Data Sampling through Collaborative Filtering for Algorithm Selection

Mustafa Mısır
College of Computer Science and Technology
Nanjing University of Aeronautics and Astronautics
Nanjing, Jiangsu, China
Email: mmisir@nuaa.edu.cn

***Abstract*—Algorithm selection has been studied to specify the best possible algorithm(s) for a given problem instance. One of the major drawbacks of the algorithm selection methods is their need for the performance data. The performance data involves the performance of a set of algorithms on a group of problem instances. Depending on the problem domain, algorithms and the experimental settings, generating such data can be computationally expensive. ALORS [1] as a collaborative filtering based algorithm selection strategy addresses this issue by performing matrix completion. Matrix completion allows to generate algorithm selection models when the performance data is incomplete. Although ALORS is able to deal with varying data incompleteness levels, it ignores the quality and cost of the performance data. The present study offers a collaborative filtering based sampling strategy to designate which algorithm(s) to run on which instance(s). The goal is to provide either computationally cheap or highly informative incomplete performance data for algorithm selection.**

I. Introduction

Algorithm selection [2] is known as an automated concept to choose the best algorithm(s) for solving a given problem instance. A traditional algorithm selection approach [3] is concerned with generating performance prediction models that can tell the performance of a given set of algorithms on a problem instance. In order to generate this type of models, data regarding the performance of these algorithms on a suite of training instances is required. Besides that, a set of features that can effectively characterize the target problem's instances is essential. Then, an algorithm selection model can simply map these instance features to the algorithms' performance. This approach is practical since it is known that there is no one single algorithm works well on every problem instance, referring to the no free lunch theorem [4]. From this perspective, algorithm selection has a high potential of outperforming given problem specific algorithms. Various studies [5], [6] already showed such problem specific algorithms can be surpassed by algorithm selection. In particular, the SAT competitions[1] are the most well-known venues indicating the way to outperform the SAT solvers designed by the true problem experts, through algorithm selection.

The algorithm-instance performance data plays a critical role on the success of algorithm selection. Thus, it is crucial to have the performance data at a certain quality to realize its anticipated behavior. However, depending on the problem domain and the algorithms accommodated for this problem, generating such performance data could be extremely costly even when the quality is being assured. For instance, the time required to generate the datasets in the algorithm selection benchmark library (ASlib)[2] [7] varies from ∼25 to ∼600 CPU days[3]. Primarily considering this cost issue, a collaborative filtering [8] based algorithm selection technique, i.e. ALORS [9], [1], was introduced. ALORS has the capability of performing algorithm selection effectively when incomplete performance data is present. This capability comes from the use of the matrix completion techniques [10] relating to collaborative filtering. ALORS showed that it can outperform the best single algorithms on varying problems with up to the 90% incompleteness. In the context of dealing with incomplete data, the motivation of ALORS is to use existing incomplete performance data for algorithm selection. Thus, ALORS can work with whatever performance data is available. However, if a new problem is being targeted with algorithm selection, it would be practical to determine how to produce this incomplete performance data in the first place.

The present study focuses on a data sampling strategy using collaborative filtering to decide on running which algorithm on which instance. The goal is to produce incomplete performance data for either reducing the time required to generate it or delivering better performance than random sampling (Random). The proposed sampling strategy– dubbed MC, accommodates matrix completion for sampling before applying matrix completion for actually completing a given incomplete matrix. The matrix completion method studied in ALORS is used for both the sampling and completion tasks. MC is initially applied to predict the performance entries which are likely to be practical for better matrix completion. Depending on the MC's predictions, a certain number of entries are sampled by running an algorithm on a problem instance referring to each entry. Then, the same MC is applied to complete the given incomplete performance matrix with additional entries came from sampling. For evaluating MC against Random, the aforementioned ASlib is utilized as the

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

[2] http://aslib.net

[3] It should be noted that the underlying computational environments aren't the same for all the datasets

978-1-5090-4601-0/17/$31.00

testbed. The ASlib version used here is composed of a set of algorithm selection datasets from the problem domains of Boolean Satisfiability, Constraint Satisfaction, Quantified Boolean Formula, Answer-set Programming and Container Pre-marshalling.

The computational results revealed that MC is proficient to pick algorithm-instance samples that can help to achieve the intended objectives for the most of the ASlib datasets, especially the ones with a limited number of algorithms.

The remainder of the paper is presented as follows. Some background knowledge both on algorithm selection and collaborative filtering is provided in Section II. Section III introduces the proposed sampling strategy, MC. Section IV delivers an experimental analysis on matrix completion together with a comparison against Random. A brief discussion alongside with the related future research plans is provided in Section V.

II. Background

A. Algorithm Selection

Algorithm selection[4] has been productively applied to different problem domains such as Boolean Satisfiability (SAT) [6], Constraint Satisfaction [11], Traveling Salesman [12], Traveling Thief [13], Black-box Optimization [14], Machine Learning [15], [16] and Game Playing [17]. The traditional algorithm selection techniques operate in an offline manner by choosing the algorithm(s) to be applied before solving a given problem instance. Studies on algorithm selection from different perspectives are available in the literature. SATZilla [5], as one of the well-known algorithm selection methods due to its success on the SAT competitions[5], involves multiple strategies to perform effective algorithm selection. It applies pre-solvers for the easy instances, i.e. requiring short time to be solved, a back-up solver for the hard instances and incorporates cost-sensitive hierarchical clustering with its follow-up version [6], as in [18]. Hydra [19] was introduced as an offline algorithm selection strategy requiring a pre-processing step for generating an algorithm portfolio by parameter tuning. 3S [20] was developed to offer algorithm schedules as the use of the resource constrained set covering problem with column generation. In [14], the exploratory landscape analysis was investigated by utilizing fitness landscape features for algorithm selection. Algorithm selection was utilized to discover the best classifiers on data streams in [21]. ASAP [22] was introduced as an algorithm selection system incorporating a pre-scheduler that runs a number of algorithms for a short period of time prior to applying algorithm selection. As a higher level approach to algorithm selection, AutoFolio [23] was proposed to perform algorithm configuration on algorithm selection to come up with the best algorithm selection setting.

Although the term algorithm selection has been commonly referred to the offline techniques, online algorithm selection methods are also present. In the online case, the goal is to choose algorithms on-the-fly, while an instance is being solved. Selection hyper-heuristics[6] [24], [25] performing online fall into this category. Adaptive operator selection [26] has also been used in the literature for the purpose of online selection.

In addition to the solely running offline and online methods, hybrid algorithms have been studied. OSCAR [27] was designed as a hybrid strategy combining offline algorithm selection in the form of algorithm portfolios [28] and online algorithm selection. Similarly, in [29], a hybrid approach combining parameter tuning for generating portfolios to be used in an online algorithm selection setting was introduced. A similar portfolio generation system was released as a web-based, cloud like, service called ADVISER+[7] [30], [31].

B. Collaborative Filtering

Collaborative filtering [8], [32], [33] is a field of recommender systems [34], popularized with the Netflix challenge[8] [35]. The goal is suggesting particular types of items such as movies [36], [37], books [38] and articles [39] or items of different types [40], to users depending on their preferences compared to others. In other words, if two users share similar preferences on the commonly experienced items, their preferences on unseen items are expected to be similar, vice versa. As the other major recommender systems idea, the content-based filtering approaches can offer items solely depending on the personal details or preferences, e.g. age, nationality, interest in history etc. which might not be always present. Thus, collaborative filtering can deliver more diverse and unexpected recommendations, e.g. suggest horror movies to a user who have never had preference on horror movies, while content-based filtering is practical to offer expected type of items, e.g. suggest an action movie to a user who is highly interested in action movies already. In this respect, collaborative filtering is more like asking others' opinions on a certain category of items, to whom shares similar preferences. Although the focus is users for collaborative filtering, in terms of the collaborative filtering algorithms, it is possible to see methods focusing on items rather than users [41]. Also, the content-based and collaborative filtering methods can be hybridized [42], [43] for better performance than each approach alone.

The main problem to be tackled by collaborative filtering is matrix completion [44] which is the focus of this paper. The idea behind completion is predict the users' preferences when their preferences are partially known. As its name suggests, matrix completion is about filling given incomplete user-item data. This could be achieved either with the memory-based or model-based approaches. The memory-based approaches predict missing entries directly depending on the available user-item data. The model-based methods build models which can do predictions on the missing entries. The generated models are usually expected to be better than the memory-based approaches for high incompleteness. Yet, it is possible

[4] An algorithm selection bibliography: http://larskotthoff.github.io/assurvey/
[5] http://www.satcompetition.org/
[6] A hyper-heuristic bibliography: https://mustafamisir.github.io/hh.html
[7] http://research.larc.smu.edu.sg/adviserplus/
[8] http://www.netflixprize.com

to provide state-of-the-art results with the memory-based approaches for decreasing incompleteness. Further improvements on both ideas could be achieved with their combinations [45]. Another critical collaborative filtering problem is cold start [46]. The objective of this problem is to infer the new users' item preferences or preferences of the existing users on a new item.

III. Method

Figure 1 shows an example incomplete rank matrix, $\mathcal{M}$. Each row of the matrix relates to a problem instance while each column represents an algorithm. Thus, $\mathcal{M}_{i,j}$ involves the rank of algorithm j on instance i. ALORS [1] utilizes two strategies of model-based and memory-based methods to complete such matrices. In the case of performance metric is different from rank, ALORS requires the data to be converted into a rank form. The model-based approach is basically a rank-based matrix factorization strategy [47]. Although it is effective for very low incompleteness levels, it is computationally expensive, compared to the memory-based approach. Besides that with decreasing incompleteness levels, the memory-based approach performs superior to the model-based one. Thus, as also in ALORS, in this study, the same memory-based strategy is employed.

The memory-based approach operates based on the instance similarities depending the available common entries between the pairs of problem instances. The instances' similarities, $sim(i,\ell)$, are assessed using the cosine similarity metric which is detailed as follows:

$$sim(i,\ell) = \frac{\sum_{k \in I_{i,\ell}} \mathcal{M}_{i,k} \,.\, \mathcal{M}_{\ell,k}}{\sqrt{\sum_{k \in I_{i,\ell}} \mathcal{M}_{i,k}^2} \cdot \sqrt{\sum_{k \in I_{i,\ell}} \mathcal{M}_{\ell,k}^2}} \tag{1}$$

where $I(i,\ell)$ is the set of common indices between $\mathcal{M}_{i,j}$ and $\mathcal{M}_{\ell,j}$. Then, the calculated similarities are used to predict the missing entries through a similarity-based weighted approach [39] as in:

$$\widehat{\mathcal{M}_{i,j}} = \overline{\mathcal{M}}_i + \frac{\sum_{\ell \ s.t. \ j \in I_\ell} sim(i,\ell)(\mathcal{M}_{\ell,j} - \overline{\mathcal{M}}_\ell)}{\sum_{\ell \ s.t. \ j \in I_\ell} sim(i,\ell)} \tag{2}$$

This study employs the exact same matrix completion approach to decide on which algorithm j to run on which problem instance i. Yet, any other matrix completion method θ can be utilized as pointed out in Algorithm 1. The algorithm starts with a matrix completion phase on a given incomplete performance data $\mathcal{M}$. To be able to apply matrix completion, it is assumed that 10% of the complete data is available, as exemplified in Figure 1. The completion is used to detect the best ranks among the unknown algorithm-instance entries. The motivation here is to keep the expectedly best entries that can help to deliver strong performance on the actual completion. While offering high completion success, it is planned to degrade the time required to generate the incomplete performance data if the algorithms' runtime behavior affects their performance, i.e. shorter runs mean better algorithms.

After completing a given incomplete matrix $\mathcal{M}$, the best expected at most n algorithms J for each instance i are determined. For all the selected entries to be sampled, the corresponding algorithms run on the specified instances. The number of entries to be added for each instance is equally distributed, depending on the total number of samples requested N.

Algorithm 1: Collaborative filtering based sampling

Input: An incomplete performance matrix $\mathcal{M}$, Matrix completion algorithm θ, Number samples to be added: N

Matrix Completion: $\theta(\mathcal{M}) \rightarrow \widehat{\mathcal{M}}$
Per Instance Sample Size: $n = N/|I|$
foreach $\widehat{\mathcal{M}}_i$ *where* $i \in I$ **do**
  $\arg\min_J (\widehat{\mathcal{M}}_i, n)$ where $\widehat{\mathcal{M}}_{i,j} \neq 0$, $|J| \leq n$
  foreach $j \in J$ **do**
    $\mathcal{M}_{i,j} \leftarrow$ Run algorithm j on instance i
  end
end

IV. Computational Results

A series of experiments is performed on the algorithm selection library (ASlib) as mentioned earlier. Table I indicates 13 benchmark datasets used. Each dataset is composed of a set of algorithms, a group of problem instances and a suite of instance features. The instance features are required for the cold start problem. Since the target of this study is to address the matrix completion problem, the instance features are ignored. Additionally, the time required to generate each dataset is mentioned.

TABLE I
The ASlib benchmarks, with the number of CPU days required to generate

Dataset	#Instances	#Algorithms	#CPU
ASP-POTASSCO	1294	11	25
CSP-2010	2024	2	52
MAXSAT12-PMS	876	6	56
PREMARSHALLING -ASTAR-2015	527	4	28
PROTEUS-2014	4021	22	596
QBF-2011	1368	5	163
SAT11-HAND	296	15	168
SAT11-INDU	300	18	128
SAT11-RAND	600	9	158
SAT12-ALL	1614	31	415
SAT12-HAND	767	31	234
SAT12-INDU	1167	31	284
SAT12-RAND	1362	31	447

These datasets utilize the algorithms' runtime information as the performance metric. They are used after the unsolvable instances, for the given algorithms within the given time limit, are removed. Besides that, initially, the runtime values are converted to ranks as mentioned earlier.

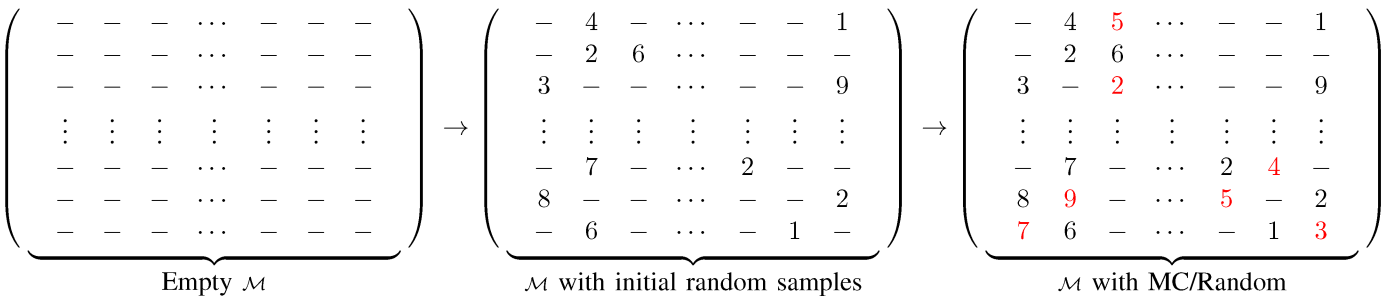

Fig. 1. A data sampling example for algorithm selection performance data $\mathcal{M}$ ($-$ indicates the missing entries)

The proposed approach is tested along with the ALORS using Random. The comparison is made for varying incompleteness levels from 10% to 90%. It is assumed that 10% of the performance data is initially available to be able to apply matrix completion. For ALORS, each incompleteness levels are tested with 10 randomly selected entries. The exact same 10% performance data with ALORS is used for MC and Random. However, after the first 10% initial data, 10% performance data is gradually added through a sampling strategy, either MC or Random.

$$\frac{t - t_{best}}{t_{worst} - t_{best}} \tag{3}$$

A comparison is presented in terms of the time required to generate incomplete performance data and average algorithm ranks. The time information is normalized using Equation 3 to able to observe how far the incomplete data is to the optimum sampling. t is the time required to generate the performance data. t_{best} refers to the case where only the best entries, i.e. the optimum case, algorithm-instance pairs, are selected while t_{worst} is for the case the worst entries, i.e. the worst case, are picked.

Figure 2 presents the results in terms of the cost of generating these datasets, Equation 3. Since the ASlib datasets used are based on run-time, choosing the best entries also mean that using the least amount of time to produce the performance data. For 9 out of 13 ASlib datasets, the time to produce the performance data is decreased. The highest gain is provided on the SAT12-ALL, SAT12-HAND, SAT12-INDU and SAT12-RAND benchmark datasets. These datasets have the same set of algorithms and the algorithm set is the largest one with 31 algorithms among all the benchmark datasets. For ASP-POTASSCO with a comparatively small algorithm set of 11 algorithms, substantial time improvement is achieved. The remaining 4 datasets (CSP-2010, MAXSAT12-PMS, PREMARSHALLING-ASTAR-2015, QBF-2011) where the time is mostly increased, involve a limited number of algorithms, i.e. 2, 6, 4 and 5 respectively. These results indicate that the number of algorithms can affect the matrix completion performance. This outcome is expected due to the nature of the matrix completion method employed. Having a rather small algorithm set makes the instance similarity evaluation challenging. Thus, the completion results can be misleading meaning that predicting the poor performing algorithms as the highly effective ones. One possible solution for this issue would be offering a matrix completion method through instance similarity. Despite the poor performance of the used matrix completion strategy on these datasets, the resulting performance can also be considered an indicator of the need for improving the existing incomplete data.

Table II shows the average time gain achieved in CPU hours when 10% performance data is sampled for varying incompleteness levels. As pointed out earlier, while for most of the cases time significant time improvement is delivered, worse time performance of the sampling method against Random can be seen. The highest time gain is provided with 1301 hours (∼54 days) for PROTEUS-2014 when 20% of the performance data is present. The worst performance is also revealed on the same dataset when 80% data is available. The time difference between MC and Random is reached to 719 hours (∼30 days). Besides PROTEUS-2014, comparable time gain is provided, from 66 hours to 1155 hours (∼48 days), on the SAT12 datasets for all the incompleteness levels.

Figure 3 illustrates the matrix completion performance in terms of average rank. For SAT12-RAND, the performance of Random of pure ALORS at 30% incompleteness is achieved with 70% incompleteness when the proposed strategy is employed. Considering that the time gain is also substantial for SAT12-RAND, the proposed sampling strategy can achieve improvements both in terms of time and performance. Similar results can be seen for SAT12-INDU. For MAXSAT12-PMS, sampling computationally expensive data pays off for the high incompleteness levels. The sampling method significantly outperforms Random for ALORS on the smallest benchmark dataset in terms of the number of algorithms, i.e. CSP-2010, involving only two algorithms. For the remaining datasets, except PREMARSHALLING-ASTAR-2015, SAT11-RAND and SAT11-ALL, the performance of MC is either better or similar compare to Random.

V. Conclusion

The present work introduces a data sampling strategy through collaborative filtering in the algorithm selection setting. Algorithm selection requires performance data generated

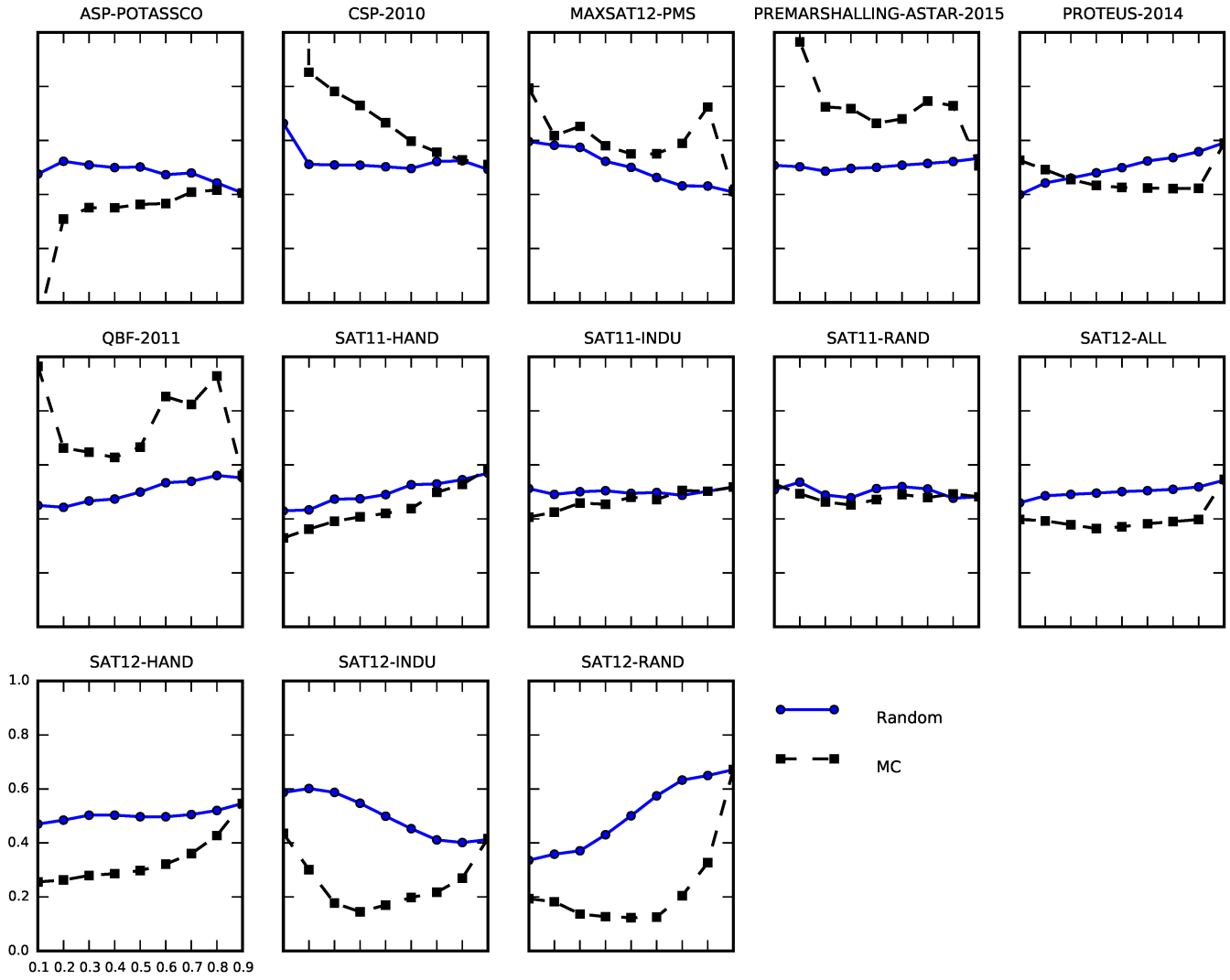

Fig. 2. Normalized time ratio for varying incompleteness levels, $0.1 \rightarrow 0.9$ (0.9 refers to the initial 10% data case)

TABLE II
AVERAGE TIME GAIN ACHIEVED IN HOURS BY MC AGAINST RANDOM (0.9 IS IGNORED SINCE IT IS THE INITIAL RANDOM SAMPLING, USED FOR BOTH METHODS)

Dataset	Incompleteness Levels							
	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**
ASP-POTASSCO	35	31	30	32	31	23	14	4
CSP-2010	-116	-37	-93	-96	-73	-45	-15	-2
MAXSAT12-PMS	-42	-13	-44	-36	-35	-56	-69	-111
PREMARSHALLING-ASTAR-2015	-103	-88	-60	-78	-66	-55	-70	-54
PROTEUS-2014	-719	-457	61	601	978	1239	1301	1270
QBF-2011	-207	-174	-172	-175	-186	-260	-273	-291
SAT11-HAND	31	38	53	48	51	59	20	9
SAT11-INDU	27	28	23	30	9	16	-10	0
SAT11-RAND	-10	37	29	33	51	35	34	-11
SAT12-ALL	83	225	360	462	472	434	386	295
SAT12-HAND	87	156	204	219	208	178	134	66
SAT12-INDU	133	494	902	986	832	629	432	218
SAT12-RAND	134	302	540	775	986	1155	996	560

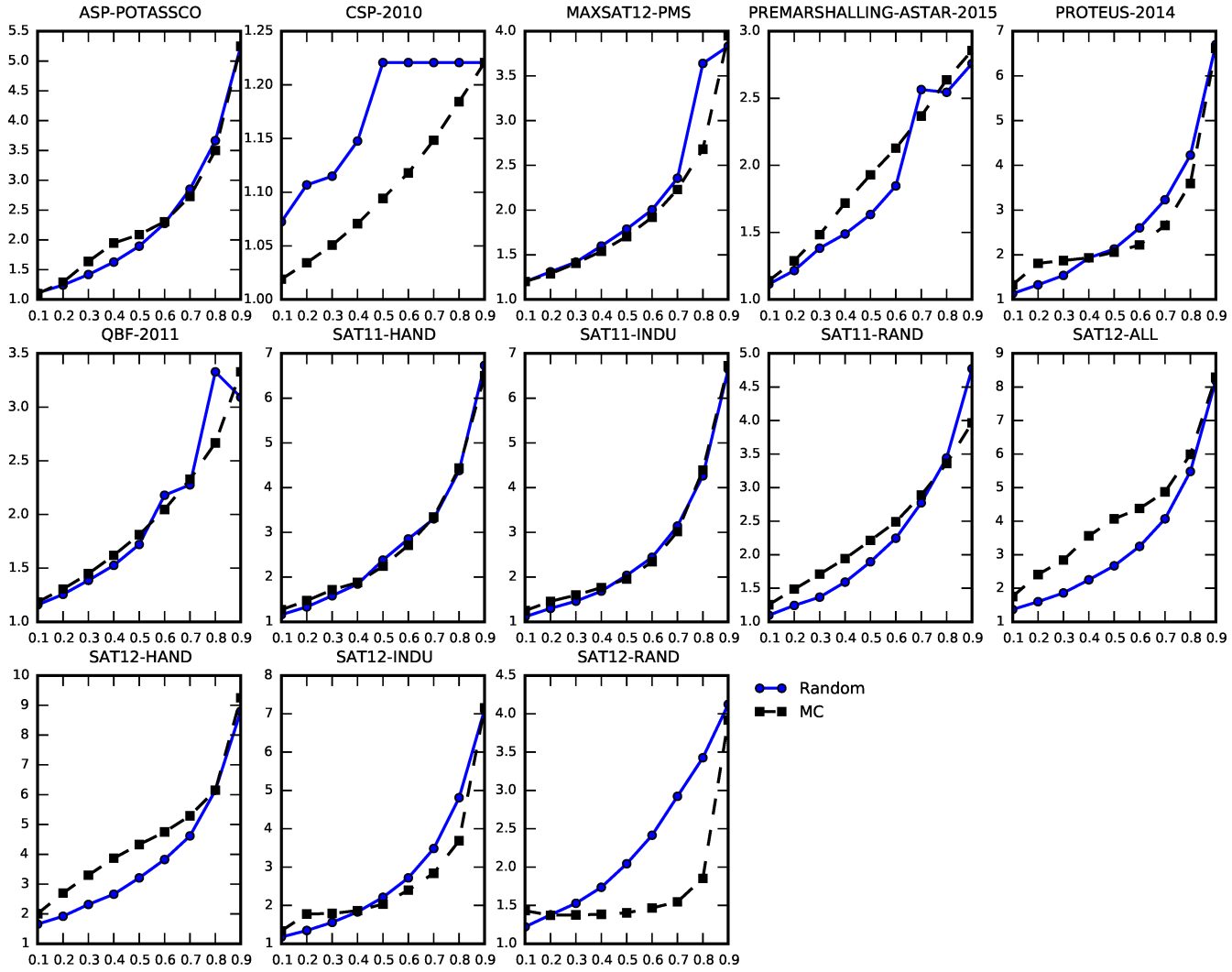

Fig. 3. Rank performance for varying incompleteness levels, $0.1 \rightarrow 0.9$ (0.9 refers to the initial 10% data case)

by running a set of algorithms on a set of problem instances. The performance data of a problem can be used to determine the best algorithm for a given problem instance. Although algorithm selection can give effective decisions on choosing algorithms, generating complete performance data could be costly if it is unavailable. In this respect, to be able to work with incomplete data could be practical. ALORS [1] accommodates collaborative filtering for this purpose. In order to use ALORS more effectively, a collaborative filtering data sampling strategy is proposed. The aim of sampling is to decide on which algorithm to be run on which problem instance for providing limited yet informative incomplete performance data. This approach can become handy for degrading the time needed to produce the performance data while keeping it high quality. The proposed method is tested on the algorithm selection library (ASlib) benchmark datasets to evaluate its performance for matrix completion. The computational results showed that for the majority of the ASlib datasets, either the data is being generated in a shorter period of time or the algorithm selection quality is being improved compared to Random.

As a part of the follow-up research plan, Active Matrix Completion (AMC) [48] will be studied to perform data sampling with active learning [49]. While the proposed strategy requires some initial samples, AMC will be used to start sampling when there is no performance entry yet. The number of performance entries to be sampled will also be automatically specified. Next, multi-objective optimization will be investigated to minimize the cost of generating the performance data while maximizing the success of matrix completion. Additionally, the effects of these strategies using varying algorithm selection performance metrics will be considered. Finally, the contribution of matrix completion to cold start, incorporating instance features, will be examined.

REFERENCES

[1] M. Mısır and M. Sebag, "ALORS: An algorithm recommender system," *Artificial Intelligence*, vol. 244, pp. 291–314, 2017.

[2] L. Kotthoff, "Algorithm selection for combinatorial search problems: A survey," *AI Magazine*, vol. 35, no. 3, pp. 48–60, 2014.

[3] J. Rice, "The algorithm selection problem," *Advances in Computers*, vol. 15, pp. 65–118, 1976.

[4] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, 1997.

[5] L. Xu, F. Hutter, H. Hoos, and K. Leyton-Brown, "SATzilla: portfolio-based algorithm selection for SAT," *Journal of Artificial Intelligence Research*, vol. 32, no. 1, pp. 565–606, 2008.

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

[7] B. Bischl, P. Kerschke, L. Kotthoff, M. Lindauer, Y. Malitsky, A. Fréchette, H. Hoos, F. Hutter, K. Leyton-Brown, K. Tierney *et al.*, "ASlib: A benchmark library for algorithm selection," *Artificial Intelligence*, vol. 237, pp. 41–58, 2017.

[8] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, vol. 2009, p. 4, 2009.

[9] M. Mısır and M. Sebag, "Algorithm selection as a collaborative filtering problem," INRIA, Tech. Rep., 2013.

[10] E. J. Candes and Y. Plan, "Matrix completion with noise," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 925–936, 2010.

[11] Y. Malitsky and B. O'Sullivan, "Latent features for algorithm selection," in *the 7th Annual Symposium on Combinatorial Search*, 2014.

[12] L. Kotthoff, P. Kerschke, H. Hoos, and H. Trautmann, "Improving the state of the art in inexact tsp solving using per-instance algorithm selection," in *Proceedings of the 9th Learning and Intelligent OptimizatioN Conference (LION)*, ser. LNCS, vol. 8994. Springer, 2015, pp. 202–217.

[13] M. Wagner, M. Lindauer, M. Mısır, S. Nallaperuma, and F. Hutter, "A case study of algorithm selection for the traveling thief problem," *arXiv preprint arXiv:1609.00462*, 2016.

[14] B. Bischl, O. Mersmann, H. Trautmann, and M. Preuß, "Algorithm selection based on exploratory landscape analysis and cost-sensitive learning," in *Proceedings of the 14th International Conference on Genetic and Evolutionary Computation (GECCO)*. ACM, 2012, pp. 313–320.

[15] K. Smith-Miles, "Cross-disciplinary perspectives on meta-learning for algorithm selection," *ACM Computing Surveys*, vol. 41, no. 1, pp. 1–25, 2008.

[16] R. B. Prudencio, M. C. De Souto, and T. B. Ludermir, "Selecting machine learning algorithms using the ranking meta-learning approach," in *Meta-Learning in Computational Intelligence*. Springer, 2011, pp. 225–243.

[17] A. Mendes, A. Nealen, and J. Togelius, "Hyperheuristic general video game playing," *Proceedings of the IEEE Computational Intelligence and Games (CIG)*, 2016.

[18] Y. Malitsky, A. Sabharwal, H. Samulowitz, and M. Sellmann, "Algorithm portfolios based on cost-sensitive hierarchical clustering," in *Proceedings of the 23rd International Joint Conference on Artifical Intelligence (IJCAI)*, 2013, pp. 608–614.

[19] L. Xu, H. Hoos, and K. Leyton-Brown, "Hydra: Automatically configuring algorithms for portfolio-based selection," in *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, 2010, pp. 210–216.

[20] S. Kadioglu, Y. Malitsky, A. Sabharwal, H. Samulowitz, and M. Sellmann, "Algorithm selection and scheduling," in *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming (CP)*, ser. LNCS, vol. 6876. Springer, 2011, pp. 454–469.

[21] J. N. van Rijn, G. Holmes, B. Pfahringer, and J. Vanschoren, "Algorithm selection on data streams," in *Discovery Science*. Springer, 2014, pp. 325–336.

[22] F. Gonard, M. Schoenauer, and M. Sebag, "Algorithm selector and prescheduler in the icon challenge," in *International Conference on Metaheuristics and Nature Inspired Computing (META2016)*, 2016.

[23] M. Lindauer, H. H. Hoos, F. Hutter, and T. Schaub, "AutoFolio: an automatically configured algorithm selector," *Journal of Artificial Intelligence Research*, vol. 53, pp. 745–778, 2015.

[24] M. Mısır, "Intelligent hyper-heuristics: A tool for solving generic optimisation problems," Ph.D. dissertation, Department of Computer Science, KU Leuven, 2012.

[25] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013.

[26] L. Da Costa, A. Fialho, M. Schoenauer, and M. Sebag, "Adaptive operator selection with dynamic multi-armed bandits," in *Proceedings of Genetic and Evolutionary Computation Conference (GECCO)*, Atlanta, Georgia, USA, 2008, pp. 913–920.

[27] M. Mısır, D. Handoko, and H. C. Lau, "OSCAR: Online selection of algorithm portfolios with case study on memetic algorithms," in *Proceedings of the 9th Learning and Intelligent OptimizatioN Conference (LION)*, ser. LNCS, vol. 8994, Lille, France, 2015, pp. 59–73.

[28] C. Gomes and B. Selman, "Algorithm portfolios," *Artificial Intelligence*, vol. 126, no. 1, pp. 43–62, 2001.

[29] A. Gunawan, H. C. Lau, and M. Mısır, "Designing and comparing multiple portfolios of parameter configurations for online algorithm selection," in *Proceedings of the 10th Learning and Intelligent OptimizatioN Conference (LION)*, ser. LNCS, vol. 10079, Naples, Italy, 2016, pp. 91–106.

[30] H. C. Lau, M. Mısır, L. Xiang, and J. Lingxiao, "Adviser$^+$: Toward a usable web-based algorithm portfolio deviser," in *Proceeedings of the 12th Metaheuristics International Conference (MIC)*, Barcelona, Spain, 2017.

[31] M. Mısır, D. Handoko, and H. C. Lau, "ADVISER: A web-based algorithm portfolio deviser," in *Proceedings of the 9th Learning and Intelligent OptimizatioN Conference (LION)*, vol. 8994, Lille, France, 2015, pp. 23–28.

[32] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, p. 3, 2014.

[33] S. Wang, M. Gong, L. Ma, Q. Cai, and L. Jiao, "Decomposition based multiobjective evolutionary algorithm for collaborative filtering recommender systems," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 672–679.

[34] S. Oliveira, V. Diniz, A. Lacerda, and G. L. Pappa, "Evolutionary rank aggregation for recommender systems," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2016, pp. 255–262.

[35] J. Bennett, S. Lanning *et al.*, "The netflix prize," in *Proceedings of KDD cup and workshop*, vol. 2007. New York, NY, USA, 2007, p. 35.

[36] I. A. Almosallam and Y. Shang, "A new adaptive framework for collaborative filtering prediction," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2008, pp. 2725–2733.

[37] L. Gu, P. Yang, and Y. Dong, "An dynamic-weighted collaborative filtering approach to address sparsity and adaptivity issues," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 3044–3050.

[38] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, vol. 7, no. 1, pp. 76–80, 2003.

[39] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: an open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. ACM, 1994, pp. 175–186.

[40] B. Li, Q. Yang, and X. Xue, "Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction." in *IJCAI*, vol. 9, 2009, pp. 2052–2057.

[41] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 285–295.

[42] P. Melville, R. J. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," in *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI)*, 2002, pp. 187–192.

[43] L. Yao, Q. Z. Sheng, A. Segev, and J. Yu, "Recommending web services via combining collaborative filtering with content-based features," in *Web Services (ICWS), 2013 IEEE 20th International Conference on*. IEEE, 2013, pp. 42–49.

[44] E. Candes and B. Recht, "Exact matrix completion via convex optimization," *Communications of the ACM*, vol. 55, no. 6, pp. 111–119, 2012.

[45] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles, "Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach," in *Proceedings of the 16th conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 2000, pp. 473–480.

[46] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2002, pp. 253–260.

[47] M. Weimer, A. Karatzoglou, Q. V. Le, and A. Smola, "CofiRank – maximum margin matrix factorization for collaborative ranking," in *Proceedings of the 21st Annual Conference on Neural Information Processing Systems (NIPS)*, 2007, pp. 222–230.

[48] S. Chakraborty, J. Zhou, V. Balasubramanian, S. Panchanathan, I. Davidson, and J. Ye, "Active matrix completion," in *2013 IEEE 13th International Conference on Data Mining*. IEEE, 2013, pp. 81–90.

[49] B. Settles, "Active learning literature survey," *University of Wisconsin, Madison*, vol. 52, no. 55-66, p. 11, 2010.