Matrix Factorization based Benchmark Set Analysis: A Case Study on HyFlex

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

***Abstract*—The present paper offers an analysis strategy to examine benchmark sets of combinatorial search problems. Experimental analysis has been widely used to compare a set of algorithms on a group of instances from such problem domains. These studies mostly focus on the algorithms' performance rather than the quality of the target benchmark set. In relation to that, the insights about the algorithms' varying performance happen to be highly limited. The goal here is to introduce a benchmark set analysis strategy that can tell the quality of a benchmark set while allowing to retrieve some insights regarding the algorithms' performance. A matrix factorization based strategy is utilized for this purpose. A Hyper-heuristic framework, i.e. HyFlex, involving 6 problem domains is accommodated as the testbed to perform the analysis on.**

I. Introduction

Experimental studies are common in research mainly in order to exhibit the superior performance of a new algorithm on a particular problem domain, mostly using a set of well-known, widely-used benchmarks. They are also helpful to present comparative analysis to further understand strengths and weaknesses of multiple algorithms. Although such studies can give useful insights on the behavior of the tested algorithms, they usually lack of characterizing the target problem instances. It might even be the case that the target instances favor a certain type of algorithms. The diversity of the instances plays a critical role in such studies or in experimental algorithm competitions. The diversity level identifies the fairness of a comparison or a competition. Otherwise, an algorithm which performs well on a special type of instances can be considered state-of-the-art if most of the instances is similar to that type. Thus, both for experimental analysis in general and the competitions, a scientific approach should be followed to indicate how fair a benchmark set is.

One way to address this issue is to focus on a set of features characterizing the problem instances. However, if the proper features are not chosen, the resulting analysis can be misleading. Another way would be automatically extracting some features that are expected to be suitable for this purpose. Matrix factorization [1] is a widely studied idea to extract some hidden features, particularly used in he field of collaborative filtering [2]. As its name suggests, matrix factorization requires a matrix in the first place. In terms of collaborative filtering, a matrix could be a user-item data which is composed of values referring to the preferences of users' on items. Matrix factorization can be used to extract hidden features using this preference matrix both for users and items.

ALORS [3] as an algorithm selection approach [4] borrows the same idea to extract features for the instance-algorithm matrices. Each matrix entry is the performance of an algorithm on a problem instance. Singular Value Decomposition (SVD) [5] was utilized as the matrix factorization method on a performance matrix. It offers an algorithm selection strategy through mapping a set of hand-picked features to those extract features regarding instances.

This study provides an analysis strategy for benchmark sets by using matrix factorization. The analysis is performed on hyper-heuristics across 6 problem domains using HyFlex [6]. Considering the problem-independent nature of hyper-heuristics, the analysis is on all the problems at once. Thus, the proposed approach differentiates from the studies focusing on a single problem domain and utilizing problem specific elements such as instance features. The CHeSC 2011 competition results on HyFlex are additionally reviewed.

The remainder of the paper is presented as follows. Section II presents a brief literature on hyper-heuristics and the problem analysis techniques. Section III introduces the proposed approach while the computational results are discussed in Section IV. The paper is finalized with a summary and the follow-up research path in Section V.

II. Background

A. Hyper-heuristics

Hyper-heuristics [7], [8] have been successfully applied to various problem domains such as scheduling [9] , timetabling [10] , routing , cutting & packing [11], [12] and decision problems [13]. They have been considered under two types including the *selection* and *generation* hyper-heuristics [14].

The selection hyper-heuristics aim at determining the best possible low-level heuristics to be used while deciding how to act on the resulting solutions by those heuristics. The selection process is handled by a *heuristic selection* mechanism. Incorporating a learning strategy into heuristic selection has been very popular such as case-based reasoning [15], choice function [16], learning automata credit-assignment , dynamic heuristic set and learning classifier systems [17] while rather simple approaches have been also studied such as simple random and greedy [18]. The adaptive operator selection

approaches [19] perform a similar task to the heuistic selection and can also be employed in hyper-heuristics [20]. Some of these selection operations are performed *online*, meaning that choosing heuristics while a problem instance is being solved. *Offline* is preferred in others in order to take advantage of prior knowledge. The traditional algorithm selection [4], [21] studies also fall into this Offline category. After a selected heuristic is being applied, it is decided whether the resulting partial/complete solution is good enough to accepted. The evaluation operation is performed by a *move acceptance* criterion.

The generation hyper-heuristics are studied to automatically design new heuristics. Genetic programming [22], grammatical evolution [12] and differential evolution [23] have been particularly used for the purpose of generation. The majority of the studies on this type of hyper-heuristics is considered Offline. Hybridisation approaches in particular are proper to perform Online generation.

B. Problem Analysis

One way of problem analysis is to consider empirical instance hardness [24], [25], [26], [27], [28]. Having some knowledge on instance hardness can help to pick the proper algorithms w.r.t. the instances' hardness levels. Such information can also be utilized to reveal what makes a particular problem instance hard to solve. The outcome can be practical to design or modify a strong algorithm targeting a certain problem/$\sim$ instance. It can also be useful to generate new instances [29], [30], [31], [32], [33] that can challenge algorithms or just with particular characteristics.

For understanding instances' comparative hardness or similarities, using a group of hand-picked, descriptive features has been practical. In [34], for example, a group of features for the traveling salesman problem was utilized to analyze the problem instances with the help of algorithm selection. The instances were initially generated such that a certain level of diversity is achieved in terms of the instance hardness. Principal component analysis (PCA) [35] was then used for visualization purposes. As mentioned earlier, in [3] , similarly, algorithm selection was used for instance analysis. However, their focus was features extracted from performance data instead of directly relying on the hand-picked features. The extracted features were derived with the help of singular value decomposition (SVD) [5]. Additionally, the Locally Linear Embedding approach [36] was used to quantify the actual contribution of each hand-picked feature to the hardness of the instances depending on the algorithms' performance on those instances.

III. Benchmark Set Analysis

The benchmark set analysis approach, detailed in Algorithm 1, relies on ALORS [3]. ALORS takes a rank matrix $\mathcal{M}_{nxm}$ where its rows represent m problem instances while its columns refer to n algorithms. Thus, $\mathcal{M}_{i,j}$ indicates the rank performance of algorithm j on instance i. A set of hidden features are extracted from $\mathcal{M}$ using matrix factorization.

Algorithm 1: Matrix Factorization based Benchmark set Analyzer

Input: A performance matrix M, Matrix factorization method θ, Clustering algorithm Λ, Number of latent features: k

1 **Rank matrix conversion**: $M \rightarrow \mathcal{M}$
2 **Feature extraction**: $\theta(\mathcal{M}_{n\times m}) = \mathrm{U}_{n\times k}\ \mathrm{V}^{\mathcal{T}}{}_{k\times m}$
3 **Instance clustering**: $C = \Lambda(\mathrm{U})$
4 **Feature analysis**: $\Phi : U \rightarrow C$

Matrix factorization basically refers to generating a number of matrices where their multiplication is equal to or approximates to the original matrix. ALORS accommodates singular value decomposition (SVD) as the matrix factorization method.

It essentially produces three matrices. U and V matrices are composed of the latent (hidden) features characterizing the rows and columns of a given matrix. Regarding this paper, U matrix represents instances while V matrix represents algorithms. The third matrix, $\sum$ is a diagonal matrix involving singular values. Singular values can be considered importance metrics of the extracted latent features. Besides that, the singular values are sorted. This means that the first k latent features both for instances and algorithms are more critical or contains more information to represent them than the second latent features. The same situation is valid for the subsequent latent features. Thus, using only the first a few initial features, i.e. dimensionality reduction, can already represent both instances and algorithms rather well. Additionally, it can help to remove the noise in the matrix in case there is. When $k = \max(n, m)$, the multiplication of the SVD's resulting matrices become equal to the original matrix $\mathcal{M}$. Thus, $1 \leq k \leq \max(n, m)$.

The latent features for the instances from U are directly used to examine the instances' diversity and similarity. For this purpose, instances are clustered by using these latent features. k-means clustering [37] is applied to discover different instance groups. The number of clusters is decided w.r.t. the silhouette score [38]. Using the resulting instance clusters, the performance of the constituent algorithms of $\mathcal{M}$ is analyzed. Then, the characteristics of each instance cluster C_i are examined by building a model Φ using random forest [39] classification[1]. The resulting model is used to disclose the contribution of each algorithms' rank to the clusters most by the help of the Gini importance [39].

IV. Computational Results

For the analysis, the results of 10 selection hyper-heuristics run reported in [40] are used. Each hyper-heuristic was tested on each instance for 1 hour. Considering the stochastic nature of the hyper-heuristics, each instance run was repeated 10 times. Those hyper-heuristics utilize one of the heuristic selection mechanisms between Adaptive Dynamic Heuristic Set (ADHS) and Simple Random (SR). SR is the uniform random selection while ADHS incorporates a learning mechanism.

[1]The scikit-learn library is used with default values

Each of these selection methods is combined with an acceptance criterion among Adaptive Iterated Limited List-based Threshold Accepting (AILLA), Great Deluge (GD), Improving or Equal (IE), Late Acceptance (LATE) and Simulated Annealing (SA). All the acceptance mechanisms except IE can accept worsening solutions. Thus, they are more aggressive in terms of diversification. IE can provide only limited diversification by accepting equal quality solutions. The hyper-heuristics utilized are listed as follows:

- ADHS-AILLA, ADHS-GD, ADHS-IE, ADHS-LATE, ADHS-SA
- SR-AILLA, SR-GD, SR-IE, SR-LATE, SR-SA

TABLE I
PROBLEM DOMAINS AVAILABLE IN HYFLEX 1.0 WITH THE CHESC 2011 INSTANCES' INDICES

Dataset	#Instances	CHeSC 2011
Boolean Satisfiability (MSAT)	12	3, 5, 4, 10, 11
1D Bin Packing (BP)	12	7, 1, 9, 10, 11
Flowshop Scheduling (FSP)	12	1, 8, 3, 10, 11
Personnel Scheduling (NRP)	12	5, 9, 8, 10, 11
Travelling Salesperson (TSP)	10	0, 8, 2, 7, 6
Vehicle Routing (VRP)	10	6, 2, 5, 1, 9

Each of these hyper-heuristics was run on HyFlex 1.0[2] [6]. As detailed in Table I, HyFlex contains 68 instances from 6 problem domains. 30 of these instances were used in the first cross-domain heuristic search challenge (CHeSC)[3], which will be referred later. To be able to deliver a strong analysis, the best/minimum and average of 10 runs (MIN-AVG) performance are added as two performance metrics for each hyper-heuristic. Thus, $\mathcal{M}$ has $n = 68$ rows and $m = 20$ columns. For SVD, k is set to 5 which means that only the first 5 dimensions are used out of $\max(68, 20) = 20$.

Table II presents the detected 7 instance clusters. The cluster C_2 involving 27 instances, about 40% of all the instances, comes as the largest cluster. Referring to the problems contribution to the overall benchmark set, all the TSP and FSP instances except one fall into the same cluster, C_2. Also, 4 BP and 2 NRP instances happen to be in this cluster. This suggest that either TSP or FSP could be switched with a new problem or the problem instances used could be revised for better benchmark diversity. The cluster C_7 also carries an interesting property of having a single problem, i.e. 10 out of 12 SAT instances. As the smallest clusters, C_4 involves only 4 BP instances while C_7 is composed of 3 BP and 1 VRP instances. Rather detailed instance similarity analysis can be performed with the hierarchical clustering as illustrated in Figure 1. For instance, NRP-11 & TSP-8 and FSP-0 & TSP-6 instances share highly similar characteristics.

In addition to evaluating the complete instance set, the fairness of the CHeSC 2011 competition is examined. For this purpose, the distribution of the instances used in the competition against the clusters are considered. From each cluster, $C_1 \rightarrow C_7$, the proportions of the competition instances are ∼38% (3/8), ∼56% (15/27), ∼29% (2/7), ∼25% (1/4), ∼38% (3/8), ∼25% (1/4) and ∼50% (5/10), respectively. Although the instance set is rather small, the selected instances for the competition show reasonable level of diversity. Additionally, the competition results (Table 3) are also taken into account to see whether the clusters found makes sense on the results. It should be noted that the competition results are based on the F1 scoring which gives 10, 8, 6, 5, 4, 3, 2, 1 and 0 from the best to the worst performing hyper-heuristics on each instance. Also, the scores are given based on the average performance of the competing algorithms. Table 4 illustrates both the Spearman's rank correlation coefficients (ρ) and p-values (with 95% confidence) between each problem domain pairs based on the competition. The results indicate there are strong correlations between the following problem pairs: MSAT-VRP, BP-FSP, BP-TSP, FSP-TSP, FSP-VRP and NRP-VRP. Referring to the cluster C_2, the strong similarity between BP, FSP and TSP is also detected on the competition setting. NRP-VRP correlation is also revealed considering that in two

[2]www.hyflex.org

[3]www.asap.cs.nott.ac.uk/external/chesc2011/

TABLE III
CHESC 2011 COMPETITION SCORES (ALGORITHMS ARE SORTED FROM THE BEST TO THE WORST BASED ON THEIR OVERALL SCORES)

Algorithm	Score	MSAT	BP	FSP	NRP	TSP	VRP
GIHH	181	34.75	45.0	37.0	9.0	40.25	15.0
VNS-TW	134	34.25	3.0	34.0	39.5	17.25	6.0
ML	131.5	14.5	12.0	39.0	31.0	13.0	22.0
PHUNTER	93.25	10.5	3.0	9.0	11.5	26.25	33.0
EPH	89.75	0.0	10.0	21.0	10.5	36.25	12.0
HAHA	75.75	32.75	0.0	3.5	25.5	0.0	14.0
NAHH	75	14.0	19.0	22.0	2.0	12.0	6.0
ISEA	71	6.0	30.0	3.5	14.5	12.0	5.0
KSATS-HH	66.5	24.0	11.0	0.0	9.5	0.0	22.0
HAEA	53.5	0.5	3.0	10.0	2.0	11.0	27.0
ACO-HH	39	0.0	20.0	9.0	0.0	8.0	2.0
GenHive	36.5	0.0	14.0	7.0	6.5	3.0	6.0
DynILS	27	0.0	13.0	0.0	0.0	13.0	1.0
SA-ILS	24.25	0.75	0.0	0.0	19.5	0.0	4.0
XCJ	22.5	5.5	12.0	0.0	0.0	0.0	5.0
AVEG-Nep	21	12.0	0.0	0.0	0.0	0.0	9.0
GISS	16.75	0.75	0.0	0.0	10.0	0.0	6.0
SelfSearch	7	0.0	0.0	0.0	4.0	3.0	0.0
MCHH-S	4.75	4.75	0.0	0.0	0.0	0.0	0.0
Ant-Q	0	0.0	0.0	0.0	0.0	0.0	0.0

TABLE IV
SPEARMAN'S RANK CORRELATION COEFFICIENTS ON THE CHESC 2011 RESULTS (TABLE 3)

		MSAT	BP	FSP	NRP	TSP
BP	ρ	0.112				
	p-value	0.640				
FSP	ρ	0.368	0.529			
	p-value	0.110	0.016			
NRP	ρ	0.477	-0.037	0.444		
	p-value	0.033	0.877	0.050		
TSP	ρ	0.129	0.582	0.786	0.314	
	p-value	0.589	0.007	0.000	0.178	
VRP	ρ	0.550	0.160	0.552	0.475	0.357
	p-value	0.012	0.502	0.012	0.034	0.122

TABLE II
CLUSTER'S PROBLEM INSTANCES (CHESC 2011 INSTANCES ARE SHOWN IN BOLD)

Cluster# (size)	MSAT	BP	FSP	NRP	TSP	VRP
C_1 (8)	-	6	-	0, 1, **10**	-	**5**, 7, 8, **9**
C_2 (27)	-	**7**, **9**, **10**, **11**	0, **1**, 2, **3**, 4, 5, 7, **8**, 9, **10**, **11**	2, **11**	**0**, 1, **2**, 3, 4, 5, **6**, **7**, **8**, 9	-
C_3 (7)	1, 2	-	6	6, 7, **8**, **9**	-	-
C_4 (4)	-	0, **1**, 4, 5	-	-	-	-
C_5 (8)	-	-	-	3, 4, **5**	-	0, **1**, **2**, 3, 4
C_6 (4)	-	2, 3, 8	-	-	-	**6**
C_7 (10)	0, **3**, **4**, **5**, 6, 7, 8, 9, **10**, **11**	-	-	-	-	-

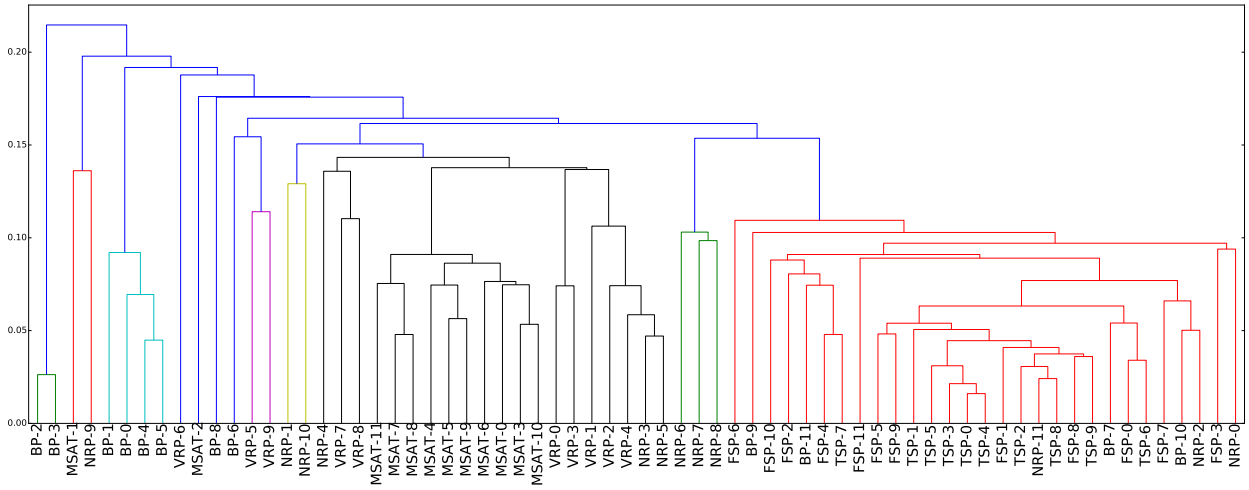

Fig. 1. Hierarchical clustering of the HyFlex instances using the instances' latent features through MIN-AVG data

clusters, C_1 and C_5, NRP and VRP have instances. Differently, MSAT-VRP pair points out strong correlation even though it was not discovered in the proposed approach. Yet, such a difference possible since the presented analysis employs a slightly different performance metric, Rank rather than the Formula 1 scoring. It also uses MIN-AVG data rather than only AVG with the runtime of 1 hour instead of 10 minutes. In the competition, the winning hyper-heuristic, i.e. GIHH, shows near/-best performance on MSAT, BP, FSP, TSP and average performance on NRP, VRP. Considering that the MSAT, BP, FSP, TSP competition instances are from C_2 and $C7$ which are the largest instance clusters, the GIHH's leading performance can be validated also from the clusters. As another example, HAHA performs rather well on MSAT, NRP and VRP yet very poor on BP, FSP and TSP. This shows that HAHA fails to address the instances of C_2 while perform effectively on C_7.

Figure 2 presents the boxplots of the average ranks for each tested hyper-heuristic across all the instances from each cluster. For the cluster C_1, the contribution of heuristic selection is rather limited. Clear performance variations are available for the move acceptance criteria. LATE and AILLA provide the top performance, likely due to their list-based characteristics. Overall, SR-LATE reaches the best average rank performance on the 8 instances of C_1. As the largest instance cluster, C_2 contains 27 instances. Thus, the performance of a hyper-heuristic on C_2 can significantly affect its performance across all the instances. Unlike C_1, C_2 seems to be benefiting from utilizing a learning-based heuristic selection mechanism. In that respect, all the hyper-heuristics incorporating ADHS outperform the hyper-heuristics with SR. Besides that, ADHS-AILLA offers significantly better performance compared to the rest. Referring to the cluster C_3, there is a resemblance to the cluster C_1. Yet, the ranks aren't as robust as in C_1. However, the average quality improves on C_3 for SR-AILLA, SR-IE, ADHS-AILLA and ADHS-IE. Especially improvement over the hyper-heuristics using IE indicates that the instances in C_3 do not require aggressive diversification. The results for C_5 indicate that $AILLA$ as an acceptance criterion is good enough to reach high quality results while ADHS can push its performance even further. Again, IE's poor performance reveals that diversification is highly critical for these instances. For C_6, ADHS-GD comes as the best hyper-heuristic for the first time. It should be noted that C_6 is composed of 3 BP and 1 VRP instances as the smallest cluster, with $C4$. In other words, these 4 instances look like rather special case compared to the complete instance set. For C_7, involving only the MSAT instances, there is no clear contribution of one hyper-heuristic

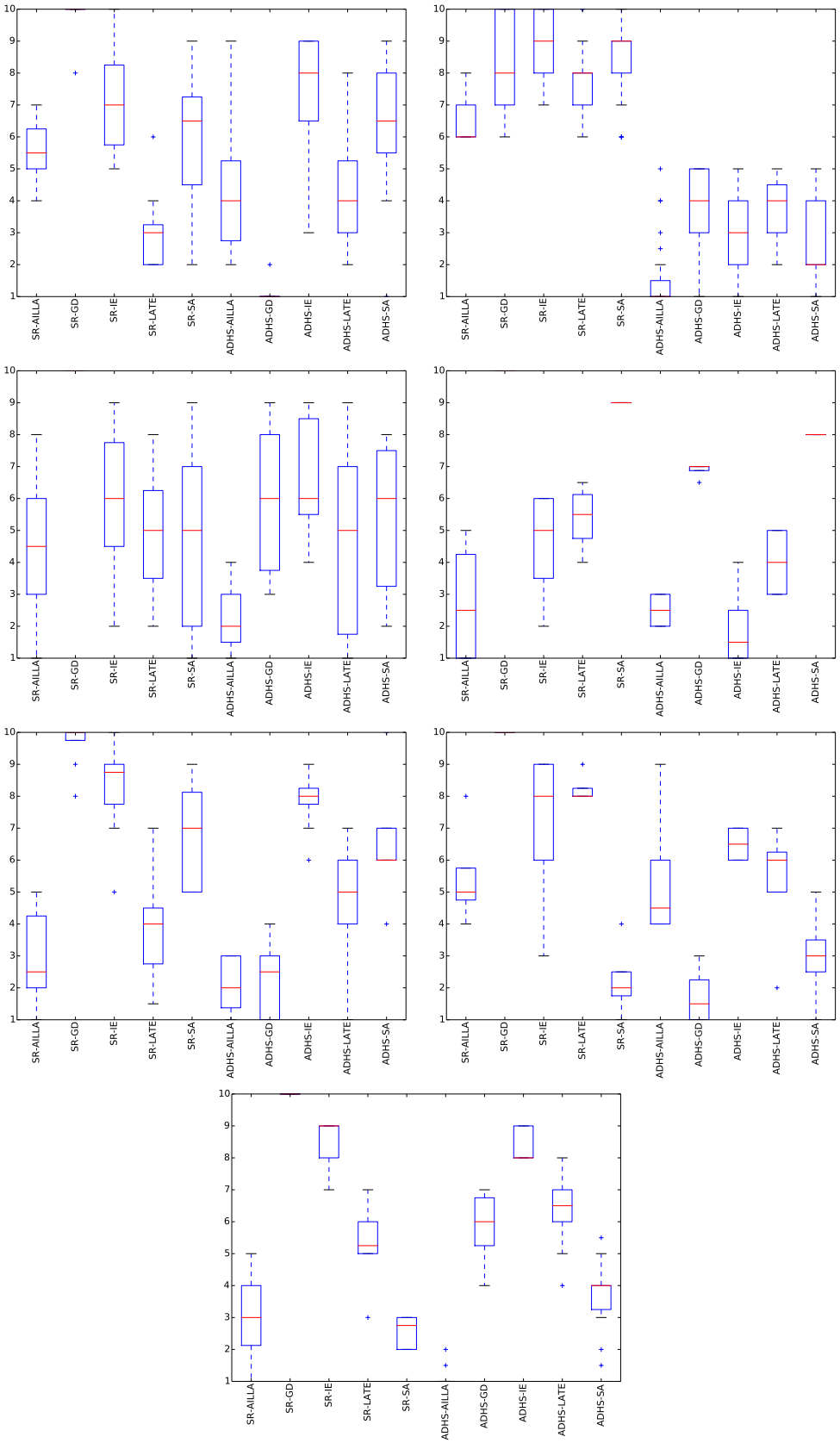

Fig. 2. Average rank boxplots, each plot refers to a cluster (from left to right, top to bottom: the clusters have 8, 27, 7, 4, 8, 4 and 10 instances, respectively)

sub-mechanism, selection or acceptance. However, the poor performance of the hyper-heuristics with IE denotes that more than limited diversification is required to successfully solve the corresponding MSAT instances.

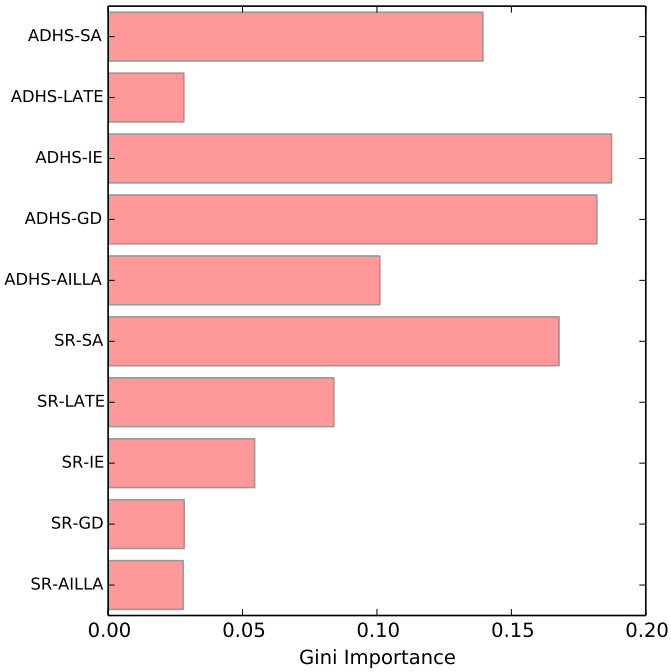

Fig. 3. Importance of each hyper-heuristic's rank performance w.r.t. the instance clusters

Figure 3 presents an importance analysis on the hyper-heuristics' rank performance referring to the latent features. ADHS-IE and ADHS-GD contribute the most to the clusters while ADHS-LATE, SR-AILLA and SR-LATE show limited contribution. This means that if the performance of ADHS-IE or ADHS-GD varies substantially, it is an indicator of the instance diversity. However, the performance of ADHS-LATE, SR-AILLA and SR-LATE give limited information about the instance diversity compared to the other hyper-heuristics tested.

Figure 4 illustrates the contribution of each performance-based feature, rank, on each latent feature unlike Figure 3 where the overall contributions are exhibited. In other words, the plots reveal the composition of each latent feature which is not obvious to tell what they represent. The compositions are interpreted through the Gini importance we retrieved from the Random Forest models mapping the matrix itself to the latent features. For the first latent feature-as the most critical feature characterizing the instances, SR-IE-AVG shows the highest contribution. SR-IE-BEST, ADHS-IE-AVG and ADHS-IE-BEST follow it as the rank-based features contributing to the first latent feature the most. Considering that the first latent feature is the most important one, running SR-IE across a selected set of instances can show the diversity level of that instance set by looking at its BEST/MIN performance. It should also be noted that IE as an acceptance criterion in particular helps to confirm diversity since its limited diversification characteristic makes it fail to show high performance on some instances. Thus, its performance can easily vary across different instances which is useful to show diversity. If we look this idea from the inverse perspective, one of the worst diversity revealing feature is ADHS-AILLA-AVG since it shows high performance across different instances thus limited performance diversity. At the same time, this demonstrates the robustness of ADHS-AILLA. Regarding heuristic selection, since SR and ADHS do not make much of a difference to reveal diversity, it indicates that move acceptance can be more critical especially when enough running time is given.

The second latent feature primarily depends a single rank-based feature which is ADHS-IE-AVG while the rest does rather limited contribution. This shows that the second latent feature ignore the separate affect of heuristic selection and move acceptance but looks at the hyper-heuristics as black-box algorithms. The third latent feature is particularly composed of ADHS-SA-AVG and SR-SA-AVG. Since SA involves more comprehensive diversification capabilities compared to IE, it makes sense to see SA for the third latent feature. The fourth latent feature predominantly comes from ADHS-GD-AVG and ADHS-BEST. For the last latent feature extracted, ADHS-AILLA-AVG contributes significantly the most as expected, referring to what we discussed regarding its limited contribution to the first latent feature.

V. Conclusion

This study offers an analysis strategy for examining the fairness and diversity of a given set of benchmark problem instances. The approach performed is practical for any problem where the performance of an algorithm can be quantified, e.g. in terms of solution quality, runtime, success rate etc. It can particularly tell whether a given set of instances is good-enough to evaluate a given algorithm while comparing it against other algorithms tested on the same problem. The analysis approach introduced employs matrix factorization that is applied to a performance data of a suite of algorithms on the instances of a benchmark set. Matrix factorization helps to extract a set of hidden features that can distinguishes the benchmark instances. For experimental analysis, the HyFlex 1.0 hyper-heuristic framework with 6 problem domains is used. Due to the problem-independent nature of hyper-heuristics, the analysis is not just about a single problem but on 6 problems together. The computational results indicated that the proposed idea is able to determine different instance groups, each mainly involving instances from different problem domains. Additionally, it was shown that the existing HyFlex problem instances maintain some level of diversity which could still be further improved. The diversity on HyFlex was also successfully reflected to the 1st cross-domain heuristic search challenge (CHeSC 2011) that was performed a subset of those HyFlex instances.

The study will be extended by using generic instance features, as in [41] , for a deeper analysis. Next, a set of hand-picked problem specific features will be used for each

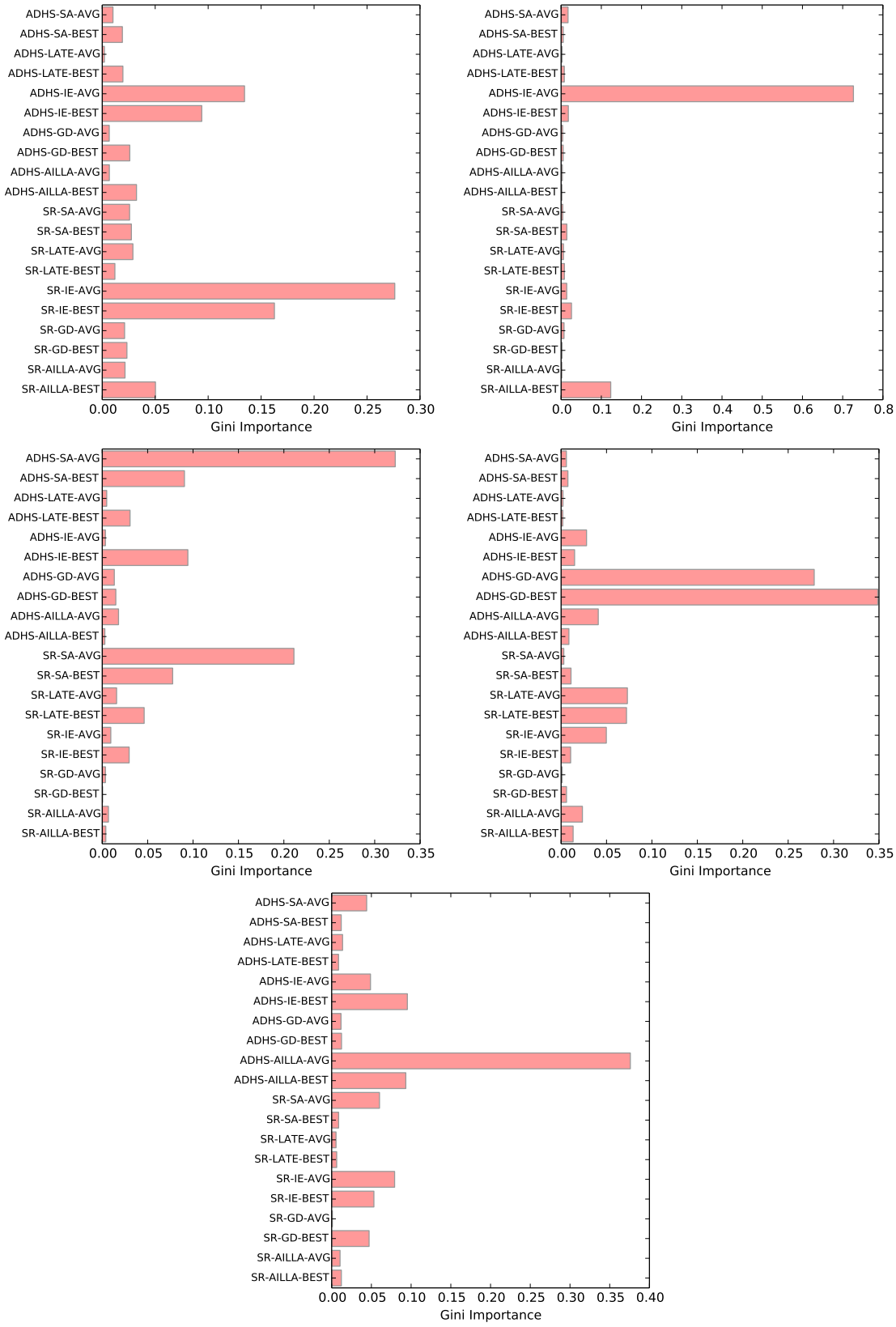

Fig. 4. What each of the top 5 latent features consists of (from left to right, top to bottom: latent feature $1 \rightarrow 5$, respectively)

problem so that the existing instance analysis techniques will be incorporated. The hand-picked features will be additionally utilized to perform algorithm selection [4] on hyper-heuristics Also, new problems will be added while providing more instances for each existing problem domain. Finally, the hyper-heuristics from CHeSC 2011 will be integrated in order to make a similar analysis on the algorithms.

REFERENCES

[1] Y. Koren, R. Bell, C. Volinsky *et al.*, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

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

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

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

[5] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische mathematik*, vol. 14, no. 5, pp. 403–420, 1970.

[6] G. Ochoa, M. Hyde, T. Curtois, J. Vazquez-Rodriguez, J. Walker, M. Gendreau, G. Kendall, B. McCollum, A. Parkes, S. Petrovic, and E. Burke, "Hyflex: A benchmark framework for cross-domain heuristic search," in *European Conference on Evolutionary Computation in Combinatorial Optimisation(EvoCOP'12)*, ser. LNCS, vol. 7245. Berlin: Springer, 2012, pp. 136–147.

[7] E. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. Woodward, "A classification of hyper-heuristic approaches," *Handbook of Metaheuristics*, pp. 449–468, 2010.

[8] 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.

[9] S. Chen, Z. Li, B. Yang, and G. Rudolph, "Quantum-inspired hyper-heuristics for energy-aware scheduling on heterogeneous computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 6, pp. 1796–1810, 2016.

[10] N. Pillay, "A review of hyper-heuristics for educational timetabling," *Annals of Operations Research*, vol. 239, no. 1, pp. 3–38, 2016.

[11] H. Terashima-Marin, A. Morán-Saavedra, and P. Ross, "Forming hyper-heuristics with gas when solving 2d-regular cutting stock problems," in *IEEE Congress on Evolutionary Computation (CEC)*, vol. 2. IEEE, 2005, pp. 1104–1110.

[12] M. A. Sotelo-Figueroa, H. J. P. Soberanes, J. M. Carpio, H. J. F. Huacuja, L. C. Reyes, J. A. S. Alcaraz, and A. Espinal, "Generating bin packing heuristic through grammatical evolution based on bee swarm optimization," in *Nature-Inspired Design of Hybrid Intelligent Systems*. Springer, 2017, pp. 655–671.

[13] M. Bader-El-Den and R. Poli, "Generating sat local-search heuristics using a gp hyper-heuristic framework," in *International Conference on Artificial Evolution (Evolution Artificielle)*. Springer, 2007, pp. 37–49.

[14] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward, "Exploring hyper-heuristic methodologies with genetic programming," in *Computational intelligence*. Springer, 2009, pp. 177–201.

[15] E. K. Burke, B. L. MacCarthy, S. Petrovic, and R. Qu, "Knowledge discovery in a hyper-heuristic for course timetabling using case-based reasoning," in *International Conference on the Practice and Theory of Automated Timetabling*. Springer, 2002, pp. 276–287.

[16] M. Maashi, G. Kendall, and E. Özcan, "Choice function based hyper-heuristics for multi-objective optimization," *Applied Soft Computing*, vol. 28, pp. 312–326, 2015.

[17] J. Marin-Blazquez and S. Schulenburg, "A hyper-heuristic framework with XCS: Learning to create novel problem-solving algorithms constructed from simpler algorithmic ingredients," in *IWLCS*, ser. LNCS, T. Kovacs, X. Llor, K. Takadama, P. Lanzi, W. Stolzmann, and S. Wilson, Eds., vol. 4399. Springer, 2007, pp. 193–218.

[18] P. Cowling, G. Kendall, and E. Soubeiga, "A hyperheuristic approach to scheduling a sales summit," in *PATAT '00: Selected papers from the Third International Conference on Practice and Theory of Automated Timetabling III*. London, UK: Springer-Verlag, 2001, pp. 176–190.

[19] 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.

[20] M. G. Epitropakis, F. Caraffini, F. Neri, and E. K. Burke, "A separability prototype for automatic memes with adaptive operator selection," in *IEEE Symposium on Foundations of Computational Intelligence (FOCI)*. IEEE, 2014, pp. 70–77.

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

[22] J. Park, Y. Mei, S. Nguyen, G. Chen, M. Johnston, and M. Zhang, "Genetic programming based hyper-heuristics for dynamic job shop scheduling: Cooperative coevolutionary approaches," in *European Conference on Genetic Programming*. Springer, 2016, pp. 115–132.

[23] M. Sotelo-Figueroa, H. Soberanes, J. Carpio, H. Fraire Huacuja, L. Reyes, and J. Soria Alcaraz, "Evolving bin packing heuristic using micro-differential evolution with indirect representation," in *Recent Advances on Hybrid Intelligent Systems*, ser. Studies in Computational Intelligence, O. Castillo, P. Melin, and J. Kacprzyk, Eds. Springer Berlin / Heidelberg, 2013, vol. 451, pp. 349–359.

[24] P. Cheeseman, B. Kanefsky, and W. M. Taylor, "Where the really hard problems are," in *IJCAI*, vol. 91, 1991, pp. 331–337.

[25] T. Jones, S. Forrest *et al.*, "Fitness distance correlation as a measure of problem difficulty for genetic algorithms," in *ICGA*, vol. 95, 1995, pp. 184–192.

[26] Y. Ruan, H. A. Kautz, and E. Horvitz, "The backdoor key: A path to understanding problem hardness," in *AAAI*, vol. 4, 2004, pp. 118–123.

[27] K. Smith-Miles and L. Lopes, "Measuring instance difficulty for combinatorial optimization problems," *Computers & Operations Research*, vol. 39, no. 5, pp. 875–889, 2012.

[28] K. Leyton-Brown, H. H. Hoos, F. Hutter, and L. Xu, "Understanding the empirical hardness of NP-complete problems," *Communications of the ACM*, vol. 57, no. 5, pp. 98–107, 2014.

[29] J. I. van Hemert, "Evolving combinatorial problem instances that are difficult to solve," *Evolutionary Computation*, vol. 14, no. 4, pp. 433–462, 2006.

[30] K. Smith-Miles and J. I. van Hemert, "Discovering the suitability of optimisation algorithms by learning from evolved instances," *Annals of Mathematics and Artificial Intelligence*, vol. 61, no. 2, p. 87, 2011.

[31] L. Lopes and K. Smith-Miles, "Generating applicable synthetic instances for branch problems," *Operations Research*, vol. 61, no. 3, pp. 563–577, 2013.

[32] K. Smith-Miles and S. Bowly, "Generating new test instances by evolving in instance space," *Computers & Operations Research*, vol. 63, pp. 102–113, 2015.

[33] Y. Malitsky, M. Merschformann, B. O'Sullivan, and K. Tierney, "Structure-preserving instance generation," in *10th International Conference on Learning and Intelligent Optimization*, ser. LNCS, P. Festa, M. Sellmann, and J. Vanschoren, Eds. Springer, 2016, vol. 10079, pp. 123–140.

[34] K. Smith-Miles and T. T. Tan, "Measuring algorithm footprints in instance space," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2012, pp. 1–8.

[35] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

[36] L. K. Saul and S. T. Roweis, "Think globally, fit locally: unsupervised learning of low dimensional manifolds," *Journal of Machine Learning Research*, vol. 4, no. Jun, pp. 119–155, 2003.

[37] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.

[38] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[39] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

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

[41] 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.