HARRIS: Hybrid Ranking and Regression Forests for Algorithm Selection

Lukas Fehring[1], **Jonas Hanselle**[1], **Alexander Tornede**[2]
[1] Department of Computer Science, Paderborn University, Germany
[2] Institute of Artificial Intelligence, Leibniz University Hannover, Germany
fehring2@mail.upb.de, jonas.hanselle@upb.de, tornede@tnt.uni-hannover.de

Abstract

It is well known that different algorithms perform differently well on an instance of an algorithmic problem, motivating algorithm selection (AS): Given an instance of an algorithmic problem, which is the most suitable algorithm to solve it? As such, the AS problem has received considerable attention resulting in various approaches – many of which either solve a regression or ranking problem under the hood. Although both of these formulations yield very natural ways to tackle AS, they have considerable weaknesses. On the one hand, correctly predicting the performance of an algorithm on an instance is a sufficient, but not a necessary condition to produce a correct ranking over algorithms and in particular ranking the best algorithm first. On the other hand, classical ranking approaches often do not account for concrete performance values available in the training data, but only leverage rankings composed from such data. We propose HARRIS-Hybrid rAnking and RegRessIon foreSts - a new algorithm selector leveraging special forests, combining the strengths of both approaches while alleviating their weaknesses. HARRIS' decisions are based on a forest model, whose trees are created based on splits optimized on a hybrid ranking and regression loss function. As our preliminary experimental study on ASLib shows, HARRIS improves over standard algorithm selection approaches on some scenarios showing that combining ranking and regression in trees is indeed promising for AS.

1 Introduction

To this day, there are competitions on solving hard instances of the SAT (boolean satisfiability problem) problem [10, 7]. In these competitions, one deals with a set of problems with the goal of solving them faster than the competitors. Here, the participants rarely use one algorithm to solve all problem instances. Instead, they utilize so-called algorithm selectors, often featuring machine learning models at their core, to predict the performance of different algorithms on the instance to select the one presumably performing best. In practice, most algorithm selectors either leverage a regression [23, 2, 8] or a ranking [3, 6, 20] model to predict the best algorithm.

Unfortunately, both ranking and regression models feature considerable drawbacks when used at the core of a selector. While creating a ranking across algorithms according to their predicted performance does indeed yield the correct ranking as long as the predictions are correct, such a ranking can also be created without correctly estimating the performance. More precisely, correct performance predictions are a sufficient, but not a necessary criterion to create a correct ranking across the algorithms. Correspondingly, one may wonder whether solving a regression problem might not be much harder than what is required. From this perspective, ranking models are a more intuitive solution. However, they often do not take the concrete performance values, which are usually present as training data, into account, but are trained based on rankings created from these.

6th Workshop on Meta-Learning at NeurIPS 2022, New Orleans.

Correspondingly, these ranking models are trained based on qualitative comparisons losing the actual quantitative information contained in the precise performance evaluations. As such, they lack the means to quantify how close two algorithms are in a predicted ranking and thus are more susceptible to problems arising from algorithms with actually very similar performance.

In this paper, we propose a new algorithm selector leveraging a machine learning model trained based on a composite loss with both a ranking and regression component, dubbed HARRIS. In particular, the core of HARRIS is formed by a random forest, whose trees are formed according to splits optimized on the aforementioned composite loss. By doing so, HARRIS combines the strengths of both ranking and regression models while alleviating their weaknesses.

2 The Algorithm Selection Problem

In Algorithm Selection (AS) [15], we aim to find the best algorithm A_i from a set of candidate algorithms $\{A_1, ..., A_k\} = \mathcal{A}$ for a problem instance $I \in \mathcal{I}$ from a problem instance space $\mathcal{I}$. Formally, we seek to find a mapping, called algorithm selector $s : \mathcal{I} \rightarrow \mathcal{A}$, which maximizes a costly-to-evaluate performance measure $m : \mathcal{A} \times \mathcal{I} \rightarrow \mathbb{R}$. Correspondingly, the optimal selector, called an oracle, is defined as

$$s^*(I) \in \arg\max_{A \in \mathcal{A}} \mathbb{E}[m(A, I)] \quad . \tag{1}$$

As the performance measure m is costly to evaluate, an exhaustive enumeration over the set of algorithms to choose the best performing one is no practical solution. This holds especially for constraint satisfaction problems, where one is finally interested in the solution to the instance, which is obtained as a result of the first algorithm run anyway. As a solution to this, most AS approaches leverage machine learning to learn a surrogate performance measure $\widehat{m} : \mathcal{A} \times \mathcal{I} \rightarrow \mathbb{R}$ mimicking the original performance measure m, while, in contrast to the original performance measure, being cheap to evaluate. Using such a surrogate $\widehat{m}$, selectors can be constructed as $s(I) = \arg\max_{A \in \mathcal{A}} \widehat{m}(A, I)$.

To learn such surrogates, we assume that we can represent instances in terms of features, which are at least somewhat correlated with the performance of one or multiple of the algorithms. Formally, these features are computed by a feature function $g : \mathcal{I} \rightarrow \mathcal{X}$ and we will write $\boldsymbol{x}_I \in \mathcal{X}$, when we want to address the features of instance $i \in \mathcal{I}$. When considering the algorithmic problem of SAT, such features could be, for example, the number of clauses or the number of variables. Moreover, we assume that we are given some prior evaluations of the performance measure m for at least some of the algorithms on some training instances $\mathcal{I}_{train} \subset \mathcal{I}$, which we can use for learning. More formally, we assume training data with labels $\boldsymbol{y}_I = [m(I, A_1), \dots, m(I, A_k)] \in \mathbb{R}^k$ where $A_i \in \mathcal{A}$, i.e.,

$$\mathcal{D}_{train} = \{(\boldsymbol{x}_I, \boldsymbol{y}_I) | I \in \mathcal{I}_{train}\} \quad . \tag{2}$$

3 From Pure Ranking or Regression to Hybrid Ranking and Regression

In practice, the surrogate performance measure $\widehat{m}$ is often implemented as a regression or ranking model based on a loss function $\ell : \mathbb{R}^k \times \mathbb{R}^k \rightarrow \mathbb{R}$, where we assume rankings to be represented as a k-dimensional real-valued vector for simplicity. While the first kind of models is trained using a regression loss function such as the mean squared error, which is aimed at minimizing the differences between the predicted algorithm performances $\widehat{m}(\cdot, \cdot)$ and the true performances $m(\cdot, \cdot)$ on the training data $\mathcal{D}$ making it a quantitative approach. Contrary to that, ranking models are trained based on ranking losses such as the (inverse of the) Spearman correlation [17], which tries to maximize the correlation between the ranking across the algorithms imposed by the predicted latent utility values $\widehat{m}(\cdot, \cdot)$ and the ranking imposed by the true performances $m(\cdot, \cdot)$ making it a qualitative approach.

Recall that both of these approaches have a significant disadvantage: On the one hand, regression approaches try to predict the performance of an algorithm on an instance as accurately as possible, solving a, perhaps, harder problem than necessary as we are actually just interested in correctly *ranking* the algorithms. On the other hand, ranking approaches often ignore the concrete performance evaluations available in the training data and instead focus only on the ground truth ranking imposed by such values and correspondingly, ignore valuable data.

This problem has been discussed before in [9] in the context of AS (and earlier in a more general setting in [16]), who advocate leveraging hybrid ranking and regression loss functions

$$\ell_\lambda(\boldsymbol{y}, \widehat{\boldsymbol{y}}) = \lambda \ell_{\mathit{regression}}(\boldsymbol{y}, \widehat{\boldsymbol{y}}) + (1-\lambda)\ell_{\mathit{ranking}}(\boldsymbol{y}, \widehat{\boldsymbol{y}}) \tag{3}$$

composed of a convex combination of a regression loss function $\ell_{\mathit{regression}} : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$ and a ranking loss function $\ell_{\mathit{ranking}} : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$. Here, $\lambda \in [0, 1]$ is a hyperparameter controlling how strong the two loss functions influence the hybrid loss. The underlying idea is to leverage the strengths of the two approach classes, i.e., focusing on the ranking problem while also incorporating the precise performance information available in the training data and as such, eliminate their main weaknesses in the context of AS. The authors of [9] found that training simple linear models and neural networks to predict latent utility values for algorithms based on such a hybrid loss function can indeed be beneficial and in particular, that values of $0 < \lambda < 1$ can yield the best performance.

4 Hybrid Ranking and Regression Forests

Building upon the successful work [9], in this work, we generalize the idea of training models based on such a hybrid loss function to tree-based models, known to be very effective in AS [21]. We build forests of hybrid trees, detailed in the following, analogously to standard random forests [4].

Recall that decision trees [5] are trained by splitting the training data $\mathcal{D}_{train}$ recursively into two subsets, i.e., nodes $\mathcal{D}^+_{train}, \mathcal{D}^-_{train}$ based on a feature until a stopping criterion is reached and hence, that particular node is not split further. Such a leaf node is assigned a label computed from the associated dataset. In our case, we associate two labels with each node: First, a regression label $\widehat{\boldsymbol{y}}^{regression}_{\mathcal{D}} \in \mathbb{R}^k$ obtained by averaging the labels in the associated dataset $\mathcal{D}$ and second, a ranking label $\widehat{\boldsymbol{y}}^{ranking}_{\mathcal{D}} \in \mathbb{R}^k$ obtained by computing a consensus ranking through Borda's method [13].

We choose splits, consisting of a feature $f^* \in \mathbb{F}$, where $\mathbb{F}$ is the set of features, and a split point p^*, to minimize the weighted sum of the resulting dataset's losses wrt. the corresponding node labels, i.e.

$$(f^*, p^*) \in \operatorname*{arg\,min}_{(f,p)\in\mathbb{F}\times\mathbb{R}} \frac{|\mathcal{D}^+_{train}|}{|\mathcal{D}_{train}|} \cdot \mathcal{L}(\mathcal{D}^+) + \frac{|\mathcal{D}^-_{train}|}{|\mathcal{D}_{train}|} \cdot \mathcal{L}(\mathcal{D}^-) \ . \tag{4}$$

These losses quantify the homogeneity of labels in the dataset and are calculated as a convex combination of ranking and regression losses $\mathcal{L}(\mathcal{D}) = \lambda\mathcal{L}_{ranking}(\mathcal{D}) + (1-\lambda)\mathcal{L}_{regression}(\mathcal{D})$ where $\mathcal{L}(\mathcal{D}) = \frac{1}{|\mathcal{D}|}\sum_{(\boldsymbol{x}_I, \boldsymbol{y}_I)\in\mathcal{D}} \ell(\boldsymbol{y}_I, \widehat{\boldsymbol{y}}_{\mathcal{D}})$ and $\widehat{\boldsymbol{y}}_{\mathcal{D}}$ either corresponds to the ranking or regression label depending on whether ℓ is a ranking or regression loss function. We solve the optimization problem in Equation 4 by a simple enumeration of all possible features and splitting points imposed by the training data and choosing the best one. We utilize the mean squared error over all algorithms and instances as a regression loss $\mathcal{L}_{regression}$ as in [9]. As a ranking loss, we leverage the Spearman correlation turned into a loss function by subtracting it from 1, as we found this to work best in preliminary experiments. For the same reason we leverage the depth of a tree as a stopping criterion.

At prediction time, we propagate the instance down the tree until a leaf node l with $\mathcal{D}_l$ is reached. Based on label $\widehat{\boldsymbol{y}}^{regression}_{\mathcal{D}_l}$ we finally return the algorithm performing best according to this label.

Since the choice of split is dependent on the utilized loss functions, their behavior is the dominant factor in the model's quality. However, we found that not all ranking loss functions are well suited for Hybrid Forests and a mismatch in the scale of ranking and regression losses can result in one loss dominating the other thereby mitigating the impact of λ. To solve this we scaled the losses to the unit interval by scaling the performance data and dividing the ranking loss by the maximum possible loss. Moreover, the performance of HARRIS heavily depends on the right choice of λ.

5 Evaluation

We assess the quality of HARRIS with an experimental evaluation on a small subset of the ASLib benchmark [2]. All experiments were run on Intel Xeon E5-2695 v3 @ 2.30GHz CPU and 64 GB RAM. To set our results into context, we evaluate against ISAC [11], random forest regressor (RFR) that predicts each algorithms performance with a random forests, and SATzilla'11 [22] as done in several recent works [21, 18, 19]. In the interest of reproducibility, all code is available at [1].

The quality of each approach is evaluated using 10-fold cross validation with Kendall's Tau-b [12] and PAR10 [2]. Kendall's Tau quantifies the correlation between two rankings, where 1 indicates a perfect and -1 an inverse correlation. The PAR10 score corresponds to the runtime of the selected algorithm, if it is below a threshold C and $10 \cdot C$ otherwise. This threshold C is provided by the benchmark and corresponds to an upper bound on the runtime.

Table 1: Quality of the best known HARRIS configuration and competitors quantified with PAR10.

Scenario Name	HARRIS		ISAC		RFR		SAT	
CSP-Minizinc-Time-2016	**476.97**	±661.60	1194.64	±592.74	1044.55	±886.96	1058.08	±1184.75
MIP-2016	**1728.82**	±1649.62	2975.35	±3205.29	4332.53	±3320.56	2989.38	±2836.52
QBF-2016	**1382.08**	±328.42	1704.74	±757.74	1722.20	±836.78	1607.81	±627.32
CPMP-2015	**4891.47**	±1205.64	6094.06	±1972.29	5634.73	±2181.76	5152.87	±1521.40
ASP-POTASSCO	209.47	±59.07	348.57	±133.53	**178.81**	±52.20	236.48	±74.78
MAXSAT12-PMS	795.44	±399.61	1067.84	±700.12	631.14	±425.60	**553.61**	±371.80
QBF-2011	2464.69	±721.31	3271.56	±1270.76	1865.75	±804.27	**1520.36**	±630.32
SAT12-HAND	2150.58	±497.06	2587.54	±484.89	1552.95	±264.20	**1135.70**	±204.81
SAT12-ALL	2476.95	±202.07	1999.36	±321.40	**1144.46**	±280.86	1349.94	±173.25
Average Rank	2.11		3.56		2.33		**2.00**	

Table 1 displays the PAR10 scores averaged across all folds of each approach on the corresponding scenario including the standard deviation. Bold letters indicate the best performance. Note that the performances shown for HARRIS are optimistic as they correspond to the best performance achieved by varying λ in steps of 0.1 and the tree depth in $\{2, 4, 6, 8.10\}$. Thus, they can only serve to get an idea of what HARRIS is capable of, if λ can be tuned correctly. According to the average rank, HARRIS is the second best approach.

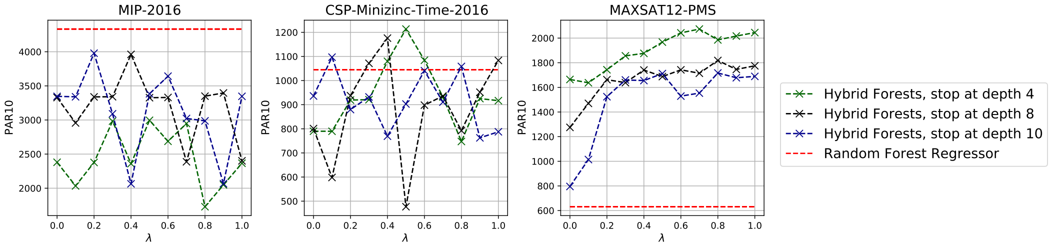

Figure 1: Visulisation of λ's impact on the quality of HARRIS

Figure 1 visualizes the change in quality of HARRIS with fixed depth for varying λ in the PAR10 metric. The results indicate that while λ strongly impacts the overall model quality, there are scenarios for which HARRIS is the superior/inferior model. More figures can be found in the appendix (Section 7).

6 Conclusion

In this work, we proposed a hybrid ranking and regression tree-based approach to AS called, HARRIS. Conceptually, HARRIS alleviates the weaknesses of pure ranking and regression AS solutions. In a prototypical experimental study, we showed that with appropriately set hyperparameters, HARRIS can outperform existing algorithm selectors on some scenarios. In future work, we plan to investigate whether tuning these hyperparameters automatically via means of hyperparameter optimization [1] yields good values on a scenario as suggested in [14]. Moreover, we plan to investigate other options for combining regression and ranking loss functions, for example, by working with probabilistic loss functions as this alleviates possible problems related to different scales.

[1] Github link: https://github.com/LukasFehring/HARRIS-Hybrid_rAnking_and_RegRessIon_foreSts

Acknowledgements

This work was supported by the German Research Foundation (DFG) within the Collaborative Research Center "On-The-Fly Computing" (SFB 901/3 project no. 160364472).

References

[1] Bernd Bischl, Martin Binder, Michel Lang, Tobias Pielok, Jakob Richter, Stefan Coors, Janek Thomas, Theresa Ullmann, Marc Becker, Anne-Laure Boulesteix, Difan Deng, and Marius Lindauer. "Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges". In: *CoRR* abs/2107.05847 (2021). arXiv: `2107.05847`. URL: `https://arxiv.org/abs/2107.05847`.

[2] Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger H. Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, and Joaquin Vanschoren. "ASlib: A benchmark library for algorithm selection". In: *Artificial Intelligence* 237 (2016), pp. 41–58. DOI: `10.1016/j.artint.2016.04.003`. URL: `https://doi.org/10.1016/j.artint.2016.04.003`.

[3] Pavel Brazdil and Carlos Soares. "A Comparison of Ranking Methods for Classification Algorithm Selection". In: *ECML 2000: Proceedings of the 11th European Conference on Machine Learning*. Vol. 1810. Lecture Notes in Computer Science. Springer, 2000, pp. 63–74. DOI: `10.1007/3-540-45164-1\_8`. URL: `https://doi.org/10.1007/3-540-45164-1%5C_8`.

[4] Leo Breiman. "Random Forests". In: *Mach. Learn.* 45.1 (2001), pp. 5–32. DOI: `10.1023/A:1010933404324`. URL: `https://doi.org/10.1023/A:1010933404324`.

[5] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984. ISBN: 0-534-98053-8.

[6] Tiago Cunha, Carlos Soares, and André C. P. L. F. de Carvalho. "A label ranking approach for selecting rankings of collaborative filtering algorithms". In: *SAC 2018: Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. ACM, 2018, pp. 1393–1395. DOI: `10.1145/3167132.3167418`. URL: `https://doi.org/10.1145/3167132.3167418`.

[7] Nils Froleyks, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda. "SAT Competition 2020". In: *Artificial Intelligence* 301 (2021), p. 103572. DOI: `10.1016/j.artint.2021.103572`. URL: `https://doi.org/10.1016/j.artint.2021.103572`.

[8] Jonas Hanselle, Alexander Tornede, Marcel Wever, and Eyke Hüllermeier. "Algorithm Selection as Superset Learning: Constructing Algorithm Selectors from Imprecise Performance Data". In: *PAKDD 2021: Proceedings of the 25th Pacific-Asia Conference*. Vol. 12712. Lecture Notes in Computer Science. Springer, 2021, pp. 152–163. DOI: `10.1007/978-3-030-75762-5\_13`. URL: `https://doi.org/10.1007/978-3-030-75762-5%5C_13`.

[9] Jonas Hanselle, Alexander Tornede, Marcel Wever, and Eyke Hüllermeier. "Hybrid Ranking and Regression for Algorithm Selection". In: *KI 2020: Proceedings of the 43rd German Conference on AI*. Vol. 12325. Lecture Notes in Computer Science. Springer, 2020, pp. 59–72. DOI: `10.1007/978-3-030-58285-2\_5`. URL: `https://doi.org/10.1007/978-3-030-58285-2%5C_5`.

[10] Marijn J. H. Heule, Matti Järvisalo, and Martin Suda. "SAT Competition 2018". In: *J. Satisf. Boolean Model. Comput.* 11.1 (2019), pp. 133–154. DOI: `10.3233/SAT190120`. URL: `https://doi.org/10.3233/SAT190120`.

[11] Serdar Kadioglu, Yuri Malitsky, Meinolf Sellmann, and Kevin Tierney. "ISAC - Instance-Specific Algorithm Configuration". In: *ECAI 2010: Proceedings of the 19th European Conference on Artificial Intelligence*. Vol. 215. Frontiers in Artificial Intelligence and Applications. IOS Press, 2010, pp. 751–756. DOI: `10.3233/978-1-60750-606-5-751`. URL: `https://doi.org/10.3233/978-1-60750-606-5-751`.

[12] Maurice G Kendall. "The treatment of ties in ranking problems". In: *Biometrika* 33.3 (1945), pp. 239–251.

[13] Shili Lin. "Rank aggregation methods". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 2.5 (2010), pp. 555–570.

[14] Marius Lindauer, Holger H. Hoos, Frank Hutter, and Torsten Schaub. "AutoFolio: An Automatically Configured Algorithm Selector". In: *Journal of Artificial Intelligence Research* 53 (2015), pp. 745–778. DOI: 10.1613/jair.4726. URL: https://doi.org/10.1613/jair.4726.

[15] John R. Rice. "The Algorithm Selection Problem". In: *Adv. Comput.* 15 (1976), pp. 65–118. DOI: 10.1016/S0065-2458(08)60520-3. URL: https://doi.org/10.1016/S0065-2458(08)60520-3.

[16] D. Sculley. "Combined regression and ranking". In: *SIGKDD 2010: Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining*. ACM, 2010, pp. 979–988. DOI: 10.1145/1835804.1835928. URL: https://doi.org/10.1145/1835804.1835928.

[17] C. Spearman. *"General Intelligence" Objectively Determined and Measured*. Studies in individual differences: The search for intelligence. East Norwalk, CT, US: Appleton-Century-Crofts, 1961. DOI: 10.1037/11491-006.

[18] A. Tornede, M. Wever, and E. Hüllermeier. "Towards Meta-Algorithm Selection". In: *Workshop on Meta-Learning (MetaLearn 2020) @ NeurIPS 2020*. 2020.

[19] Alexander Tornede, Lukas Gehring, Tanja Tornede, Marcel Wever, and Eyke Hüllermeier. "Algorithm selection on a meta level". In: *Machine Learning* (2022), pp. 1–34.

[20] Alexander Tornede, Marcel Wever, and Eyke Hüllermeier. "Extreme Algorithm Selection with Dyadic Feature Representation". In: *DS 2020: Proceedings of the 23rd International Conference on Discovery Science*. Vol. 12323. Lecture Notes in Computer Science. Springer, 2020, pp. 309–324. DOI: 10.1007/978-3-030-61527-7_21. URL: https://doi.org/10.1007/978-3-030-61527-7%5C_21.

[21] Alexander Tornede, Marcel Wever, Stefan Werner, Felix Mohr, and Eyke Hüllermeier. "Run2Survive: A Decision-theoretic Approach to Algorithm Selection based on Survival Analysis". In: *ACML 2020: Proceedings of The 12th Asian Conference on Machine Learning*. Vol. 129. Proceedings of Machine Learning Research. PMLR, 2020, pp. 737–752. URL: http://proceedings.mlr.press/v129/tornede20a.html.

[22] Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. "Hydra-MIP: Automated algorithm configuration and selection for mixed integer programming". In: *RCRA workshop on experimental evaluation of algorithms for solving problems with combinatorial explosion@IJCAI 2011* (2011), pp. 16–30.

[23] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. "The Design and Analysis of an Algorithm Portfolio for SAT". In: *CP 2007: Proceedings of the 13th International Conference on Constraint Programming*. Vol. 4741. Lecture Notes in Computer Science. Springer, 2007, pp. 712–727. DOI: 10.1007/978-3-540-74970-7_50. URL: https://doi.org/10.1007/978-3-540-74970-7%5C_50.

7 Appendix

Benchmark Scenarios

As mentioned in the paper, we evaluated the competitors performances with the ASlib [2] benchmark. However, we were not able to evaluate on all scenarios but just a subset of them. Key properties of them are shown in Table 2.

Table 2: Properties of the benchmark scenarios used for model evaluation.

Scenario	Problem	Instances	Algorithms	Features	Unsolved Instances	Proportion Unsolved Instances	Proportion Missing Evaluation	Cutoff
ASP-POTASSCO	ASP	1294	11	138	82	0.06	0.20	600.0
CPMP-2015	CPMP	527	4	22	0	0.00	0.28	3600.0
CSP-Minizinc-Time-2016	CSP	100	20	95	17	0.17	0.50	1200.0
MAXSAT12-PMS	MAXSAT12	876	6	37	129	0.15	0.41	2100.0
MIP-2016	MIP	218	5	143	0	0.00	0.20	7200.0
QBF-2011	QBF	1368	5	46	314	0.23	0.55	3600.0
QBF-2016	QBF	825	24	46	55	0.07	0.36	1800.0
SAT12-HAND	SAT12	767	31	115	229	0.30	0.67	1200.0
SAT12-INDU	SAT12	1167	31	115	209	0.18	0.50	1200.0

An instance is unsolved if no candidate algorithm solves the instance before the cutoff is reached. An evaluation of some algorithm on an instance is missing if the algorithm does not finish it's calculation before the cutoff is reached.

Further Evaluation Results

In the paper we were only able to give a brief overview over the results of our evalaution. Further resutls are shown in the following figures.

Figure 2 shows the results of our PAR10 evaluation for all considered scenarios.

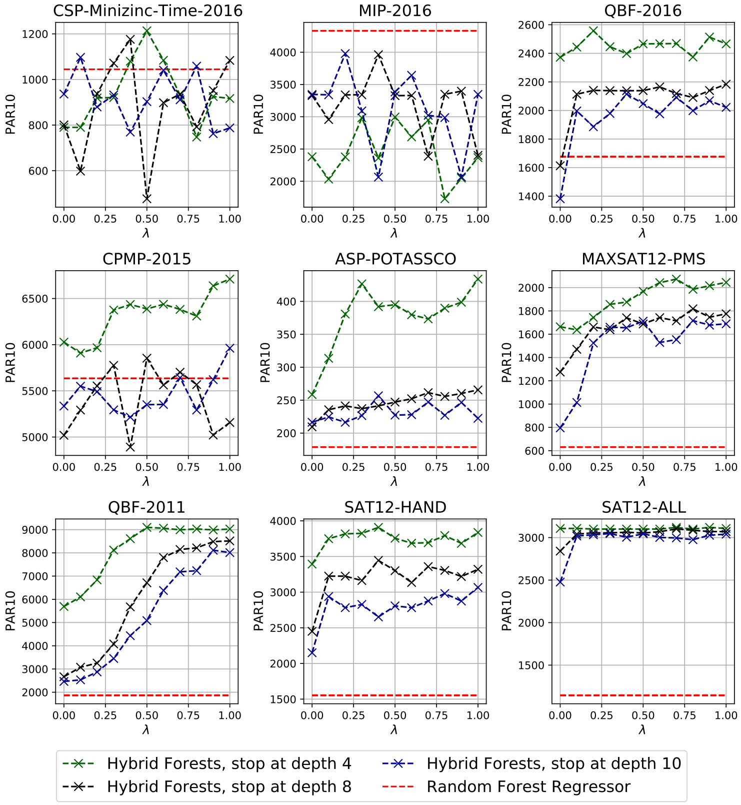

Figure 2: Quality Comparison of different HARRIS configurations with the Random Forest Regressor. The model quality is quantified with PAR10.

Figure 3 shows the results of our Kendall's Tau-b evaluation for all considered scenarios.

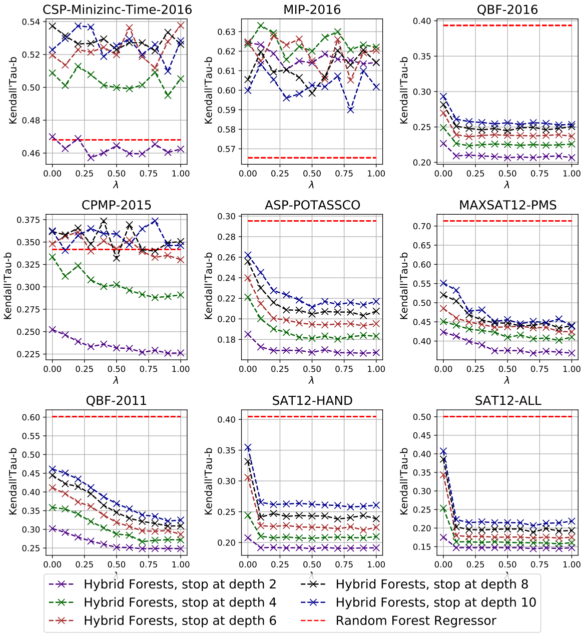

Figure 3: Quality Comparison of different HARRIS combinations with the Random Forest Regressor. The model quality is quantified with the Kendall's Tau metric.

Figure 4 shows the results of our evaluation of different tree depths in terms of the PAR10 number of the resulting algorithm selector.

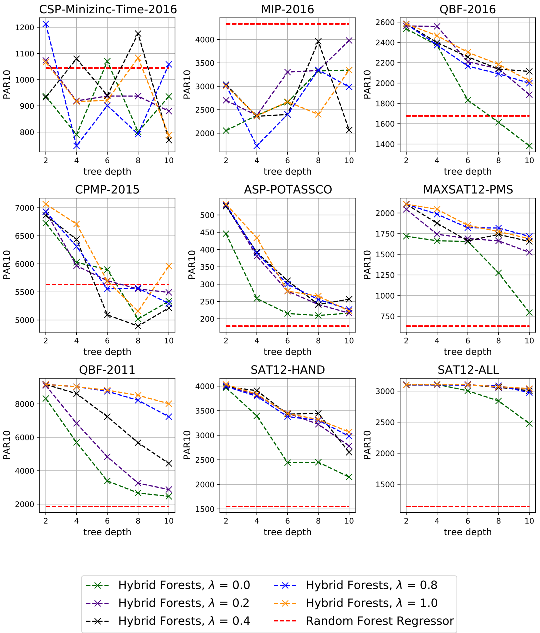

Figure 4: Evaluation of the stopping criterion's impact on the overall model quality. Note that the results indicate that HARRIS might improve for increasing depth on some scenarios.