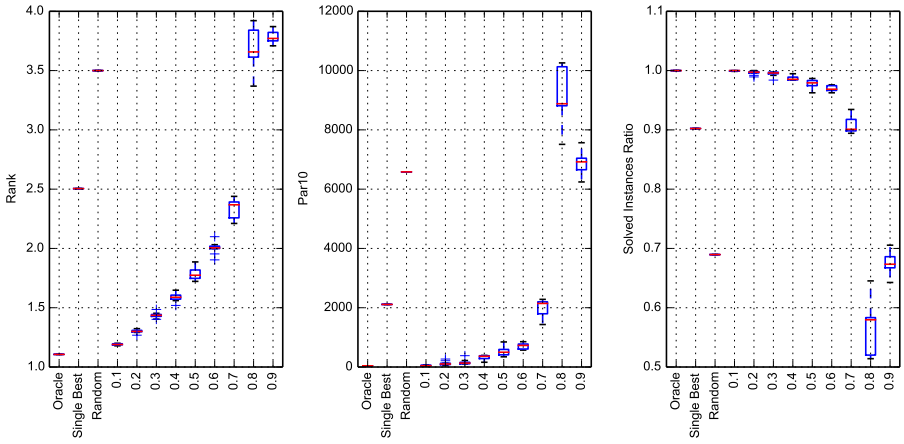


Figure 6: Matrix Completion performances (left: Rank; middle: Penalized Runtime; right: Ratio Solved). On each plot, from left to right: Oracle, Single best and Average baselines, and ALORS (for incompleteness rate in 10% ... 90%) on MAXSAT12-PMS.

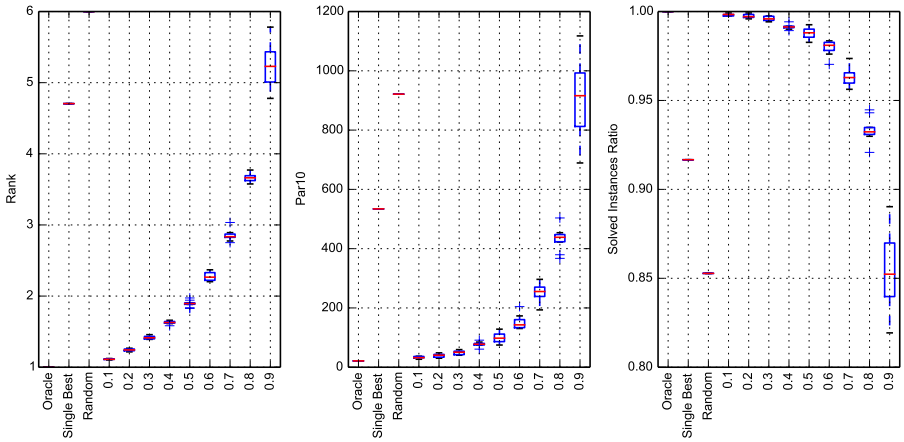


Figure 7: Matrix Completion performances (left: Rank; middle: Penalized Runtime; right: Ratio Solved). On each plot, from left to right: Oracle, Single best and Average baselines, and ALORS (for incompleteness rate in 10% ... 90%) on ASP-POTASCCO.

The ALORS performances are displayed on the PROTEUS-2014 dataset (Fig. 8), which is representative of the fourth category; they gracefully degrade as p increases, with a lower variance than for the third category. ALORS significantly outperforms the single best baseline for $p = 90\%$.

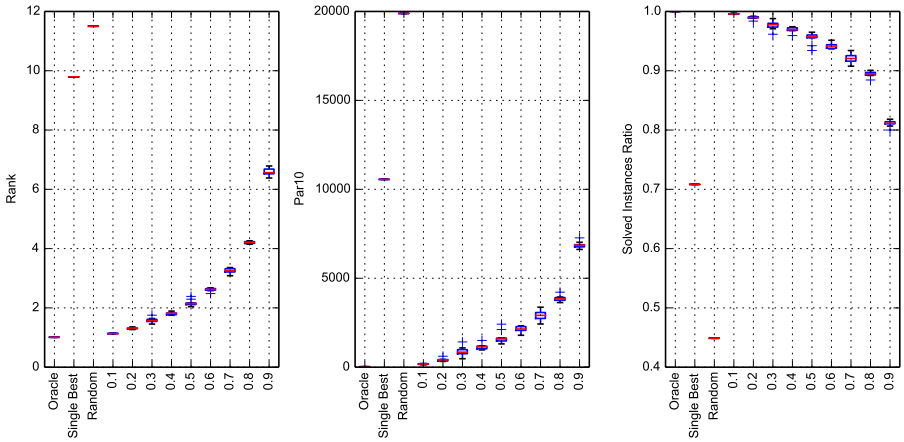


Figure 8: Matrix Completion performances (left: Rank; middle: Penalized Runtime; right: Ratio Solved). On each plot, from left to right: Oracle, Single best and Average baselines, and ALORS (for incompleteness rate in 10% ... 90%) on PROTEUS-2014.

On the OpenML dataset, a different domain performance indicator (the regret or excess prediction loss compared to the oracle algorithm, section 5.1) is used, preventing the direct comparison with the ASlib benchmark. The results (Fig. 9) show a high diversity of the problem instances, with the rank of the single best baseline circa 60 – with the caveat that the statistical significance of the differences between the algorithm performances is not available. The representativity of the problem instances is very good as ALORS very significantly outperforms the single best baseline for incompleteness rate $p <= 90\%$.

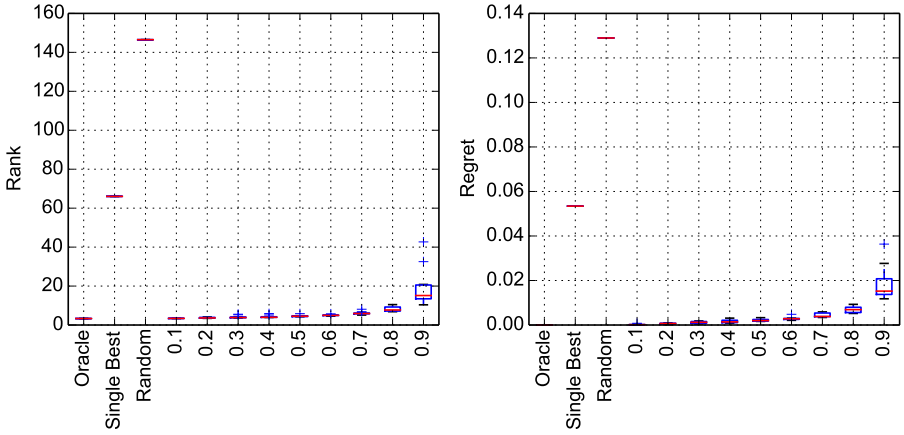


Figure 9: Matrix Completion performances (left: Rank; right: Regret). On each plot, from left to right: Oracle, Single best and Average baselines, and ALORS (for incompleteness rate in 10% ... 90%) on OpenML.

6.2 The Cold Start Performance

As said, the Cold Start performance depends on the representativity of the benchmark (governing the quality of the latent factors) and the quality of the initial representation (governing the estimation of the latent factors). On the ASlib benchmark, Table 5 reports the Rank and SolvedRatio of ALORS for $p = 0\%$ and 50%, together with the maximal incompleteness rate such that ALORS Rank significantly outperforms the single best Rank.

The CSP-2010 dataset stands out as being particularly difficult, for the same reasons as in the Matrix Completion case.

A second category of problems includes PREMARSHALLING-ASTAR-2015 (Fig. 10), SAT11-INDU and SAT12-RAND (Fig. 11). On these problems, ALORS with $p = 50\%$ reaches similar performances as the single best baseline; surprisingly, ALORS with $p = 0$ (expectedly yielding better latent factors) does not much improve on $p = 50\%$. This second CS category does not much overlap with the second MC category: the Matrix Completion performances were excellent on the SAT11-INDU and SAT12-RAND datasets (99% of solved problems for $p = 50\%$), suggesting that the problem might come from the initial features.

The third category includes all other datasets, where the results are good (ASP-POTASSCO, MAXSAT12-PMS, SAT11-HAND, SAT11-INDU, SAT12-INDU, illustrated on the representative case of ASP-POTASSCO on Fig. 12), or very good (PROTEUS-2014, QBF-2011, SAT11-RAND, SAT12-ALL, SAT12-HAND, illustrated on the representative cases of PROTEUS-2014, Fig. 13 and QBF-2011, Fig. 14).

As expected, the performances are better for Matrix Completion with $p =$

Dataset	Average Ranks		Solved Inst. Ratio		Max p
	p=0%	p=50%	p=0%	p=50%	
ASP-POTASSCO	2.94	3.53	0.98	0.96	60%
CSP-2010	1.12	1.22	1	0.98	20%
MAXSAT12-PMS	1.63	1.99	0.98	0.95	60%
PREMARSHALLING-ASTAR-2015	1.85	2	0.86	0.83	20%
PROTEUS-2014	4.82	5.91	0.87	0.83	80%
QBF-2011	1.38	1.75	0.97	0.9	70%
SAT11-HAND	3.36	4.74	0.88	0.77	40%
SAT11-INDU	5.59	6.38	0.9	0.87	$\sim$
SAT11-RAND	3.19	3.37	0.9	0.81	50%
SAT12-ALL	5.39	8.67	0.92	0.8	70%
SAT12-HAND	5.2	8.97	0.91	0.75	60%
SAT12-INDU	4.83	5.92	0.96	0.93	50%
SAT12-RAND	3.42	3.69	0.96	0.96	$\sim$

Table 5: Cold Start performances on the ASlib benchmark: Rank and SolvedRatio of ALORS for $p = 0\%$ and 50% (the single best baseline performances are reported in Table 4); Maximum p such that ALORS SolvedRatio is significantly higher than the single-best baseline; $\sim$ indicates no statistically significant difference between ALORS and single best SolvedRatios.

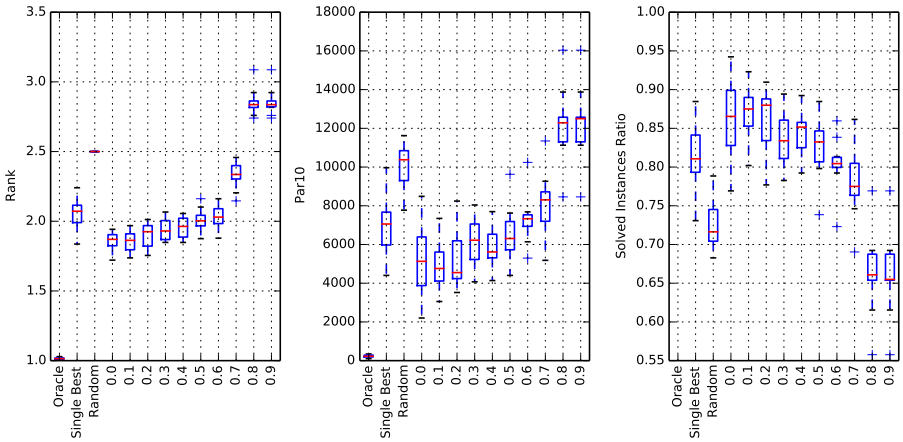


Figure 10: Cold Start performances (left: Rank; middle: Penalized Runtime; right: Ratio Solved). On each plot, from left to right: Oracle, Single best and Average baselines, and ALORS (for incompleteness rate in 10% ... 90%) on PREMARSHALLING-ASTAR-2015.

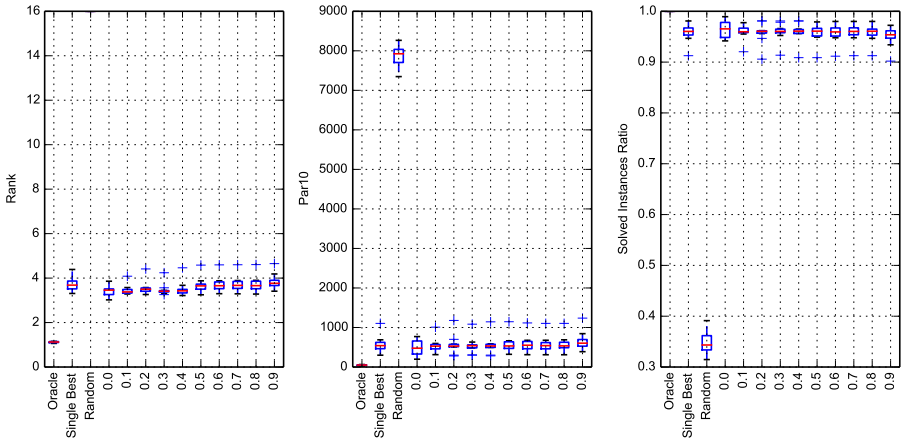


Figure 11: Cold Start performances (left: Rank; middle: Penalized Runtime; right: Ratio Solved). On each plot, from left to right: Oracle, Single best and Average baselines, and ALORS (for incompleteness rate in 10% ... 90%) on SAT12-RAND.

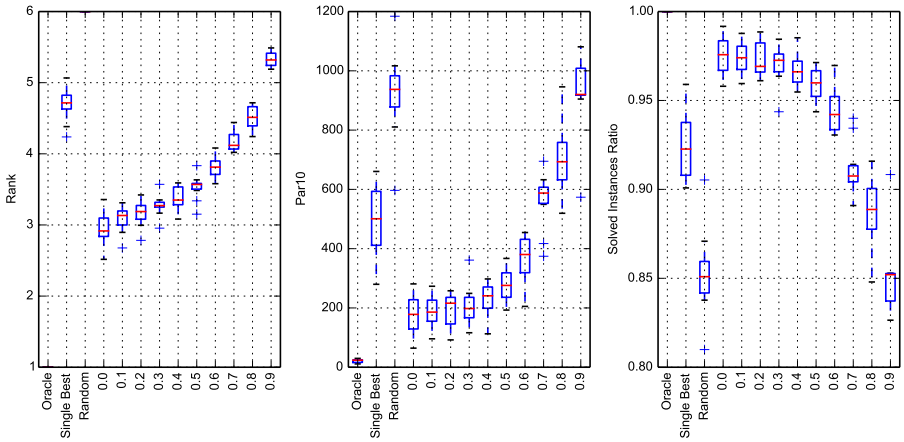


Figure 12: Cold Start performances (left: Rank; middle: Penalized Runtime; right: Ratio Solved). On each plot, from left to right: Oracle, Single best and Average baselines, and ALORS (for incompleteness rate in 10% ... 90%) on ASP-POTASSCO.

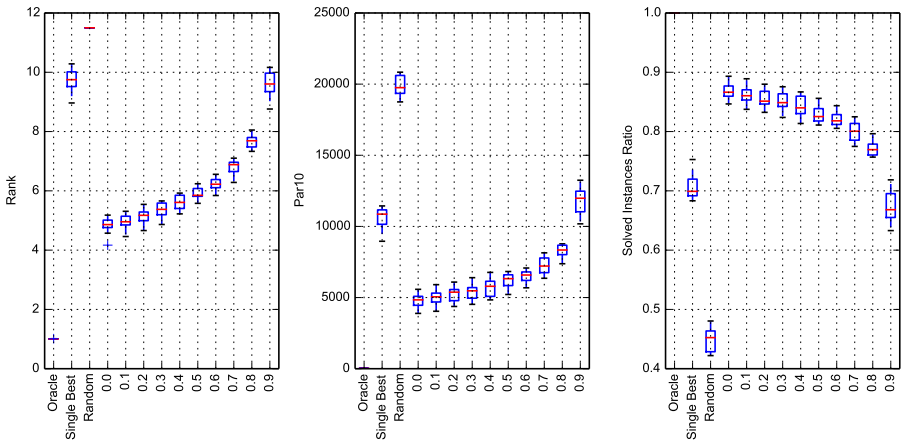


Figure 13: Cold Start performances (left: Rank; middle: Penalized Runtime; right: Ratio Solved). On each plot, from left to right: Oracle, Single best and Average baselines, and ALORS (for incompleteness rate in 10% ... 90%) on PROTEUS-2014.

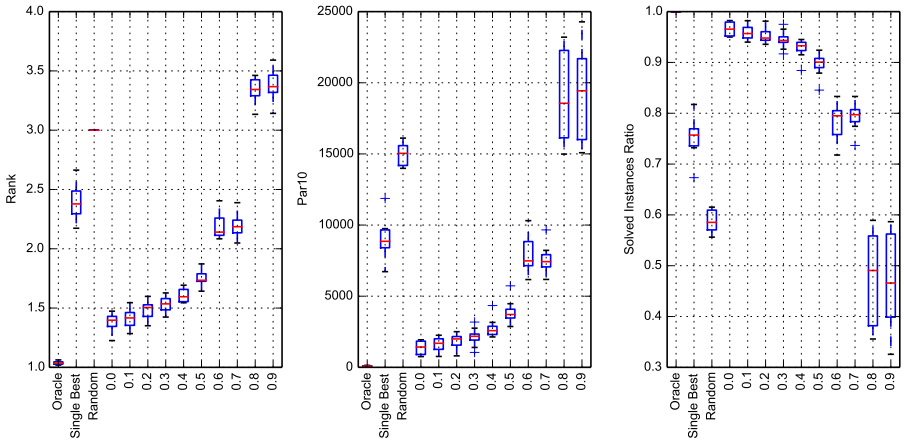


Figure 14: Cold Start performances (left: Rank; middle: Penalized Runtime; right: Ratio Solved). On each plot, from left to right: Oracle, Single best and Average baselines, and ALORS (for incompleteness rate in 10% ... 90%) on QBF-2011.

50% than for Cold Start for $p = 0\%$, particularly so for PREMARSHALLING-ASTAR-2015 (SolvedRatio decreases from .94 to .87) and PROTEUS-2014 (SolvedRatio decreases from .95 to .87); the decrease is moderate in the general case.

On the OpenML benchmark, quite the contrary, the Cold Start performances are very degraded compared to the Matrix Completion performances (Fig. 15). ALORS is outperformed by the single best baseline, for all p values. This counter-performance can only be blamed on the initial representation of the OpenML problem instances; we shall return to this point in section 6.4.

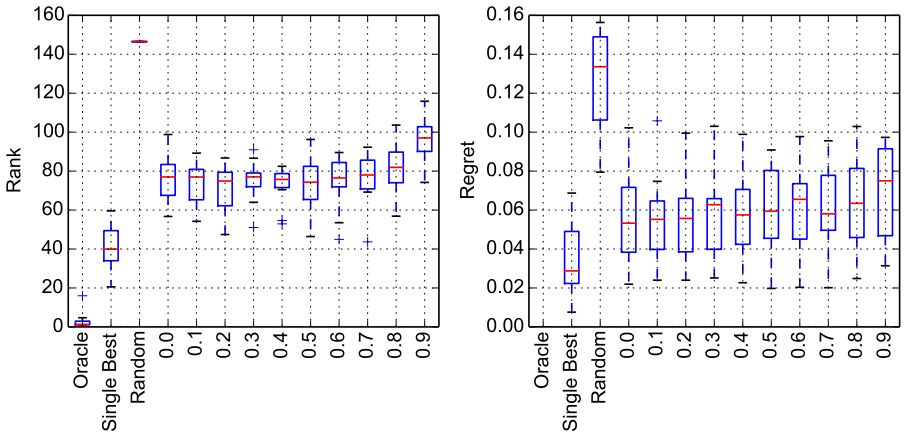


Figure 15: Cold Start performances (left: Rank; right: Regret). On each plot, from left to right: Oracle, Single best and Average baselines, and ALORS (for incompleteness rate in 10% ... 90%) on OpenML. The single-best baseline improvement compared to the Matrix Completion setting (Fig. 9) is explained as the single best algorithm is determined from 1/10th of the problem instances (the test fold).

The sensitivity of the results with respect to the number k of latent factors is illustrated on the representative case of SAT11-HAND (Fig. 16); the performance gracefully improves for $k \geq 2$ and a plateau is reached for $k = 10$.

The computational runtime is reported on Fig. 17.

6.3 Comparative evaluation of ALORS and MATCHBOX

We thank Stern et al. (2010) for enabling us to use MATCHBOX through the web-based Azure ML studio.

The comparison of MATCHBOX and ALORS on the ASlib benchmark does not show statistically significant differences; on the OpenML benchmark it faces some technical issues[9].

[9] MATCHBOX does not run on the full OpenML dataset, due to having more algorithms than

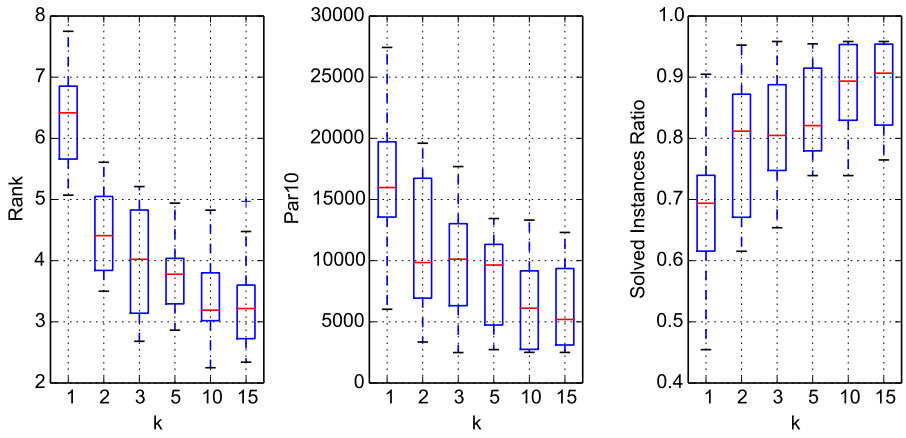


Figure 16: ALORSCold Start performances (left: Rank; middle: Penalized Runtime; right: Ratio Solved): Sensitivity analysis w.r.t. the number k of latent factors on SAT11-HAND.

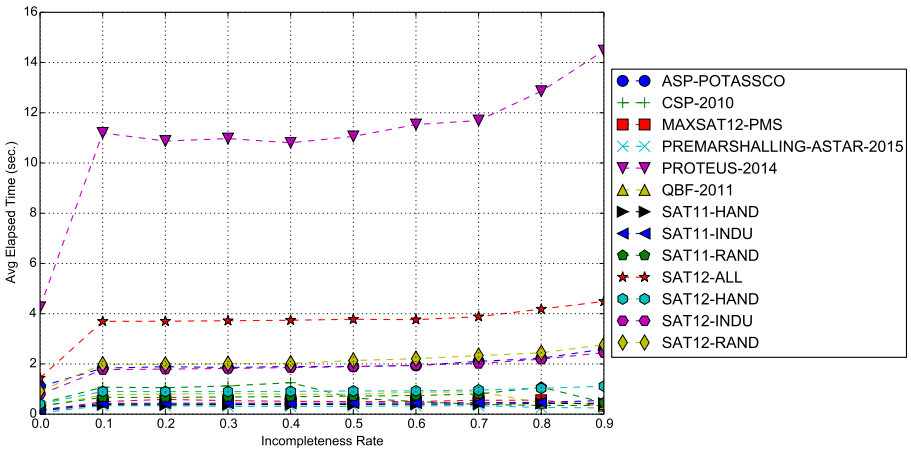


Figure 17: ALORS runtime (latent factors extraction, matrix filling-in, latent factors learning, prediction) on ASlib datasets *vs* incompleteness rate p (remind that there is no matrix filling-in for $p = 0$).

Computationally-wise, the fair comparison of the two approaches is hindered as they do not run in the same environment. To give a rough idea, the cold start results averaged for all p values for the OpenML dataset required 0.16 seconds on average for ALORS and 19.4 seconds for MATCHBOX.

The comparison between MATCHBOX and ALORS thus considers the artificial dataset introduced in section 5.3 ($n = 200, m = 30$). The sensitivity w.r.t. the performance noise ϵ (in $\{.1, .25, .5, 1\}$) and the incompleteness rate (p in $\{0, .2, .5, .8\}$) is studied along the same Cold Start experimental setting (section 5.2).

As shown on Table 6, ALORS outperforms MATCHBOX (with 95% confidence after Wilcoxon Rank Sum Test) for low noise values, $\epsilon = .1$ or .25. The proposed interpretation is that the number of irrelevant features enables MATCHBOX to overfit the decomposition of the collaborative filtering matrix (remind that latent factor $\mathrm{U}_{\mathbf{x}}$ is the product of the sought matrix U with the initial representation $\mathbf{x}$). This interpretation is supported by the fact that MATCHBOX results are improved when the incompleteness rate increases ($p = .2$ or .5), relaxing the constraints on the U matrix. In the meanwhile, ALORS extracts the latent factors by considering the only collaborative filtering matrix; the influence of the irrelevant features only intervenes when learning the latent factors.

For higher noise values ($\epsilon = .5$ or 1), both approaches yield similar results, with the average rank close to the random guess ($m = 30$).

This experiment confirms that MATCHBOX and ALORS rely on different assumptions about the available data, which lead to different regularization strategies when extracting the latent factors. ALORS only uses the number of latent factors as regularizer, while MATCHBOX additionally enforces a strong regularization by requiring the latent factors to linearly depend on the initial features. As said, the MATCHBOX strategy was primarily designed to handle the noisy large-size collaborative filtering problem, whereas ALORS was designed to handle the noiseless small- or medium-size algorithm selection problem.

6.4 Comparing the latent and the initial representations

Under the assumption that the Matrix Completion results are good, the latent factors can be viewed as "oracle features" in the sense that they encapsulate all information required for AS (unfortunately, only available for known problem instances). This section discusses how the comparison of the oracle and the initial features can provide insights into the AS problem.

6.4.1 Visual inspection of the clusters

A first possibility is to visually inspect both representations, using multi-dimensional scaling (Borg and Groenen, 2005) to map the set of problem instances in the initial or latent representations onto $\mathbb{R}^2$. The idea is that if the topology defined by the initial representation coincides with the topology defined by the latent

problem instances ($n < m$).

ϵ	p	MATCHBOX	ALORS
.1	0	6.47	3.08
	.2	5.57	4.38
	.5	4.77	5.38
	.8	10.97	6.38
.25	0	9.32	7.65
	.2	7.17	7.3
	.5	8.37	8
	.8	13.57	10.43
.5	0	10.55	11.98
	.2	10.6	10.15
	.5	11.85	9.68
	.8	12.42	10.9
1	0	12.9	14.28
	.2	12.67	13.6
	.5	16.6	13.38
	.8	15.7	15.53

Table 6: Cold Start: Comparison of MATCHBOX and ALORS on the artificial problem (section 5.3), with $n = 200$ problem instances and $m = 30$ algorithms.

one, then similar problem instances can confidently be recommended same algorithms. This holds if the clusters based on the latent representation, thereafter called latent clusters, match clusters in the initial representation, thereafter called initial clusters.

Let $\mathbf{x}_i$ (respectively $\mathbf{z}_i$) thereafter denote the initial (resp. latent) representation of the i-th problem instance. The consistency of the latent clusters in the initial representation is visually inspected as follows:
i) Standard K-means clustering (Hartigan, 1975) is applied on the $\mathbf{z}_i$s, where the appropriate value of K is determined using the Silhouette score (Rousseeuw, 1987);
ii) Each $\mathbf{z}_i$ is associated the color of its cluster (Fig. 18, bottom right);
iii) The $\mathbf{x}_i$s with same color as the $\mathbf{z}_i$s are visualized in the initial representation (Fig. 18, bottom left), checking whether neighbor points have same color.
The same methodology is applied to visualize the consistency of the initial clusters in the latent representation (Fig. 18, top row).

The comparison of the latent and initial representations is illustrated on three datasets. All datasets have very good Matrix Completion performances; their Cold Start performances vary from very good (SAT12-ALL) to average (PREMARSHALLING-ASTAR) to poor (OpenML). For SAT12-ALL, clusters are dense for both representations; while clusters are mixed, points are not isolated (there are points of the same color in close neighborhoods, Fig. 18 top right or bottom left). Clusters are much less dense for the PREMARSHALLING-ASTAR-2015 dataset, particularly so for the latent representation as $k = m = 4$ (Fig. 19, top right and bottom right). The increased difficulty of the cold start

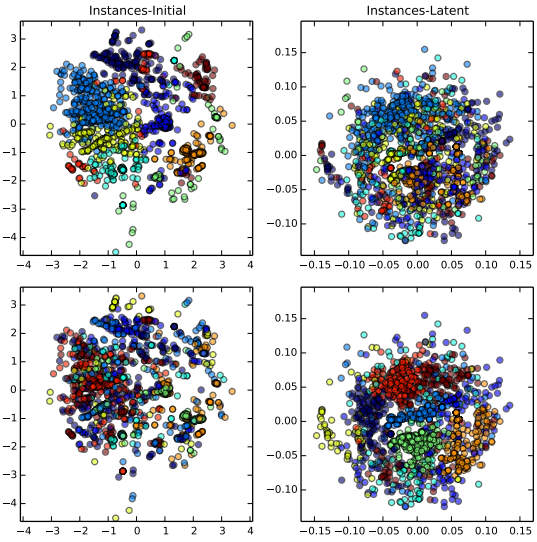


Figure 18: The initial and latent representations of the problem instances in the SAT12-ALL dataset. Top row: Left: the clusters in the initial representation. Right: their image in the latent representation. Bottow row: Right: the clusters in the latent representation. Left: their image in the initial representation (better seen in color).

problem is witnessed as clusters are much more mixed than for the SAT12-ALL. For OpenML, the clusters are even more mixed and no structure seems to be shared by the initial and latent representations (with respectively 11 and 10 features).

6.4.2 Quantitative assessment of the initial features

A second possibility is to use the latent representation to evaluate the initial features, as follows. Let us assume that both latent and initial representations are consistent, in the sense that there exists a differentiable mapping ϕ from the latent onto the initial space, mapping the latent representation $\mathbf{z}_i$ of every i-th problem instance onto its initial representation $\mathbf{x}_i$. Let us consider the neighborhood of some $\mathbf{z}_i$ and let us approximate $\mathbf{z}_i$ by a linear combination of its nearest neighbors $\mathbf{z}_{i,1}, \ldots \mathbf{z}_{i,r}$:

$$\mathbf{z}_i = \sum_{j=1}^{r} w_{i,j} \mathbf{z}_{i,j}$$

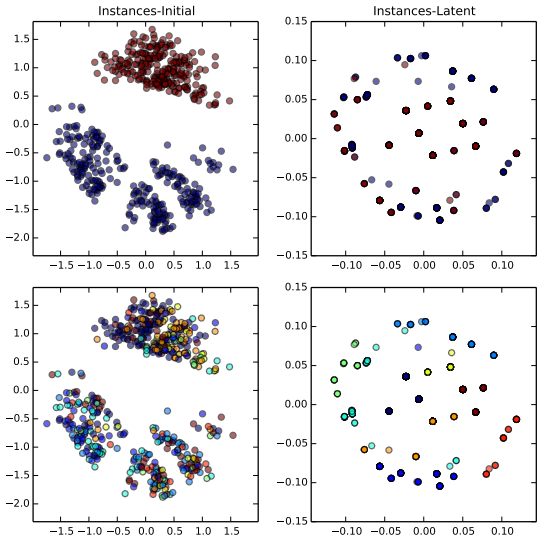


Figure 19: The initial and latent representations of the problem instances in the PREMARSHALLING-ASTAR-2015 dataset. Top row: Left: the clusters in the initial representation. Right: their image in the latent representation. Bottow row: Right: the clusters in the latent representation. Left: their image in the initial representation (better seen in color)..

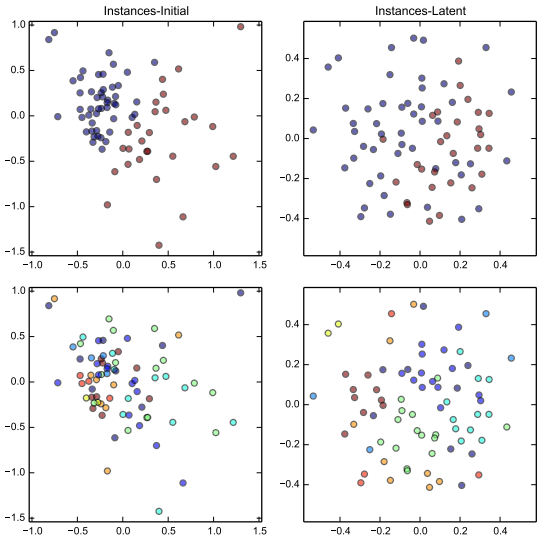


Figure 20: The initial and latent representations of the problem instances in the OpenML dataset. Top row: Left: the clusters in the initial representation. Right: their image in the latent representation. Bottow row: Right: the clusters in the latent representation. Left: their image in the initial representation (better seen in color).

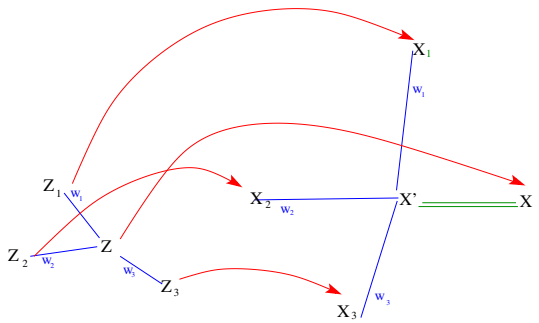


Figure 21: Mapping the latent representation ($\mathbf{z}$, $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3$) onto the initial representation ($\mathbf{x}$, $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$), and measuring its local distortion between $\mathbf{x}$ (the image of $\mathbf{z} = \sum_i w_i \mathbf{z}_i$) and $\mathbf{x}$' $= \sum_i w_i \mathbf{x}_i$), sum of the weighted images of the $\mathbf{z}_i$.

Under the assumptions done, ϕ is locally linear in the neighborhood of $\mathbf{z}_i$, yielding:

$$\phi(\mathbf{z}_i) = \phi(\sum_{j=1}^{r} w_{i,j} \mathbf{z}_{i,j}) \approx \sum_{j=1}^{r} w_{i,j} \phi(\mathbf{z}_{i,j}) = \sum_{j=1}^{r} w_{i,j} \mathbf{x}_{i,j}$$

On the other hand, $\phi(\mathbf{z}_i) = \mathbf{x}_i$ by construction. The consistency of the two representations, that is, the existence of such a ϕ, can therefore be checked by inspecting the vector $\mathbf{x}_i - \sum_{j=1}^{r} w_{i,j} \mathbf{x}_{i,j}$ (Fig. 21).

Furthermore, the consistency can be inspected independently *for each initial feature*: the distortion of the ℓ-th feature in the neighborhood of $\mathbf{z}_i$ is the absolute value of the ℓ-th coordinate of $\mathbf{x}_i - \sum_{j=1}^{r} w_{i,j} \mathbf{x}_{i,j}$. Noting e_ℓ the unit vector with a 1 on the ℓ coordinate,

$$\text{Distortion } (\ell, \mathbf{x}_i) = |\langle \mathbf{x}_i - \sum_{j=1}^{r} w_{i,j} \mathbf{x}_{i,j}, e_\ell \rangle|$$

Fig. 22 depicts the distortion of the OpenML features (preliminarily normalized in $[0, 1]$), reporting for each feature the $n = 76$ distortion values (ordered by increasing value for readability). The distortion due to the feature *number of numerical attributes*, for instance, appears to be much smaller than the distortion due to the *number of categorical attributes*.

7 Discussion and Perspectives

The original contribution of this paper, the ALORS method, provides a sound methodology to tackle algorithm selection, and to analyze where the potential difficulties come from. A first possible source of errors is the insufficient representativity of the problem instances w.r.t. the algorithm portfolio. This error is

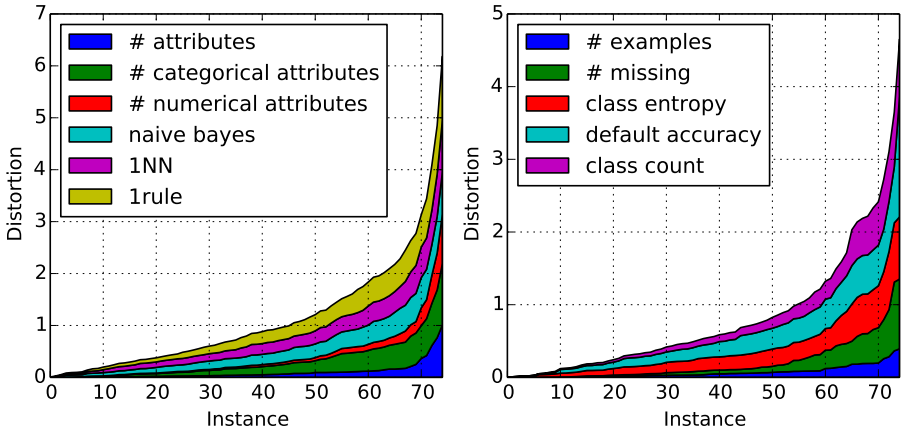


Figure 22: Distortion of the OpenML features over the 76 problem instances. Stacked plot of the distorsion of the number of attributes, of categorical and symbolic attributes, error of naive Bayes, 1-nearest neighbor and 1 rule (Left plot) and number of examples, number of missing values,class entropy, default accuracy, classcount (Right plot).

diagnosed when the matrix reporting the performance of the algorithms on the problem instances can hardly be reconstructed from an excerpt thereof. Otherwise, the CF matrix yields an accurate latent representation of the problem instances and the algorithms.

A second possible source of errors is the inadequate representation of the problem instances, preventing the learning of the latent representation. A main result of the paper is to provide empirical evidence that both difficulties are distinct: on the OpenML benchmark, excellent Matrix Completion results are seen with poor Cold Start results, suggesting that the initial OpenML features are insufficient. The comparison of the latent and initial representations further provides a computable measure of relevance for each initial feature (section 6.4).

A short-term perspective for further work is to extend ALORS to achieve both algorithm selection and configuration, selecting the best suited algorithm and the optimal hyper-parameters for a problem instance. A natural idea is to consider each algorithm-configuration as an algorithm with a (varying length) feature description, its hyper-parameter setting. Some care must be taken, though, in order to limit the sparsity of the collaborative matrix when dealing with continuous hyper-parameters. Another perspective is to extend ALORS to build pre-schedulers. A natural possibility is to exploit the fact that ALORS provides a per-instance ranking of the algorithms. Pre-schedulers could then be derived by allocating to the top-ranked algorithms a given part of the computational budget; as these top-ranked algorithms depend on the current instance, one would thus get a per-instance pre-scheduler.

Our main research perspective is to support the search for the design of initial features. As witnessed by the AS successes in SAT and CSP, the initial features in these domains are quite accurate; but their design required significant manual efforts, and further efforts might be required when the SAT and CSP domains will face new domains. In the ML and KDD domain, the design of initial features also consumed huge manual for over two decades. However the search for efficient features might have more chances of success now, as the comparison between the latent and the initial features gives a direct and independent assessment of each initial feature.

Acknowledgments

The authors wish to thank Holger Hoos, Frank Hutter, Joaquin Vanschoren, Marc Schoenauer, Philippe Caillou, Balazs Kégl, for many discussions and suggestions, and the anonymous reviewers for their many critiques and suggestions that helped considerably to improve the paper. The authors are grateful for the online MATCHBOX available on Azure ML Studio[10].

This work was done while the first author was at the Lab. of Computer Science at Université Paris-Sud, funded by an ERCIM post-doc grant.

References

Agarwal, D., Chen, B.-C., Elango, P., Ramakrishnan, R., 2013. Content recommendation on web portals. Communications of the ACM 56 (6), 92–101.

Ahn, H. J., 2008. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. Information Sciences 178 (1), 37–51.

Bardenet, R., Brendel, M., Kégl, B., Sebag, M., 2013. Collaborative hyperparameter tuning. In: Proceedings of the 30th International Conference on Machine Learning (ICML). Vol. 28. pp. 199–207.

Bennett, J., Lanning, S., 2007. The netflix prize. In: Proceedings of of the 13th International Conference on Knowledge Discovery and Data Mining (KDD) Cup and Workshop. p. 35.

Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B., 2011. Algorithms for hyper-parameter optimization. In: J. Shawe-Taylor, R.S. Zemel, P. B. F. P. K. W. (Ed.), Proceedings of the 25th Annual Conference on Neural Information Processing Systems (NIPS). Vol. 24 of Advances in Neural Information Processing Systems. Granada, Spain.

Bischl, B., Kerschke, P., Kotthoff, L., Lindauer, M., Malitsky, Y., Frechette, A., Hoos, H., Hutter, F., Leyton-Brown, K., Tierney, K., Vanschoren, J., 2015. Aslib: A benchmark library for algorithm selection.

[10] studio.azureml.net

Bobadilla, J., Ortega, F., Hernando, A., Gutiérrez, A., 2013. Recommender systems survey. Knowledge-Based Systems 46, 109–132.

Borg, I., Groenen, P. J., 2005. Modern multidimensional scaling: Theory and applications. Springer Verlag.

Brazdil, P., Carrier, C. G., Soares, C., Vilalta, R., 2008. Metalearning: applications to data mining. Springer.

Brazdil, P., Soares, C., 2000. A comparison of ranking methods for classification algorithm selection. In: de Mántaras, R. L., Plaza, E. (Eds.), the 11th European Conference on Machine Learning (ECML). Vol. 1810 of LNCS. Springer, pp. 63–74.

Chu, W., Ghahramani, Z., 2005. Extensions of Gaussian processes for ranking: semisupervised and active learning. In: Proceedings of the NIPS 2005 Workshop on Learning to Rank. MIT, pp. 29–34.

Epstein, S. L., Freuder, E. C., Wallace, R., Morozov, A., Samuels, B., 2006. The adaptive constraint engine. In: Hentenryck, P. V. (Ed.), the 8th International Conference on Principles and Practice of Constraint Programming (CP). Vol. 2470 of LNCS. pp. 525–540.

Gagliolo, M., Schmidhuber, J., 2011. Algorithm portfolio selection as a bandit problem with unbounded losses. Annals of Mathematics and Artificial Intelligence 61 (2), 49–86.

Gomes, C. P., Selman, B., 2001. Algorithm portfolios. Artificial Intelligence 126 (1-2), 43–62.

Gunawardana, A., Meek, C., 2008. Tied Boltzmann machines for cold start recommendations. In: Pu, P., Bridge, D. G., Mobasher, B., Ricci, F. (Eds.), Proceedings of the ACM conference on Recommender Systems (RecSys). ACM, pp. 19–26.

Hartigan, J., 1975. Clustering algorithms. John Wiley and Sons.

Hutter, F., Hoos, H. H., Leyton-Brown, K., 2011. Sequential model-based optimization for general algorithm configuration. In: Coello, C. A. C. (Ed.), the 5th Learning and Intelligent Optimization Conference (LION). Vol. 6683 of LNCS. Springer, pp. 507–523.

Hutter, F., Hoos, H. H., Leyton-Brown, K., Stützle, T., 2009. ParamILS: an automatic algorithm configuration framework. Journal of Artificial Intelligence Research 36 (1), 267–306.

Järvelin, K., Kekäläinen, J., 2000. IR evaluation methods for retrieving highly relevant documents. In: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, pp. 41–48.

Kadioglu, S., Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M., 2011. Algorithm selection and scheduling. In: the 17th International Conference on Principles and Practice of Constraint Programming (CP). Vol. 6876 of LNCS. Springer, pp. 454–469.

Kadioglu, S., Malitsky, Y., Sellmann, M., Tierney, K., 2010. ISAC - instance-specific algorithm configuration. In: Proceedings of the 19th European Conference on Artificial Intelligence (ECAI). Vol. 215 of Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 751–756.

Kotthoff, L., 2012. Hybrid regression-classification models for algorithm selection. In: Raedt, L. D., Bessière, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, P. J. F. (Eds.), Proceedings of the 20th European Conference on Artificial Intelligence (ECAI). Vol. 242 of Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 480–485.

Kotthoff, L., 2014. Algorithm selection for combinatorial search problems: A survey. AI Magazine 35 (3), 48–60.

Leyton-Brown, K., Nudelman, E., Andrew, G., McFadden, J., Shoham, Y., 2003. A portfolio approach to algorithm selection. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI). Acapulco, Mexico, pp. 1542–1542.

Lindauer, M. T., Hoos, H. H., Hutter, F., Schaub, T., 2015. Autofolio: An automatically configured algorithm selector. J. Artif. Intell. Res. (JAIR) 53, 745–778.

Liu, J.-H., Zhou, T., Zhang, Z.-K., Yang, Z., Liu, C., Li, W.-M., 2014. Promoting cold-start items in recommender systems. PloS one 9 (12), e113457.

Malitsky, Y., O´ Sullivan, B., 2014. Latent features for algorithm selection. In: Proceedings of the Seventh Annual Symposium on Combinatorial Search, SOCS.

Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M., 2013. Algorithm portfolios based on cost-sensitive hierarchical clustering. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI). AAAI Press, pp. 608–614.

Mısır, M., Sebag, M., 2013. Algorithm selection as a collaborative filtering problem. Tech. rep., INRIA-Saclay.
URL http://hal.inria.fr/hal-00922840

Nguyen, P., Hilario, M., Kalousis, A., 2014. Using meta-mining to support data mining workflow planning and optimization. J. Artif. Intell. Res. (JAIR) 51, 605–644.

Nikoli, M., Mari, F., Janii, P., 2013. Simple algorithm portfolio for SAT. Artificial Intelligence Review 40 (4), 457–465.

O´ Mahony, E., Hebrard, E., Holland, A., Nugent, C., O´ Sullivan, B., 2008. Using case-based reasoning in an algorithm portfolio for constraint solving. In: Irish Conference on Artificial Intelligence and Cognitive Science. pp. 210–216.

Oentaryo, R. J., Handoko, S. D., Lau, H. C., 2015. Algorithm selection via ranking. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI).

Park, S.-T., Chu, W., 2009. Pairwise preference regression for cold-start recommendation. In: Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys). ACM, pp. 21–28.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Matthieu, B., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in python. The Journal of Machine Learning Research 12, 2825–2830.

Pulina, L., Tacchella, A., 2010. Aqme'10. JSAT 7 (2-3), 65–70.
URL http://jsat.ewi.tudelft.nl/content/volume7/JSAT7_5_Pulina.pdf

Rice, J., 1976. The algorithm selection problem. Advances in computers 15, 65–118.

Rousseeuw, P. J., 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Computational and Applied Mathematics 20.

Salakhutdinov, R., Mnih, A., 2008. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In: Proceedings of the 25th International Conference on Machine Learning (ICML). pp. 880–887.

Samulowitz, H., Memisevic, R., 2007. Learning to solve QBF. In: Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI). Vol. 22. pp. 255–360.

Schein, A. I., Popescul, A., Ungar, L. H., Pennock, D. M., 2002. Methods and metrics for cold-start recommendations. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Tampere, Finland, pp. 253–260.

Stern, D. H., Herbrich, R., Graepel, T., 2009. Matchbox: large scale online Bayesian recommendations. In: Quemada, J., León, G., Maarek, Y. S., Nejdl, W. (Eds.), Proceedings of the 18th International Conference on World Wide Web (WWW). ACM, pp. 111–120.

Stern, D. H., Samulowitz, H., Herbrich, R., Graepel, T., Pulina, L., Tacchella, A., 2010. Collaborative expert portfolio management. In: Fox, M., Poole, D. (Eds.), Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI). AAAI Press.

Strang, G., 1980. Linear Algebra and its Applications. Academic Press, New York.

Streeter, M., Smith, S., 2008. New techniques for algorithm portfolio design. In: Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI). Vol. 10. p. 26.

Su, X., Khoshgoftaar, T., 2009. A survey of collaborative filtering techniques. Advances in Artificial Intelligence 2009, 19.

Sun, Q., Pfahringer, B., 2013. Pairwise meta-rules for better meta-learning-based algorithm ranking. Machine Learning 93 (1), 141–161.
URL http://dx.doi.org/10.1007/s10994-013-5387-y

Teo, C. H., Smola, A. J., Vishwanathan, S. V. N., Le, Q. V., 2007. A scalable modular convex solver for regularized risk minimization. In: Berkhin, P., Caruana, R., Wu, X. (Eds.), Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 727–736.

Thornton, C., Hutter, F., Hoos, H. H., Leyton-Brown, K., 2013. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). ACM, New York, NY, USA, pp. 847–855.

van Rijn, J., Bischl, B., Torgo, L., Gao, B., Umaashankar, V., Fischer, S., Winter, P., Wiswedel, B., Berthold, M., Vanschoren, J., 2013. OpenML: A collaborative science platform. In: Blockeel, H., Kersting, K., Nijssen, S., elezn, F. (Eds.), Proceedings of the Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD Part III. Vol. 8190 of LNCS. Springer, pp. 645–649.

Weimer, M., Karatzoglou, A., Le, Q., Smola, A., 2007. CofiRank-maximum margin matrix factorization for collaborative ranking. In: Proceedings of the 21st Annual Conference on Neural Information Processing Systems (NIPS). pp. 222–230.

Weston, J., Wang, C., Weiss, R. J., Berenzweig, A., 2012. Latent collaborative retrieval. In: Proceedings of the 29th International Conference on Machine Learning (ICML). Edinburgh, Scotland, UK.

Wolpert, D., Macready, W., 1997. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1 (1), 67–82.

Xu, L., Hutter, F., Hoos, H., Leyton-Brown, K., 2008. SATzilla: portfolio-based algorithm selection for SAT. Journal of Artificial Intelligence Research 32 (1), 565–606.

Xu, L., Hutter, F., Hoos, H., Leyton-Brown, K., 2012a. Evaluating component solver contributions to portfolio-based algorithm selectors. In: Cimatti, A., Sebastiani, R. (Eds.), the 15th International Conference on Theory and Applications of Satisfiability Testing (SAT). Vol. 7317 of LNCS. Springer, Trento, Italy, pp. 228–241.

Xu, L., Hutter, F., Hoos, H., Leyton-Brown, K., 2012b. Features for SAT, university of British Columbia, http://www.cs.ubc.ca/labs/beta/Projects/SATzilla/Report_SAT_features.pdf.

Xu, L., Hutter, F., Shen, J., Hoos, H., Leyton-Brown, K., 2012c. Satzilla2012: Improved algorithm selection based on cost-sensitive classification models. In: Proceedings of SAT Challenge 2012: Solver and Benchmark Descriptions. pp. 57–58.

Zhou, M., Wang, C., Chen, M., Paisley, J., Dunson, D., Carin, L., 2010. Nonparametric Bayesian matrix completion. In: Proceedings of the IEEE Sensor Array and Multichannel Signal Processing Workshop.