Subproject B2: Configuration and Evaluation

Jonas Hanselle[1], Eyke Hüllermeier[2], Felix Mohr[3], Axel Ngonga[1], Mohamed Ahmed Sherif[1], Alexander Tornede[4], Marcel Wever[2]

1 Department of Computer Science, Paderborn University, Germany
2 Institute of Informatics, LMU Munich, Munich, Germany
3 Universidad de La Sabana, Chía, Colombia
4 Institute of Artificial Intelligence, Leibniz Unversity Hannover, Germany

Subproject B2 "Configuration and Evaluation" deals with methods and algorithms for the configuration and evaluation of software services in the OTF Computing scenario. During the three funding periods, various techniques have been developed and implemented for this purpose. Moreover, these techniques have been instantiated and evaluated on case studies from different domains: image processing, automated machine learning (AutoML), and question answering (QA) systems.

1 Introduction

Subproject B2 plays a central role within the CRC as a whole. Its task is to develop methods for the configuration of software services according to the requirement specifications provided by the user (cf. subproject B1). For this purpose, services traded on OTF markets are collected and assembled in an appropriate way. Before a service composition is executed, it will be analyzed for functional correctness (cf. subproject B3).

In the first period of the CRC, the focus of the subproject has been on

- the construction of a basic service configurator,
- the matching of services as an important part of the configuration process,
- the use of machine learning (ML) methods for the adptation of evaluation functions (in agreement with the users' preferences),
- the investigation of theoretical limits of the (automation of the) configuration process.

In the second period, subproject B2 concentrated on

- the extension of the configuration approach from a sequential to a sequential-hierarchical, template-based process,

jonas.hanselle@uni-paderborn.de (Jonas Hanselle), eyke@lmu.de (Eyke Hüllermeier), felix.mohr@unisabana.edu.co (Felix Mohr), axel.ngonga@upb.de (Axel Ngonga), mohamed.sherif@uni-paderborn.de (Mohamed Sherif), a.tornede@ai.uni-hannover.de (Alexander Tornede), marcel.wever@ifi.lmu.de (Marcel Wever)

- the improvement of ML-based adaptation techniques by new methods from the field of preference learning,
- the adaptation of the configuration to market changes (e.g., the offering of new services or changing user preferences),
- a better interlinking of the configuration and the execution phase.

The third period was dedicated to

- the integration of the user in the configuration process, and the realization of this process in an online manner,
- the increase of the efficiency of automatic service configuration by exchanging information between different but related configuration processes,
- the use of quality criteria of services as part of the objective function,
- the broadening of the evaluation by means of a complementary case study in the field of question answering systems.

While the focus of the subproject was mainly on conceptual and methodological contributions, the development of methods and algorithms has been accompanied by concrete implementations from the very beginning. Moreover, conceptual solutions have been instantiated and evaluated on case studies from different domains, starting with image processing in the first funding period. Later on, the instantiation of service configuration has been realized for the practically relevant case of machine learning functionality, with the vision to establish "OTF Machine Learning" as an extension of what is currently known as "Automated Machine Learning" (AutoML) [HKV19]. In the last funding period, question answering systems have been added as a third application domain.

2 Highlights and Lessons Learned

In the following, we give an overview of the most important achievements of the subproject and summarize the key results that have been accomplished during the three funding periods of the CRC.

2.1 Domain-Independent Service Composition

During the first phase of the project, the main focus was on the functional aspect of automated service composition without a commitment to a specific domain. In this scenario, the user provides a formal specification of the functional requirements in terms of inputs, outputs, preconditions, and effects (IOPE) of the desired service. It is assumed that there is a set of existing services with the same type of descriptions that can be used to create the new desired service. The preconditions define types of and potential relationships between inputs. The effects describe conditions that the service guarantees to hold on the inputs or outputs after execution. The language used to describe preconditions and effects is a decidable subset of first-order logic. In particular, they serve to describe the *meaning* of outputs with respect to inputs. A simple example is a currency converter service that takes

a value *x* in EUR and returns its equivalent *y* in USD, and the effect could be described by *EUR2USD(x,y)*.

This task is an extended version of the classical planning problem. The services correspond to planning operators, instances of which can be connected into a chain of service calls, which correspond to actions in the classical planing setup. From this viewpoint, a service composition is a *plan*, and the initial state is the preconditions specified in the query, and the goal state is the effect specified in the query. The first crucial aspect that differentiates automated service composition from classical planning is that the operators can create new objects (the outputs), which is not supported in classical planning. A second difference is that the quality of a solution is not a scalar but a vector. In classical planning, the cost of a plan is the sum of the (scalar) costs of actions. However, in a service composition, several qualities of service (QoS) such as throughput, availability, privacy, etc. have to be considered in addition to the price.

Importantly, this type of problem is much more challenging than the much more commonly studied problem of pure QoS optimization over a *pre-defined* service workflow, where it is assumed that the general controller of the desired service is already implemented and connects to yet unspecified components through fixed interfaces. For each interface, a finite set of candidates is assumed to be available and can be plugged into the solution. The goal is then to pick the best combination of such candidates that, when put together, optimize the overall QoS of the configuration. This is a *configuration* of a controller, and the decisions the agent need to make to set up a solution are a strict subset of the decisions the agent has to make in the case of automated service composition.

Within this phase, we were the first to propose a planning algorithm that is capable of automatically creating service compositions based on functional descriptions in which the effects relate outputs to inputs while optimizing QoS. The approach is based on backward planning [MJB15] and is the first algorithm of its kind that is able to compose services not only based on the types of the inputs and outputs (monadic preconditions and effects), but can work with preconditions and effects of any arity. Starting from the goal definition with an empty composition, it tries to prepend service calls to the current composition that resolve at least one open requirement. Such a prepended service call typically comes with its own preconditions, which are added to the agenda unless they are provided in the initial state (preconditions warranted by the client in the query).

Another highlight of the approach is that it is able to prune nodes from the search space, if they are redundant in some way to make the search more efficient. First, it cuts nodes that encode compositions containing a service twice with the same inputs. Second, the algorithm prunes nodes of compositions that have a precondition that includes the precondition of one of its own subcompositions. That is, a node is pruned if it is associated with a composition whose preconditions are a superset of the preconditions of another composition.

In spite of its innovative aspects, the background search still suffered from a number of inefficiencies, which could be overcome by the development of a partial order planning algorithm [Moh]. The main advantage of a partial order planning (POP) compared to forward or backward planning is that it takes into account that the only constraint on the order of the planning operators is the flow of data. This specific property of the automated service composition problem implies that a huge number of serialized compositions are

equivalent from both the functional and the QoS viewpoint. This implies that the search space in forward or backward composition contains a huge number of mirrors, which are avoided in POP. In POP, orders in the plan are only partially fixed as far as *necessary*, based on the preconditions and effects of the operators used.

A further limitation of these compositions is that they cannot contain conditional paths let alone loops. To overcome this limitation, we proposed a template approach for loops in which a general structure with generic preconditions and effects is defined [MW15]. A replacement of service placeholders leads to a concrete service instantiation with concrete preconditions and effects. During a regular composition process, this mechanism can be invoked as a subroutine to create services with non-linear control flows on the fly.

All of the above composition approaches are based on orchestration. That means that it is assumed that there is a central instance that controls the service invocations and the data flow between them. In contrast to this, a choreography approach does not have a central controller, but all the participants of a composition are told about where, i.e., to which other services, they should send their output for a specific composition. This decentralized execution of service compositions can lead to enormous performance improvements, because the data has to travel very short distances (maybe even within the same compute center) compared to a centralized approach.

Based on the previous work, a choreography-based approach was developed in [JK16]. The main challenge in this approach is to trigger the execution of a service decentrally as soon as all the data of a service has arrived. In this work, the logic of service composition execution is modeled through Petri nets, in which data is seen as a resource and services as transitions that consume and produce data. Needless to say, the service does not actually consume the data, but the semantics of Petri nets are used to model the behavior of such a service.

In an alternative research thread in this phase, we investigated the issue that purely formal service descriptions are usually not suffecient to capture the user expectations. A common example for this is image processing. At the symbolic level, it is virtually impossible for the user to specify the desired transformation of an image. Instead, it can be sensible to show different proposals to the user and ask him for feedback. Such feedback can be binary, i.e., the user is rather satisfied than not with a result, or one provides several options and lets the user rank the alternatives. From this feedback, it is then in principle possible to learn which services (and their configurations) the user prefers over others. The main challenge is here to identify to which of the services within a composition to attribute a good or bad ranking.

To address this problem and to learn the relevance of a specific service (for a particular user in a concrete context), temporal difference (TD) learning was used [JM15]. We recognize that every service is, in a specific context, associated with a latent reward that is neither known before nor cannot be observed directly. However, if one interprets the set of partial compositions as the state space of an MDP, it is possible to learn the appropriateness of the components through TD learning. This is because TD learning propagates back the final evaluations to the state over time and, in this way, indirectly assigns ratings to partial compositions.

In this last approach, the composition technique deviates from the other techniques in that a forward search is adopted based on rules or tasks. The original problem is still to

convert an initial condition into a goal condition, but the services are no longer explicitly equipped with specific preconditions and effects, except maybe the types of the inputs and outputs. To decide whether a service is suitable for a specific task, the approach uses the concept of rules, which can be seen as possible ways of solving tasks. This view is closely related to hierarchical planning, which is also the basis of the ML-Plan approach developed in the second phase. The composition problem is then described through a context-free grammar in which the initial state is the start symbol, non-terminal symbols encode tasks, and production rules encode how tasks can be resolved. Such a rule maps a task to a series (usually of length 1) of services and possibly a new non-terminal. It can hence be seen as a task composition.

In summary, the first phase of the project focused on automated composition of services into a new service that satisfies the functional requirements specified by the user. To this end, classical planing was extended to support the generation of new planning objects and to support vector-valued QoS optimization. Based on the observation that fully ordered planning leads to significant inefficiencies, an alternative approach based on partial order planning was developed, that significantly outperforms the previously developed backward search. These orchestration-based approaches have been modified and extended in order to support choreography-based compositions, the latter of which were achieved by the means of Petri nets. Orthogonal to these efforts, we investigated the potential of composition approaches that take into account the fact that many important aspects of even the functional behavior of services cannot be captured in symbolic encodings. This makes it necessary to propose to the user a set of potentially satisfying solutions, all of which comply with the formal requirements posed by the user, and to ask the user for feedback.

The insights and experiences gained during this first phase were crucial for the definition of the goals in the following phases. One of the most important insights was indeed the limitation of formal service specifications. One very prominent example of automated service composition, where formal specifications are of no use is, automated machine learning. At the formal level, the goal here is simply to find a machine learning pipeline. The challenge is, however, that such pipelines work differently well on different datasets, and a pipeline that works well on one dataset may not work well on another one. Among hundreds of possible pipelines and billions of their configurations, the goal is to find the best suited one suited according to a performance measure such as accuracy.

2.2 ML-Plan: Configuring Machine Learning Pipelines

As mentioned in the previous section, a particularly interesting and practically relevant domain is machine learning. In this domain, services can process and model data in a wide variety of ways for different tasks. While there are many different functionally equivalent services for a task, the real interest is in finding services that satisfy certain non-functional properties, such as high accuracy and/or low prediction time. Since the non-functional properties can vary widely for different datasets and thus which service is best suited, it is important to determine the most appropriate service for each data set, which in turn requires expertise in the field of machine learning.

The need for applications with machine learning techniques has increased rapidly, especially in recent years, and cannot be satisfied by the available experts in this field. This

situation gave rise to the vision of automated machine learning (AutoML), which deals, among other things, with the automated selection and parameterization of machine learning algorithms. These algorithms are often arranged in a so-called pipeline where first the data is pre-processed and transformed in a certain way and eventually passed to a learning algorithm. Choosing the right algorithms also in the right order is of high importance to obtain the best possible results with respect to the non-functional requirements. In other words, AutoML deals with the automated and personalized delivery of machine learning applications.

Considering machine learning algorithms as services, the search for suitable machine learning algorithms or services fits seamlessly into the setting of OTF Computing, where a machine learning service or a composition of machine learning services on the requirements should be provided according to the user. While existing AutoML tools rely on techniques such as Bayesian optimization [THHL13; FKE+15], genetic programming [OBUM16; GV19] or reinforcement learning, we have continued our work with planning algorithms from the previous funding phase. More specifically, we have developed an AutoML system based on the paradigm of HTN-planning and using a best-first search for the search, which borrows concepts from the Monte Carlo tree search for the node evaluation.

Another problem is that with the ongoing search for suitable machine learning services, these adapt too much to the training data provided and do not generalize as well, which results in lower accuracy on new, unseen data. To avoid this effect, we propose a two-step AutoML process with ML-Plan [MWH18b], in which part of the training data is retained for a later final candidate selection. In a first phase, a pool of promising candidates can be put together with the reduced training data set and a final candidate can later be selected from this pool with the data that has not yet been used. In this way, the previously described effect, which is also referred to as overfitting in the literature, can be largely avoided.

In [MLHW18], we first developed an extension of HTN-planning to programmatic task network planning (PTN-planning), which can be used to combine the static search space model with dynamically determined ones. Information can be entangled to make the search space dependent on certain dynamic state properties. In addition, we compared ML-Plan with the state-of-the-art approaches and were able to determine a competitive performance for ML-Plan. A key component for the success of ML-Plan is the search space modeling based on HTN-planning, more specifically PTN-planning. The search space naturally exhibits hierarchical structures, e.g., learning algorithms that wrap other learning algorithms, for example, to tackle subproblem of the original problem. Furthermore, machine learning algorithms typically expose so-called hyperparameters which are parameters of the learning algorithm that may impact the learning behavior. Depending on which machine learning algorithm is chosen, different hyperparameters need to be optimized, introducing additional hierarchical structures and constraints. A schematic illustration of these hierarchical structures is shown in Figure 21 as well as in the following section (cf. Figure 23).

As already pointed out before, HTN-planning allows to capture those hierarchical structures and dependencies in a very natural way. In Figure 22 a search tree induced by HTN-planning and fast forward decomposition is shown, where an initial complex task is iteratively refined by other complex tasks or primitive task via so-called methods until only primitive tasks are left. The search space model follows a divide-and-conquer approach so that complex tasks are step-by-step broken down to (hopefully) simpler

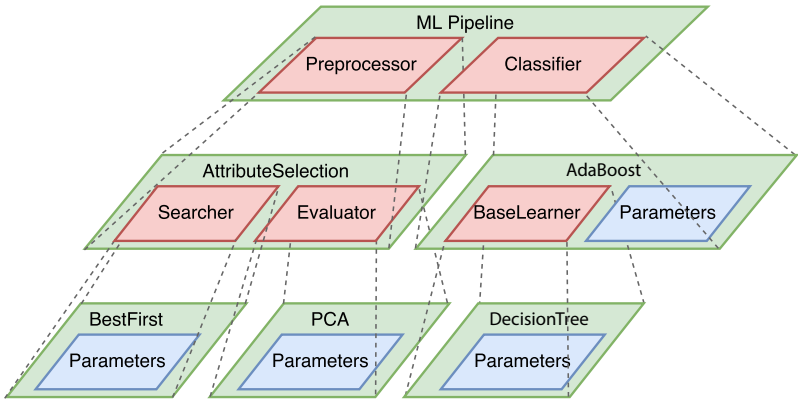

Figure 21: *A schematic illustration of a machine learning pipeline consisting of a preprocessing step and a learning algorithm.*

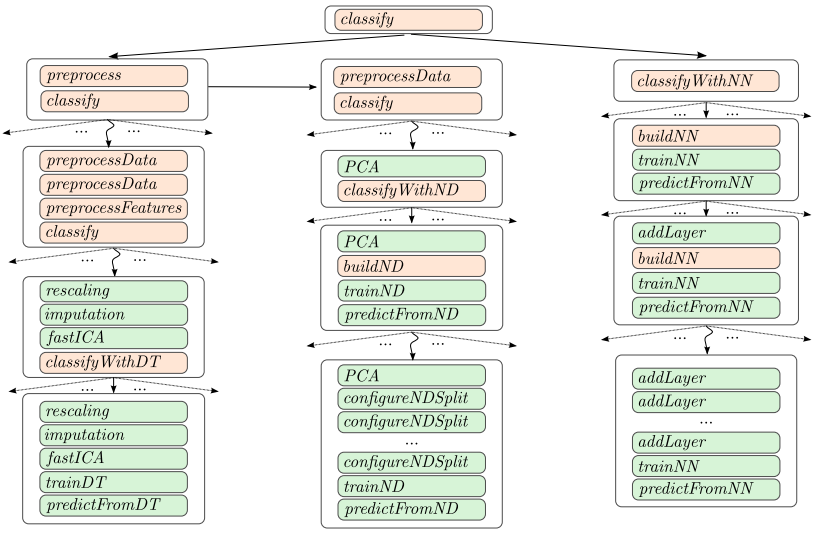

Figure 22: *Derivation of pipelines via hierarchical planning. Complex tasks are colored in red and primitive tasks in green. Arcs indicate methods.*

tasks until all complex tasks have been refined by primitive tasks eventually. Intuitively speaking, ML-Plan tries to imitate a human developer faced with a complex task of providing a machine learning pipeline for a given problem. Then, this abstract task is decomposed step-by-step to smaller abstract tasks such as choosing a learning algorithm and preprocessing algorithms until all decisions regarding machine learning algorithms and their hyperparameter values have been made.

ML-Plan has served as a starting point for several subsequent works. In a first sequel, we extended the search space of ML-Plan from the commonly configured two-step pipelines, involving a single pre-processing algorithm and a learning algorithm, to pipelines comprising a potentially unlimited number of pre-processing algorithms arranged in a tree-shaped structure and again a learning algorithm. In this way, ML-Plan is capable of building more sophisticated data transformations to pre-process the given data. Furthermore, while ML-Plan was originally developed for binary and multinomial classification tasks, it has been extended to regression, multi-label classification [WMH18; WMTH19], and more recently, remaining useful lifetime estimation in the realm of predictive maintenance [TTW+20].

Even in these settings, which are sometimes quite different from the original classification setting, it has shown strong performance, rendering ML-PLan a relatively flexible AutoML framework. To prepare ML-Plan for its deployment in an OTF market, where machine learning algorithms are provided in the form of cloud services which are computed in a distributed system, ML-Plan was also extended to work with services in a distributed environment [MWHF18; MWH18a].

Beyond scientific successes and academic publications, ML-Plan was a key component in the proof-of-concept project, a demonstrator joining various subprojects of the collaborative research center. More specifically, ML-Plan was used in the implementation of the on-the-fly provider for the configuration of the machine learning services. Therefore, ML-Plan represents the beating heart of one of the considered on-the-fly scenarios, i.e., on-the-fly machine learning [MWTH19].

2.3 Automated Configuration of Multi-Label Classifiers

Another relevant learning problem, which is also extremely interesting from an AutoML point of view, is the so-called multi-label classification. Here, in contrast to conventional, single label, classification problems (SLC), instances can be associated not only with one class, but with several classes at the same time. Consequently, instead of mapping from $\mathcal{X}$ to $\mathcal{L}$, where $\mathcal{X}$ is the instance space and $\mathcal{L}$ is the set of class labels, models map to the power set of $\mathcal{L}$, i.e., all possible label combinations. While single-label classification tries to learn primarily dependencies between $\mathcal{X}$ and $\mathcal{L}$, much of the MLC literature also tries to exploit dependencies between labels, i.e., between elements in $\mathcal{L}$, in order to increase the generalization goodness.

Based on methods for SLC, a diverse repertoire of MLC methods has been developed over time. One strain of the literature adapts SLC models and/or learning algorithms for the MLC setting, so-called algorithm adaptation approaches. Alternatively, MLC problems are transformed into one or multiple SLC problem(s) such that in turn already well-studied SLC methods can be applied to the induced problems.

An exemplary selection of algorithms constituting a multi-label classifier is illustrated in Figure 23.

From an AutoML perspective, problem transformation methods need to be configured with an SLC method as a base learner, and the choice of both the problem transformation method as well as the baselearner depends on the task in question, i.e., the dataset and loss function. Hence, the search space for automatically selecting algorithms and optimizing their hyperparameters is a multiple of the search space of SLC, which is reported already huge, since the search space for SLC is included for each MLC problem transformation method. Also in this direction of extending AutoML methods it is questionable to what extent the already proposed methods can be applied to the MLC setting. More precisely, questions of scalability arise.

Another question is how to design a fair and meaninigful comparison. In the literature, oftentimes complete AutoML systems are proposed that combine an optimization method with a custom search space definition and a module for evaluating solution candidates. However, we are interested in how well optimization methods scale with the increasing

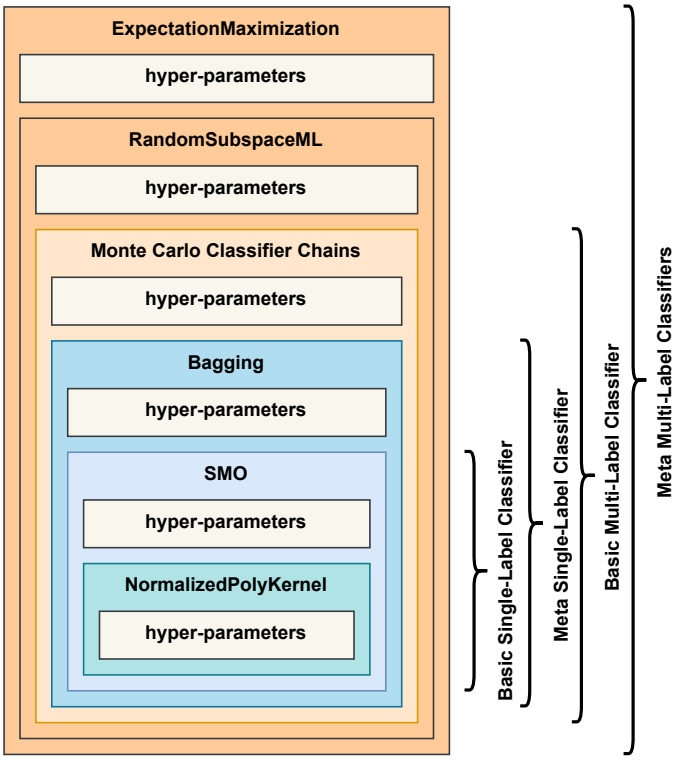

Figure 23: *A schematic illustration of the structure of a multi-label classifier following a problem transformation strategy. Such a multi-label classifier may comprise multiple "layers" of algorithms where for each layer one can in principle choose between different algorithms of that type.*

search space size, and how well they can handle the AutoML for MLC problem. To be clear, we are not interested in an AutoML system as such, but to investigate which optimization method is best suited for this particular setting. This requires to unify certain design decisions as for instance the search space and the evaluation module, as well as other more technical design decisions: parallelization, degree of parallelization, memory constraints, etc. Moreover, AutoML systems often work with different ML libraries as a backend, i.e., some systems may work with scikit-learn [PVG+11], some with WEKA [HFH+09], and again others may work with mlr3 [LBR+19]. However, implementations of the same methods may differ significantly and oftentimes some methods are not even available in all libraries. Thus, for a fair comparison of optimization methods they should be benchmarked in a unified environment such that all the optimization methods use exactly the same implementations of the solution candidate evaluation and operate on the same search space. We interpret the latter in a way that all optimization methods may potentially encounter every candidate another optimization method may be able to find. Of course, the precise specification of the search space may differ from optimization method to optimization method.

In [WTMH21] we present answers to both questions: How to benchmark different optimization methods proposed for AutoML in the SLC setting and to what extent those methods appear to scale well with the specifics of the MLC setting. To this end, we first propose a benchmarking framework which, in principle, can be used to benchmark any type of combined algorithm selection and hyperparameter optimization problem setting.

Meaning the benchmark is not limited to MLC problems but can be used for SLC problems as well. The implementation of the benchmark framework is cross-platform and can be used to integrate, for example, implementations in Python and Java. This allows optimization methods implemented for different platforms to work with the same evaluation module without the need for re-implementing the optimization method for another platform. Beside the software, the benchmark also comes with constraints on the hardware to use and a limit on the time budget. Regarding the time budget the proposed benchmark restricts both the total runtime and the runtime for evaluating a solution candidate. The latter is an important aspect since the time allowed for evaluating a single solution candidate implicitly prunes slower candidates from the search space and thus different approaches would again operate on different search spaces.

Based on this benchmark framework, an extensive empirical study was conducted comparing 6 optimization methods for a total of 24 datasets with 10 different train-test splits each. Furthermore, we considered a total of 3 performance measures for optimization, which generalize the F1 measure in three different ways from the SLC to the MLC setting. Generally speaking, we found that all the six optimization methods are struggling with the MLC setting, taking quite some time to return reasonable solutions. In fact, on average, only after 4 hours the optimization methods reach a level which is at least close to the best result that can be obtained after 24h. Most interestingly, we find that rather greedy approaches such as the optimization method employed in our AutoML system ML-Plan and Hyperband, an optimization method based on the successive halving paradigm, appear to perform overall best. We hypothesize that this is due to the fact that the choice of the algorithm is more important than tuning the hyperparameters. Furthermore, the evaluation of solution candidates is typically more costly so that multi-fidelity optimization appears to be indeed a crucial characteristic for an optimization method aiming to automate multi-label classification.

2.4 Censored Data in Algorithm Selection

Algorithm selection (AS) [Ric76; KHNT19] is the task of finding the most suitable algorithm for a given problem instance of an algorithmic problem domain, such as the Boolean satisfiability problem (SAT) or classification (machine learning). Typically, suitability is measured in terms of a performance measure, which characterizes some sort of solution quality which shall be maximized or some kind of cost which shall be minimized. One of the most prominent performance measures is the runtime an algorithm needs in order to return a valid solution, which is of special interest in the domain of hard combinatorial problems such as SAT or integer optimization. A common approach towards per-instance algorithm selection is the use of machine learning, in which runtime measurements of algorithms from previous runs are used in order to estimate the algorithms' runtimes on new, previously unseen problem instances.

AS is of particular interest for the CRC, as it is a subproblem of AutoML that can be solved with conceptually simpler approaches making it easier to study certain properties associated with it. In particular, in this section, we elaborate on the problem of so-called right-censored training data [KK10], which can be found quite frequently in AS, algorithm configuration (AC) [SBT+22], hyperparameter optimization (HPO) [FH19; BBL+21] and AutoML problems.

In order to deduce estimators for the algorithms' runtimes, one generally assumes that problem instances can be represented in terms of characteristics, so called features or meta-features in the context of meta-learning [Van18], that should be correlated with the performance measure, in this case runtime. Correspondingly, the training data needed to learn such estimators consists of feature descriptions of problem instances and the runtimes achieved by various algorithms on this particular problem instance. Since algorithms for such hard problems may exhibit extremely long runtimes, they are generally not run for an indefinite amount of time until they eventually terminate, but are rather terminated externally once the time exceeds a certain threshold T, called cutoff. Thus, the training data contains right-censored datapoints, i.e. observations of which we do not know the exact runtime, but only a lower bound T.

Naturally, such a right-censored datapoint is not a scalar value, but rather a right-open interval and thus cannot be used as a standard regression training datapoint, but has to be treated differently. The community has suggested a variety of approaches in the literature of how to handle such datapoints in the context of AS, AC, HPO and AutoML [XHHL07; HTWH20; HTWH21; HHL11; ELH+18; EHM+20].

The simplest approach for dealing with such censored samples is to ignore them all together, which, however, comes with a loss of information. Although the censored data cannot be used directly as training points, they do contain information, which should be incorporated. Another simple strategy is imputation. For example, in the case when algorithm selectors are evaluated based on the so-called *PAR10* score [KHNT19] - a penalized version of runtime - censored samples are commonly replaced by the cutoff time T or a multiple thereof, such as $10\,T$. Obviously, such imputations can easily result in a strong bias of the model learned on such data [Gre05]. A more sophisticated approach to impute right-censored data developed by [SH79] samples from a truncated normal distribution and is leveraged by many AS and AC approaches (e.g. [XHHL07; ELH+18]). However, as we show in [TWW+20b], these approaches do not necessarily improve upon the naive imputation schemes discussed above.

All of the approaches presented above share the problem that they are rather indirect solutions for dealing with censored data and as such, come with the disadvantages noted above. In contrast to that, methods from the field of survival analysis [KK10] (SA) can inherently deal censored datapoints and are thus much more suited for the kind training data often found in AS, AC, HPO and AutoML problems. Correspondingly, in [TWW+20a] we adapt rigorous statistical SA methods for constructing algorithm selectors using partially right-censored runtime data. In particular, using random survival forests [IKBL08], we learn algorithm runtime distributions, which we then leverage to obtain an estimated algorithm runtime.

In order to derive a point estimate of the runtime of an algorithm from the runtime distribution, a first natural choice is the expectation of the distribution, i.e., the expected algorithm runtime. While this does indeed yield reasonably good algorithm selections in many practical cases, the expected value can be overly optimistic for the selection of an algorithm, if the performance measure penalizes timeouts of algorithms excessively as is the case for the *PAR10* score.

To mitigate overly optimistic selections in such cases, we advocate for a decision-theoretic selection approach that incorporates the concept of risk aversion, which coincides with

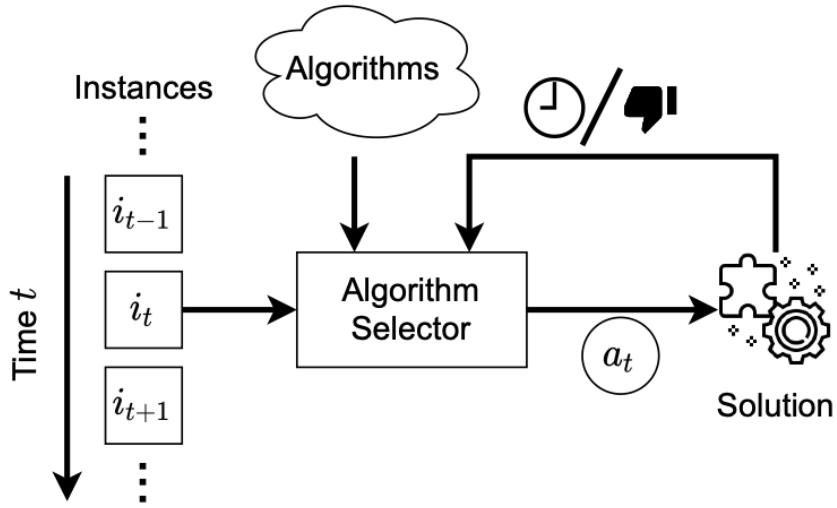

Figure 24: *General process of the online algorithm selection setting. In each round, the selector is asked to select an algorithm, which is then evaluated using the performance measure resulting in an evaluation result fed back to the learner. Based on this result, the learner can update its internal model.*

timeout aversion in our case. To this end, we compute the expectation of a risk-averse loss function applied to the random variable modeling the runtime of an algorithm instead of directly computing the expectation of that random variable.

Combining the concepts of SA and decision-theoretic risk aversion allows us to achieve state-of-the-art algorithm selection performance on the de-facto standard AS benchmark, called ASLib [BKK+16], beating the hitherto state of the art by roughly 15%.

Standard AS considers an offline problem in the sense that one usually assumes a phase prior to the actual application of the selector, where any form of data generation and learning can take place. In contrast, online AS (OAS) weakens this assumptions and instead aims at selectors, which are learned and updated online in a round-wise manner without any prior learning phase. For this purpose, in each round, the selector is asked to select an algorithm, which is then evaluated using the performance measure, resulting in an evaluation result fed back to the learner. Based on this result, the learner can update its internal model. The process is depicted in Figure 24.

The problems associated with censored data are even more prominent in the online case, as one only obtains a single datapoint each round, which might even be censored. If censored datapoints are, for example, dropped in such cases instead of incorporated into the learning process, the model cannot be updated and no learning takes place for that round.

Unfortunately, the SA methods we previously discussed cannot be used in the online setting, as the vast majority of such methods are inherently designed as offline approaches. For example, Cox' proportional hazards model [Cox72] leverages the Breslow estimator to estimate the baseline survival function, which has to store all data previously seen in the form of risk-sets [Bre72]. Naturally, storing all previously seen data is not a viable approach in an online setting as the storage complexity grows with the time horizon in such a case.

As an alternative solution, in [TBH22] we suggest to adapt well-known bandit algorithms to OAS and runtime-oriented loss functions. In particular, we investigate the bias incurred by directly applying a UCB strategy [ACF02], when censored samples are dropped completely or imputed with the cutoff T. The corresponding bias-correction terms result in extremely large confidence bounds that do no longer yield reasonable algorithm selections in practice. To alleviate these problems, we propose a Thompson sampling approach [Tho33; RRK+18] that is adapted to losses strongly penalizing algorithm timeouts and imputes censored samples by an online variant of the Schmee&Hahn approach [BJ79].

With this approach we can improve upon existing OAS approaches in terms of selection performance while featuring a runtime and space complexity independent of the time horizon - a property that other existing approaches do not offer.

2.5 Adagio - Automated Data Augmentation of Knowledge Graphs Using Multi-Expression Learning

The creation of an RDF knowledge graph for a particular application commonly involves a pipeline of tools that transform a set of input data sources into an RDF knowledge graph in a process called dataset augmentation. The components of such augmentation pipelines often require extensive configuration to lead to satisfactory results. Thus, non-experts are often unable to use them. In this section, we present the basic idea behind Adagio [DSN22], an efficient supervised algorithm based on genetic programming for learning knowledge graph augmentation pipelines of arbitrary length. Our approach uses multi-expression learning to learn augmentation pipelines able to achieve a high F-measure on the training data. Our evaluation suggests that our approach can efficiently learn a larger class of RDF dataset augmentation tasks than the state of the art while using only a single training example. Even on the most complex augmentation problem we posed, our approach consistently achieves an average F_1-measure of 99% in under 500 iterations with an average runtime of 16 seconds.

RDF Dataset. An *RDF* dataset D is a set of triples $\{(s, p, o) \in (\mathcal{R} \cup \mathcal{B}) \times \mathcal{R} \times (\mathcal{R} \cup \mathcal{B} \cup \mathcal{L})\}$, where $\mathcal{R}$ is the set of all RDF IRI resources, $\mathcal{B}$ is the set of all RDF blank nodes and $\mathcal{L}$ is the set of all RDF literals. We denote the set of all RDF datasets as $\mathcal{D}$.

Dataset Operators. A function $\mathbb{O}_{(n,m)} \colon \mathcal{D}^{n+1} \to \mathcal{D}^m$ is called a *dataset operator*. Intuitively, a dataset operator $\mathbb{O}_{(n,m)}$ processes n input datasets using another dataset C as configuration to produce m output datasets. We call n the in-degree and m the out-degree of $\mathbb{O}_{(n,m)}$ and will resort to writing just $\mathbb{O}$ when the lack of their specification will incur no loss of generality. Given integers $i \in [1, n]$, $j \in [1, m]$, we call the ith argument of $\mathbb{O}_{(n,m)}$ and the jth component in the output of $\mathbb{O}_{(n,m)}$ the in-port i and out-port j, respectively. The set of all dataset operators is denoted as $\mathbb{O}$.

Augmentation Graphs. An augmentation graph $G = (\mathbf{O}, \mathbf{E}, \mathbf{L}, \mathbf{M})$ is a directed acyclic labeled multigraph where $\mathbf{O}$ is a set of dataset operators, which act as vertices; $\mathbf{E}$ is the set of edges, which represent flow of data; $\mathbf{L}$ is the edge labeling function, which defines mappings between dataset operator out-ports and in-ports for a given edge; and $\mathbf{M}$ is a mapping from vertices to configuration datasets. We call the subsets of vertices with 0 in-degree *root vertices*. *leaf vertices* are the and subsets of vertices with 0 out-degree.

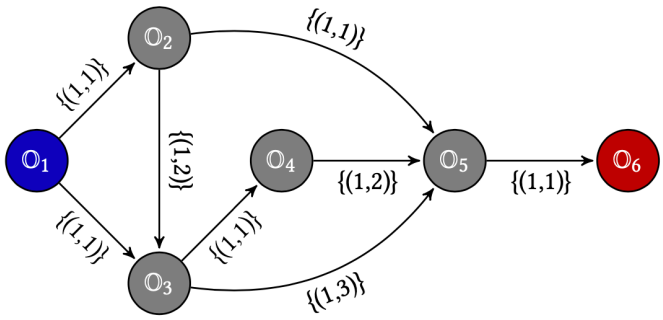

Figure 25: *Running example augmentation graph.*

All other vertices are *inner vertices*. Note that, per definition all root vertices of an augmentation graph must be dataset emitters, all leaf vertices must be dataset acceptors and all inner vertices must be augmentation operators. In our running example augmentation graph in Figure 25, we coloured all root vertices blue and all leaf vertices red. The intuition behind $\mathbf{L}$ is that, given $e = (\mathbb{O}_1, \mathbb{O}_2)$, we need to define which of $\mathbb{O}_1$'s out-ports map to which of $\mathbb{O}_2$'s in-ports. For instance, in our running example in Figure 25, the label set on the edge between $\mathbb{O}_4$ and $\mathbb{O}_5$ indicates that $\mathbb{O}_4$'s first output dataset is the second argument to $\mathbb{O}_5$. To evaluate an augmentation graph, we first obtain the RDF datasets as output of the root vertices in $\mathbf{O}_r$. These datasets then flow through the graph as specified by the semantics we associated with the edge set $\mathbf{E}$ and the label multiset $\mathbf{L}$. Whenever a dataset operator $\mathbb{O}_{(n,m)} \in \mathbf{O}_i$ has received all its n input datasets, it is evaluated using $\mathbf{M}(\mathbb{O}_{(n,m)})$ as its last argument. The flow through the graph continues until eventually all vertices have been evaluated. We call an augmentation graph G *linear* if there exists at most a single path between $\mathbb{O}_1$ and $\mathbb{O}_2$; *semi-linear* if there is a pair of vertices $u, v \in \mathbf{O}$, $u \neq v$ for which there exist multiple paths from u to v; *confluent* if it has multiple root vertices and exactly one leaf vertex; *inherently confluent* if it is confluent and it only contains confluent augmentation operators, *general* otherwise.

Augmentation Tables. An augmentation table $\mathbb{T}$ is a condensed linear representation for inherently confluent augmentation graphs based on column tables [KP98]. The idea behind this representation is that each row represents one dataset operator. We can go through this table from top to bottom and evaluate the dataset operators which correspond to a row i using only the results of rows 1 to $i - 1$. Since dataset acceptors produce no output, they are omitted in this representation for the sake of simplicity.

Let $G = (\mathbf{O}, \mathbf{E}, \mathbf{L}, \mathbf{M})$ be an inherently confluent augmentation graph. Moreover, let $N(\mathbf{O}) := \max\{n \mid \mathbb{O}_{(n,m)} \in O\}$ denote *the maximum in-degree* in $\mathbf{O}$. An augmentation table is a table with $3 + N(\mathbf{O})$ columns and $|\mathbf{O}|$ rows, where the first column contains dataset operators, the second column contains configuration datasets and the third column contains the in-degrees of the dataset operators in the first column. The last $N(\mathbf{O})$ columns contain the indices of the rows used as input to the corresponding dataset operator. Given an augmentation table $\mathbb{T}$, we write $\mathbb{T}_i$ and $\mathbb{T}_{i,j}$ to refer to the ith row and the jth column in the ith row of $\mathbb{T}$, respectively. Applying this representation to our running example augmentation graph in Figure 25 gives the augmentation table depicted in Table 1. The algorithm for the computation of an augmentation table from a given inherently confluent augmentation graph is given in [KP98].

Table 1: *Running example augmentation table.*

$\mathbb{T}_1$:	$\mathbb{O}_1$	C_1	0	0	0	0
$\mathbb{T}_2$:	$\mathbb{O}_2$	C_2	1	1	0	0
$\mathbb{T}_3$:	$\mathbb{O}_3$	C_3	2	1	2	0
$\mathbb{T}_4$:	$\mathbb{O}_4$	C_4	1	3	0	0
$\mathbb{T}_5$:	$\mathbb{O}_5$	C_5	3	2	4	3

We call a row within an augmentation table an *output row*, if it is not used as input to a subsequent row. Note that output rows always correspond to dataset acceptors and that our previous definition of augmentation tables allows for only a single output row, as our augmentation tables must be isomorphic to inherently confluent augmentation graphs.

Multi-Expressive Augmentation Tables. A *multi-expressive augmentation table* is a generalized augmentation table that has more than one *output row*. Note that, any row in a multi-expressive augmentation table can be seen as an output row by just disregarding all rows below it. Given such a *reference output row* in a multi-expressive augmentation table, we can derive a normal augmentation table by following the procedure introduced in [DSN22].

Problem Definition. The problem under study is to find an adequate enrichment graph for a given training example. We restrict ourselves to learning the subclass of inherently confluent enrichment graphs. We will furthermore restrict our study to enrichment graphs where the maximum in-degree of the involved enrichment operators and the number of involved dataset emitters are at most two.

Learning Algorithm. The core of our learning approach is a population-based $(\mu + \lambda)$ *multi-expression learning* (MEP) algorithm[10] that is able to learn the subclass of inherently confluent augmentation graphs. Our population consists of a fixed number $\mu + \lambda$ of multi-expressive augmentation tables that we also call *genotypes*. All genotypes have a fixed number r of rows. Tournament selection [MG95] with a tournament size of 3 and a selection probability of 0.75 is applied for determining the mating pool and for selecting the survivors. We use 1-elitist selection [Mit98] to avoid a decrease in fitness. Both the offspring and the survivors are subject to mutation. The *offspring fraction* $\alpha = \frac{\mu}{\lambda}$, *mutation probability* σ and *mutation rate* ρ are hyperparameters that need to be determined experimentally. Therefore, we ran a series of grid searches on augmentation tasks with increasing difficulty and used our insights from previous runs to fine-tune the next. We report the final grid search results in Figure 26. These results suggest that the best set of hyperparameters are the *offspring fraction* $\alpha = 1$, the *mutation probability* $\sigma = 0.5$ and the *mutation rate* $\rho = 0.5$.

The algorithm will stop when either a perfect solution is found, a maximum number g of generations is exceeded or our convergence detection terminates it. As the results of RDF dataset augmentation are commonly expected to have a regular structure, we can expect the output dataset to be decomposable into a number of subgraphs that are isomorphic up to a certain error w.r.t. some structural graph similarity measure. We therefore regard the training examples as a list of source *concise bounded descriptions* (CBDs) and a single target CBD of sufficient depth to representatively capture the desired augmentation. This is

[10] μ is the *population size* and λ is the *recombination pool size*.

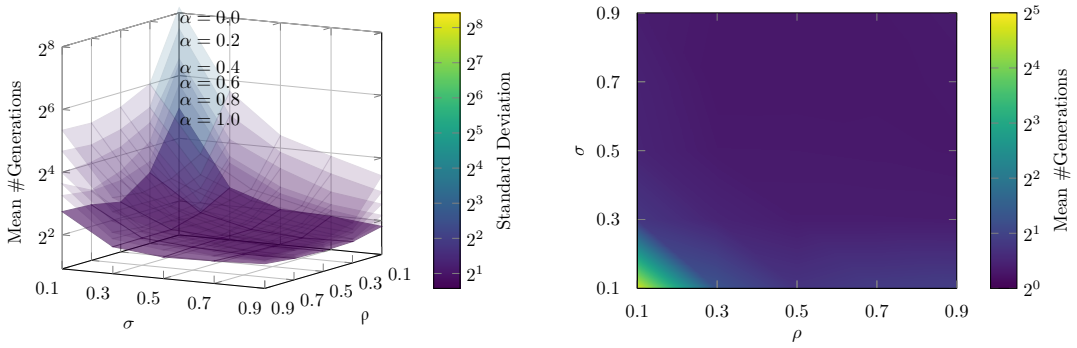

Figure 26: ***Hyperparameter Optimization Results.***

in accordance with the observation that a single pair of CBD often suffices for the training of augmentation pipelines [SNL15]. Note our choice to restrict the number of involved dataset emitters to at most two.

3 Conclusion and Outlook

Subproject B2 plays a central role within the overall architecture of the CRC, since the automatic configuration of software services is at the core of the OTF Computing paradigm. As such, it is closely connected to other subprojects, which either build on the service configurations provided by B2 (such as B3, which is responsible for the formal verification of configurations), or provide important input (most notably the service specifications produced by B1).

Starting with a relatively abstract, logic-based approach using formal specifications of functional requirements and techniques from automated planning for service composition, the focus of this subproject has shifted toward more concrete applications, such as automated machine learning (AutoML) and query answering (QA), and the use of data-driven methods for service composition. Interestingly, this has led to attributing a double role to machine learning, which served as a key methodology for automated service composition and, simultaneously, as an important use case.

Tackling the problem of automated service configuration for more concrete applications was mainly motivated by the observation that developing methods for this task, including the formal specification of requirements with preconditions and effects, the formalization of underlying domain knowledge, etc., is very difficult and hardly practicable on a completely generic level. Moreover, many criteria influencing the quality of a service, and hence being essential for the optimization of a composition, cannot be assessed in a purely formal way. Instead, a service composition must be realized and executed — for example, the quality of a machine learning pipeline can only be judged on an implementation level, by running it and applying it to a real data set.

The research conducted in the course of this subproject has not only contributed to the OTF framework of the CRC, but also created impact in other fields and scientific disciplines. A notable example is our work on AutoML, most visibly manifested in the AutoML

tool ML-Plan, which has been well received by the research community. In spite of this success, the vision we have for this field has not yet been realized: Going beyond the use of individual tools for AutoML, we envision "OTF Machine Learning" as a natural next step in the evolution of AI technology, and an important contribution to the democratization of AI. What we mean by OTF-ML is the realization of the OTF computing paradigm for the specific case of machine learning (or, more generally, data science) functionality, not restricted to individual software tools but including the entire compute and market infrastructure. We are convinced that this vision will become reality in the not too distant future, also thanks to the foundations that have been laid by this CRC.

Bibliography

[ACF02] Auer, P.; Cesa-Bianchi, N.; Fischer, P.: Finite-time analysis of the multiarmed bandit problem. In: *Machine Learning* 47 (2002), no. 2-3, pp. 235–256.

[BBL+21] Bischl, B.; Binder, M.; Lang, M.; Pielok, T.; Richter, J.; Coors, S.; Thomas, J.; Ullmann, T.; Becker, M.; Boulesteix, A.-L., et al.: Hyperparameter optimization: Foundations, algorithms, best practices and open challenges. In: *arXiv preprint arXiv:2107.05847* (2021)

[BJ79] Buckley, J.; James, I.: Linear regression with censored data. In: *Biometrika* 66 (1979), no. 3, pp. 429–436

[BKK+16] Bischl, B.; Kerschke, P.; Kotthoff, L.; Lindauer, M.; Malitsky, Y.; Fréchette, A.; Hoos, H. H.; Hutter, F.; Leyton-Brown, K.; Tierney, K.; Vanschoren, J.: ASlib: A benchmark library for algorithm selection. In: *Artif. Intell.* 237 (2016)

[Bre72] Breslow, N. E.: Contribution to discussion of paper by DR Cox. In: *Journal of the Royal Statistical Society* 34 (1972), pp. 216–217

[Cox72] Cox, D. R.: Regression models and life tables (with discussion). In: *Journal of the Royal Statistical Society* 34 (1972), no. 2, pp. 187–220

[DSN22] Dressler, K.; Sherif, M. A.; Ngomo, A.-C. N.: ADAGIO - Automated Data Augmentation of Knowledge Graphs Using Multi-expression Learning. In: *Proceedings of the 33rd ACM Conference on Hypertext and Hypermedia*. 2022.

[EHM+20] Eggensperger, K.; Haase, K.; Müller, P.; Lindauer, M.; Hutter, F.: Neural model-based optimization with right-censored observations. In: *CoRR* abs/2009.13828 (2020). arXiv: 2009.13828.

[ELH+18] Eggensperger, K.; Lindauer, M.; Hoos, H. H.; Hutter, F.; Leyton-Brown, K.: Efficient benchmarking of algorithm configurators via model-based surrogates. In: *Machine Learning* 107 (2018), no. 1, pp. 15–41.

[FH19] Feurer, M.; Hutter, F.: Hyperparameter optimization. In: *Automated machine learning*. Springer, Cham, 2019, pp. 3–33

[FKE+15] Feurer, M.; Klein, A.; Eggensperger, K.; Springenberg, J. T.; Blum, M.; Hutter, F.: Efficient and Robust Automated Machine Learning. In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. Ed. by Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; Garnett, R. 2015, pp. 2962–2970.

[Gre05] Greene, W. H.: Censored data and truncated distributions. In: *NYU Working Paper* (2005)

[GV19] Gijsbers, P.; Vanschoren, J.: GAMA: Genetic Automated Machine learning Assistant. In: *J. Open Source Softw.* 4 (2019), no. 33, p. 1132.

[HFH+09] Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I. H.: The WEKA data mining software: an update. In: *ACM SIGKDD explorations newsletter* 11 (2009), no. 1, pp. 10–18

[HHL11] Hutter, F.; Holger H. Hoos; Leyton-Brown, K.: Bayesian optimization with censored response data. In: *NIPS workshop on Bayesian Optimization, Sequential Experimental Design and Bandits*. Dec. 2011

[HKV19] Hutter, F.; Kotthoff, L.; Vanschoren, J., eds.: *Automated Machine Learning - Methods, Systems, Challenges*. The Springer Series on Challenges in Machine Learning. Springer, 2019.

[HTWH20] Hanselle, J.; Tornede, A.; Wever, M.; Hüllermeier, E.: Hybrid ranking and regression for algorithm selection. In: *KI 2020: Advances in Artificial Intelligence - 43rd German Conference on AI*. 2020, pp. 59–72.

[HTWH21] Hanselle, J.; Tornede, A.; Wever, M.; Hüllermeier, E.: Algorithm selection as superset learning: Constructing algorithm selectors from imprecise performance data. In: *Advances in Knowledge Discovery and Data Mining - 25th Pacific-Asia Conference, PAKDD 2021*. 2021, pp. 152–163.

[IKBL08] Ishwaran, H.; Kogalur, U. B.; Blackstone, E. H.; Lauer, M. S.: Random survival forests. In: *The annals of applied statistics* 2 (2008), no. 3, pp. 841–860

[JK16] Jungmann, A.; Kleinjohann, B.: Automatic Composition of Service-Based Image Processing Applications. In: *2016 IEEE International Conference on Services Computing (SCC)*. 2016, pp. 106–113

[JM15] Jungmann, A.; Mohr, F.: An approach towards adaptive service composition in markets of composed services. In: *Journal of Internet Services and Applications* 6 (2015), no. 1, pp. 1–18

[KHNT19] Kerschke, P.; Hoos, H. H.; Neumann, F.; Trautmann, H.: Automated algorithm selection: Survey and perspectives. In: *Evolutionary Computation* 27 (2019), no. 1, pp. 3–45.

[KK10] Kleinbaum, D. G.; Klein, M.: *Survival Analysis*. Vol. 3. Springer, 2010

[KP98] Kvasnièka, V.; Pospíchal, J.: Simple Implementation of Genetic Programming by Column Tables. In: *Soft Computing in Engineering Design and Manufacturing*. Ed. by Chawdhry, P. K.; Roy, R.; Pant, R. K. London: Springer London, 1998, pp. 48–56.

[LBR+19] Lang, M.; Binder, M.; Richter, J.; Schratz, P.; Pfisterer, F.; Coors, S.; Au, Q.; Casalicchio, G.; Kotthoff, L.; Bischl, B.: mlr3: A modern object-oriented machine learning framework in R. In: *J. Open Source Softw.* 4 (2019), no. 44, p. 1903.

[MG95] Miller, B. L.; Goldberg, D. E.: Genetic Algorithms, Tournament Selection, and the Effects of Noise. In: *Complex Systems* 9 (1995), no. 3.

[Mit98] Mitchell, M.: *An introduction to genetic algorithms*. MIT Press, 1998.

[MJB15] Mohr, F.; Jungmann, A.; Büning, H. K.: Automated Online Service Composition. In: *2015 IEEE International Conference on Services Computing*. 2015, pp. 57–64

[MLHW18] Mohr, F.; Lettmann, T.; Hüllermeier, E.; Wever, M.: Programmatic task network planning. In: *Proceedings of the 1st ICAPS Workshop on Hierarchical Planning*. 2018, pp. 31–39

[Moh] Mohr, F.: Towards automated service composition under quality constraints. PhD thesis. Dissertation, Paderborn, Universität Paderborn, 2016

[MW15] Mohr, F.; Walther, S.: Template-based generation of semantic services. In: *International Conference on Software Reuse*. Springer. 2015, pp. 188–203

[MWH18a] Mohr, F.; Wever, M.; Hüllermeier, E.: Automated Machine Learning Service Composition. In: *CoRR* abs/1809.00486 (2018). arXiv: 1809.00486.

[MWH18b] Mohr, F.; Wever, M.; Hüllermeier, E.: ML-Plan: Automated machine learning via hierarchical planning. In: *Mach. Learn.* 107 (2018), no. 8-10, pp. 1495–1515

[MWHF18] Mohr, F.; Wever, M.; Hüllermeier, E.; Faez, A.: (WIP) Towards the Automated Composition of Machine Learning Services. In: *2018 IEEE International Conference on Services Computing, SCC 2018, San Francisco, CA, USA, July 2-7, 2018*. IEEE, 2018, pp. 241–244

[MWTH19] Mohr, F.; Wever, M.; Tornede, A.; Hüllermeier, E.: From Automated to On-The-Fly Machine Learning. In: LNI P-294 (2019), pp. 273–274

[OBUM16] Olson, R. S.; Bartley, N.; Urbanowicz, R. J.; Moore, J. H.: Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science. In: *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, Denver, CO, USA, July 20 - 24, 2016*. Ed. by Friedrich, T.; Neumann, F.; Sutton, A. M. ACM, 2016, pp. 485–492.

[PVG+11] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V., et al.: Scikit-learn: Machine learning in Python. In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830

[Ric76] Rice, J. R.: The Algorithm Selection Problem. In: *Adv. Comput.* 15 (1976), pp. 65–118.

[RRK+18] Russo, D.; Roy, B. V.; Kazerouni, A.; Osband, I.; Wen, Z.: A tutorial on Thompson sampling. In: *Foundations and Trends in Machine Learning* 11 (2018), no. 1, pp. 1–96.

[SBT+22] Schede, E.; Brandt, J.; Tornede, A.; Wever, M.; Bengs, V.; Hüllermeier, E.; Tierney, K.: A Survey of Methods for Automated Algorithm Configuration. In: *Journal of Artificial Intelligence* (2022)

[SH79] Schmee, J.; Hahn, G. J.: A simple method for regression analysis with censored data. In: *Technometrics* 21 (1979), no. 4

[SNL15] Sherif, M. A.; Ngomo, A.-C. N.; Lehmann, J.: Automating RDF Dataset Transformation and Enrichment. In: *The Semantic Web. Latest Advances and New Domains*. Ed. by Gandon, F.; Sabou, M.; Sack, H.; d'Amato, C.; Cudré-Mauroux, P.; Zimmermann, A. Cham: Springer International Publishing, 2015, pp. 371–387

[TBH22] Tornede, A.; Bengs, V.; Hüllermeier, E.: Machine Learning for Online Algorithm Selection under Censored Feedback. en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36 (June 2022), no. 9. Number: 9, pp. 10370–10380.

[THHL13] Thornton, C.; Hutter, F.; Hoos, H. H.; Leyton-Brown, K.: Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In: *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*. Ed. by Dhillon, I. S.; Koren, Y.; Ghani, R.; Senator, T. E.; Bradley, P.; Parekh, R.; He, J.; Grossman, R. L.; Uthurusamy, R. ACM, 2013, pp. 847–855.

[Tho33] Thompson, W. R.: On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. In: *Biometrika* 25 (1933), no. 3/4, pp. 285–294

[TTW+20] Tornede, T.; Tornede, A.; Wever, M.; Mohr, F.; Hüllermeier, E.: AutoML for Predictive Maintenance: One Tool to RUL Them All. In: *IoT Streams for Data-Driven Predictive Maintenance and IoT, Edge, and Mobile for Embedded Machine Learning - Second International Workshop, IoT Streams 2020, and First International Workshop, ITEM 2020, Co-located with ECML/PKDD 2020, Ghent, Belgium, September 14-18, 2020, Revised Selected Papers*. Ed. by Gama, J.; Pashami, S.; Bifet, A.; Mouchaweh, M. S.; Fröning, H.; Pernkopf, F.; Schiele, G.; Blott, M. Vol. 1325. Communications in Computer and Information Science. Springer, 2020, pp. 106–118.

[TWW+20a] Tornede, A.; Wever, M.; Werner, S.; Mohr, F.; Hüllermeier, E.: Run2Survive: A Decision-theoretic Approach to Algorithm Selection based on Survival Analysis. en. In: *Proceedings of The 12th Asian Conference on Machine Learning*. ISSN: 2640-3498. PMLR, Sept. 2020, pp. 737–752.

[TWW+20b] Tornede, A.; Wever, M.; Werner, S.; Mohr, F.; Hüllermeier, E.: Run2Survive: A Decision-theoretic Approach to Algorithm Selection based on Survival Analysis. In: *Proceedings of The 12th Asian Conference on Machine Learning, ACML 2020, 18-20 November 2020, Bangkok, Thailand*. Vol. 129. Proceedings of Machine Learning Research. PMLR, 2020, pp. 737–752.

[Van18] Vanschoren, J.: Meta-learning: A survey. In: *arXiv preprint arXiv:1810.03548* (2018)

[WMH18] Wever, M. D.; Mohr, F.; Hüllermeier, E.: ML-Plan for unlimited-length machine learning pipelines. In: *ICML 2018 AutoML Workshop*. 2018.

[WMTH19] Wever, M. D.; Mohr, F.; Tornede, A.; Hüllermeier, E.: Automating multi-label classification extending ml-plan. In: *ICML 2019 AutoML Workshop*. 2019.

[WTMH21] Wever, M.; Tornede, A.; Mohr, F.; Hüllermeier, E.: AutoML for Multi-Label Classification: Overview and Empirical Evaluation. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43 (2021), no. 9, pp. 3037–3054

[XHHL07] Xu, L.; Hutter, F.; Hoos, H. H.; Leyton-Brown, K.: SATzilla-07: The design and analysis of an algorithm portfolio for SAT. In: *International Conference on Principles and Practice of Constraint Programming*. Springer. 2007, pp. 712–727