Data Mining and Knowledge Discovery (2021) 35:1258–1286
https://doi.org/10.1007/s10618-021-00749-5

Widening: using parallel resources to improve model quality

Michael R. Berthold[1,2] · **Alexander Fillbrunn**[1] · **Arno Siebes**[3]

Received: 7 June 2020 / Accepted: 5 March 2021 / Published online: 9 April 2021

Abstract
This paper provides a unified description of *Widening*, a framework for the use of parallel (or otherwise abundant) computational resources to improve model quality. We discuss different theoretical approaches to Widening with and without consideration of diversity. We then soften some of the underlying constraints so that Widening can be implemented in real world algorithms. We summarize earlier experimental results demonstrating the potential impact as well as promising implementation strategies before concluding with a survey of related work.

Keywords Widening · Machine learning · Data mining · Algorithms · Parallelization

1 Motivation and introduction

The trend to add more cores to modern processors and the growing popularity of cloud based compute resources has increased the importance of parallel algorithm development. The accompanying ability to generate more and more data has led to a wealth of research around machine learning and data mining algorithms that focuses on analyzing these growing data repositories using distributed methods. However, most datasets are still in the order of tens or at most hundreds of gigabytes (or as

Responsible editor: Johannes Fürnkranz

✉ Michael R. Berthold
Michael.Berthold@ieee.org

Alexander Fillbrunn
Alexander.Fillbrunn@Uni-Konstanz.DE

Arno Siebes
A.P.J.M.Siebes@uu.nl

1 University of Konstanz, Konstanz, Germany

2 KNIME AG, Zurich, Switzerland

3 University of Utrecht, Utrecht, The Netherlands

David Hand stated "only a small number of data analysis projects are actually big data projects"[1]).

In this paper we therefore take a different stance. We do **not** aim to scale to more (i.e."big") data or solve previously infeasible problems with parallel resources. Instead our goal is to improve model quality without increasing the overall time spent by investing parallel resources into better exploration of the (model) search space. These types of search problems are widespread in machine learning and data mining with models relying on numerical parameters that need optimization, discrete models turning this into a combinatorial search problem, and sometimes a hybrid of both.

Generally one can distinguish between two types of model search algorithms: those where the structure of the model space allows us to discard large regions where we know that it is unlikely or even impossible to find good models, and those which require an exhaustive search of the model space in order to find a good solution. There is a wealth of heuristics that have been introduced in past decades to make the latter complex model searches feasible in reasonable time on single core systems—these heuristics often resemble greedy algorithms, which are driven by making locally optimal choices. Our aim is to reduce the influence of these heuristics and produce the same type of model but at a substantially heightened level of quality. We achieve this by *widening* the search. Instead of pursuing only one solution at each step, we consider several solution candidates. In its easiest incarnation this results in a simple beam search. More complex widening approaches strive to broaden the search further, trying to always keep a diverse set of (intermediate) models. This mitigates the inclination of the greedy search to get stuck in local optima, as the diverse paths through the model space mean that while some of the parallel workers will hit a local optimum that they cannot escape, others may miss it and therefore have a chance of finding even better solutions.

However, even in the simple case of a beam search we are already faced with the problem of needing to synchronize continuously among all the parallel workers in order to redistribute the top choices that are contained in the beam. We therefore need other approaches to explicitly or implicitly pre-partitioning the model space to explore these different paths in parallel with no or only very limited synchronization requirements.

Widening differs from other approaches of investing parallel resources into improving model accuracy (see Sect. 5 for a more detailed comparison to related work). Our goal is to find a single (potentially even interpretable) model with better accuracy. In an ideal world, we would be able to find the optimal model—if we were just given enough parallel resources. In practice, of course, this still requires too many computational resources but we will get closer to the optimum.

The goals of this paper are threefold. First, we present a formalization for greedy model search algorithms. Secondly, we provide a formalization of *Widening* combining, expanding, and unifying earlier publications (Akbar et al. 2012; Ivanova and Berthold 2013) that describe a number of ideal methods for *Widening* of this type of search and reducing the impact of the greedy heuristic. These choices differ in how they widen the search with various partitioning methods. We then discuss alternatives

[1] Plenary at IDA 2013, London, UK.

for the practical realization of Widening for real world algorithms and also address the issues of communication overhead and diversity. Afterwards we summarize earlier encouraging experimental results (Akbar et al. 2012; Ivanova and Berthold 2013; Fillbrunn et al. 2017; Fillbrunn 2019), to illustrate the potential of Widening as a framework and also recap the *Bucket Selector*, a promising approach to inject diversity into Widening while at the same time reducing the amount of communication between parallel workers significantly. We conclude by discussing related work in more detail.

2 Greedy search machine learning

2.1 Preliminaries

Most greedy search based machine learning algorithms can be represented as a series of refinement and selection operators, which are applied to intermediate models until either a model is found that fits some criteria, until no other model can be created, or until some other criterion is fulfilled. These algorithms are the ones we are interested in, as their greedy nature lends itself to be improved by widening. The iterative scheme such algorithms follow, in which a given model is refined at each iteration, can be expressed as follows:

$$m' = s\,(r(m))$$

where m is the original model, for instance a neural network or a decision tree under construction, and m' is the next intermediate model, e.g. the neural network after one more gradient descent step or the decision tree after another node has been added. The two functions $s(\cdot)$ and $r(\cdot)$ describe a *selection* and a *refinement* operator, respectively. In the case of the neural network, the refinement operator represents the gradient descent step and there is no real selection operator. For the decision tree, the refinement operator would actually return a set of expanded decision trees around a particular (intermediate) leaf node and the selection operator picks the one with the highest information gain. Coverage based rule learning approaches match this setup as well: During the refinement step additional rule candidates are created and the selection operator picks the one that improves coverage the most.

The machine learning algorithms dealing with a large model space that cannot be reduced employ some sort of heuristic optimization. Gradient descent performs a local optimization during the refinement operator. Greedy searches usually embed heuristics in one or both operators, i.e. they generate only a subset of all possible refinements and/or the selection operator has no way of estimating absolute model quality, but instead has to rely on a local, greedy heuristic. Figure 1 illustrates this local exploration of the search space.

In order to more clearly investigate the possibilities of widening this type of search process, we need to better understand how the search space can be subdivided. In the following, we therefore first formalize the two operators before exploring alternatives to enable a more efficient exploration of the model search space.

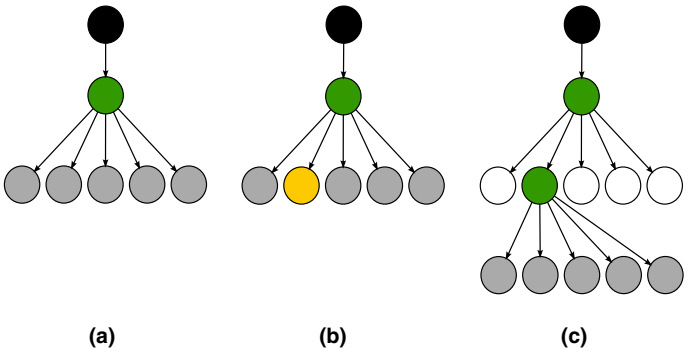

Fig. 1 The classic heuristic (often greedy) search algorithm. On the left (**a**), the current model m is depicted in green, the refinement options $r(m)$ are shown in gray. The selection operator s picks the yellow refinement (**b**) and the next level continues the search based on this choice (from Akbar et al. 2012)

2.2 Selection and refinement

The refinement operator $r(\cdot)$ essentially creates all new, intermediate model candidates that are to be evaluated. For a decision tree induction algorithm this would represent the generation of all possible new splits.

Definition 1 Let $\mathcal{M}$ be a family of models. A ***refinement operator*** $r : \mathcal{M} \mapsto 2^{\mathcal{M}}$ assigns to each model $m \in \mathcal{M}$ a set of models $\{m'_1, \ldots, m'_n\} \subseteq \mathcal{M}$.

Note that the definition of the refinement operator does not rely on the training data although in practice the training data is often used for the generation of refinements. Such a data dependency is more generally modeled within the selection operator introduced below.

However, before doing this we need to identify the starting elements of our search for a suitable model. These root element(s) are not refinements of any other model.

Definition 2 Let $\mathcal{M}$ be a family of models and r a refinement operator. $m \in \mathcal{M}$ is called a ***base model*** under refinement operator r iff $\forall m' \in \mathcal{M} : m \notin r(m')$.

The second part is the selection operator, which is used to pick the best refinement at each step based on the available data.

Definition 3 Let X be a domain of data instances and $\mathcal{M}$ a family of models. A ***selection operator*** $s : 2^X \times 2^{\mathcal{M}} \mapsto \mathcal{M}$ selects from a set of models $\mathcal{M}' \subseteq \mathcal{M}$ one model $m \in \mathcal{M}'$ based on a dataset $D \subseteq X$.

The selection operator s evaluates all intermediate model candidates and selects the locally best one(s). For the decision tree example this would correspond to evaluating all of the splits and selecting the one that optimizes the information gain measure. In the following, we will often omit the dataset D in the notation for the sake of readability.

Based on the refinement operator we can now define a partial order on our models, similar to the *more-specific/general-than* ordering discussed by Mitchell (1997) or the data generator inclusion relationship described by Siebes (2012).

Definition 4 Let $m_1, m_2 \in \mathcal{M}$ be two models from a model family $\mathcal{M}$. Given a refinement operator r, model m_2 is a ***direct refinement*** ($\succ_r^d$) of model m_1, if m_2 is created by applying the refinement operator on m_1: $m_1 \succ_r^d m_2 \Leftrightarrow m_2 \in r(m_1)$.

and, more generally:

Definition 5 Let $m_1, m_l \in \mathcal{M}$ be two models from a model family $\mathcal{M}$. Given a refinement operator r, model m_l is a ***refinement*** ($\succ_r$) of model m_1 if there exists a chain of direct refinements to reach m_l from m_1, that is: $m_1 \succ_r m_l \Leftrightarrow \exists \{m_2, \ldots, m_{l-1}\}, l \geq 2 : \forall i = 1, \ldots, l-1 : m_{i+1} \in r(m_i)$.

The simple Find-S algorithm (Mitchell 1997) finds all maximally specific models that are compatible with the training data. It does this by iterating over all positive examples and generalizing all current hypotheses to continue to be compatible with the data. Here no selection is performed, that is **all** suitable intermediate solution candidates are kept and further refined. This is, of course, not feasible for real world machine learning algorithms; most, in fact, consider only one solution. They then often rely on a local optimality criterion, which turns the algorithm into a greedy heuristic.

3 Widening

How can we reduce the impact of the greedy choice at each iteration? As mentioned before, greedily choosing the next model refinement can lead to getting stuck in local optima. To limit the impact such a local optimum has on the search, multiple parallel workers can search the model space in parallel, effectively *widening* the search. However, especially when the individual workers all start from a single base model, it is crucial that they do not converge onto the same (also: intermediate) solution.

We will start by describing the Widening framework itself followed by its most obvious incarnation, a straightforward extension of the greedy search, called Top-k Widening. Following this, we will discuss more structured approaches to guide the parallel searches through (approximate) search space partitions and by individually limiting the search paths of each worker. None of these approaches attempt to deliberately inject diversity into the search, which we address in the following sections. Again we will first describe an ideal setup before discussing a randomized version.

Figure 2 illustrates the key idea behind Widening. At each step, we select k models to be considered in the next step. Parameter k determines the width of our search, that is the amount of Widening that we are performing. In reality this will often be correlated to the available compute resources.

3.1 Top-*k* widening

A rather basic version of Widening is actually the beam search, which simply considers the (locally) best k models at each step. The Top-k selection operator works by comparing models across all branches at each step. This will be problematic if we want to parallelize this process, as it requires frequent communication across parallel workers: First each worker has to send its best k models to a central location, and from

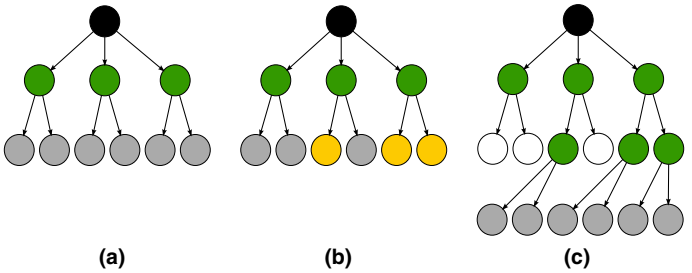

Fig. 2 Ideal Widening. From a set of models m (green circles), the refinement operator creates (in parallel) several sets of models (gray), shown on the left (**a**). The selection now picks a subset of the refined models (yellow circles in (**b**)) and the search continues from these on the right (**c**) (from Akbar et al. 2012)

those models the globally best k are selected. Next, these have to be distributed among the workers again. Not only does that mean that there are two exchange phases, but one central worker also has to collect all models and perform a selection operation on them while all other workers wait for the results, resulting in a *m:1:m*-communication scheme. Its disadvantages are the central processing and the fact that one worker has to receive a large number of models.

In addition, from a machine learning perspective, it may be more desirable to actually explore good yet diverse solution candidates in parallel in order to increase the chances of investigating substantially different solutions. We will demonstrate later that this is, in fact, also beneficial for real world cases.

The communication problem is due to our assumption that we explicitly make a selection from our pool of candidate solutions at each step. If we had a mechanism to a-priori define which candidates are to be explored by which worker, there would be no need for communication between the workers. If that is not possible, we may define beforehand which worker is responsible for which models and use that knowledge to reduce the required communication. We will discuss approaches for such distributed versions of Widening in the next section.

3.2 Ideal partitioning for widening

To avoid all workers being drawn into the same local optimum, the model space can be partitioned, ideally into non-overlapping partitions so that parallel workers can work independently. The underlying goal of Widening is therefore to split the search space (e.g. the space of all available models) across parallel workers in such a way that each worker has an independent set of models to consider and an equal chance of finding the optimum, avoiding communication with other workers as much as possible. However, we want to ensure that we can still guarantee finding a solution at least as good as the one found by the sequential, greedy algorithm. This requires a few properties of the resulting partitioning of our search space such as a closure property of the search space subsets:

Definition 6 Let $M \subseteq \mathcal{M}$ be a set of models. Given a refinement operator r the set of models M is ***closed*** under $\succ_r$ iff $\forall m \in M \ : \ r(m) \subseteq M$.

However, we can weaken this condition considerably, as in many cases we do not care too much about the intermediate models but only about the end result of each iteration, i.e. the models that end up being selected.

Definition 7 Let $M \subseteq \mathcal{M}$ be a set of models. Given a refinement operator r and a selection operator s, the set of models M is ***weakly closed*** under $\succ_r$ iff $\forall m \in M : s(r(m)) \in M$.

Note that this may end up creating duplicate work for parallel workers when we start to partition our model space, as the same model may be considered in several partitions as an intermediate solution candidate before selection. Based on this we can now define our ideal situation: a partitioning of the search space in such a way that any algorithmic implementation using the given operators will be simple to distribute because we know that each worker will never leave its subspace.

Definition 8 Given a family of models $\mathcal{M}$ and a refinement operator r, the partitioning $M_1, \ldots, M_k$ is called a ***closed partitioning*** iff $\forall k' = 1, \ldots, k : M_{k'}$ is closed under $\succ_r$.

However, here too, it is useful to define a weaker version which, as we will see later, allows for more communication-efficient implementations.

Definition 9 Given a family of models $\mathcal{M}$ and a refinement operator r, the partitioning $M_1, \ldots, M_k$ is called a ***weakly closed partitioning*** iff $\forall k' = 1, \ldots, k : M_{k'}$ is weakly closed under $\succ_r$.

But even this fairly soft constraint on the search space partitioning is often an unrealistic setup. In many cases, the refinement process initially starts from the same (often trivial) model, which would therefore need to be part of each subset, violating—initially—the above criterion. In the following we will therefore investigate how we can better approximate search space partitioning *implicitly* through specific modifications of the refinement and/or selection operator.

3.3 Approximate partitioning for widening

What we are interested in is a more general idea of how we can split the search space in such a way that—at least in principle—all potential solutions are still reachable via our refinement/selection operators, described above. For this we need a more relaxed definition of being closed—we do not require all elements and their neighbors to be part of the same set; we require only that each element is reachable along at least one path:

Definition 10 Given a family of models $\mathcal{M}$ and a refinement operator r, a subset $M \subseteq \mathcal{M}$ is called a ***path-closed set*** iff $\forall m' \in M \wedge m'$ is not a base model $: \exists m \in M : m' \in r(m)$.

That is, each model in our subset has either at least one element in the same subset of which it is a refinement, or no such model exists at all (e.g. the model is a base model of this subset). Using this we can define the split of our search space as a set of subsets which, together, cover the entire search space and each model is reachable in at least one subset:

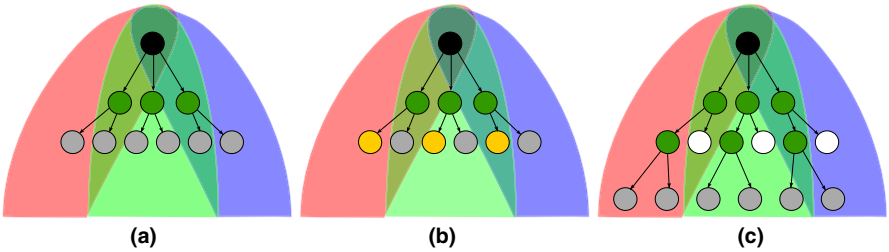

Fig. 3 Approximate Partition-Based Widening. The search space is divided into a set of overlapping subsets, however, we guarantee that each model is reachable within at least one subset. Instead of selecting from a set of intermediate candidates, each (parallel) worker now only selects the one best candidate in its own subset of the search space

Definition 11 Given a family of models $\mathcal{M}$, a division into k subsets $M_1, \ldots, M_k$ is called a ***path-closed approximate partitioning*** iff $M_1 \cup \cdots \cup M_k = \mathcal{M}$ and $\forall i : M_i$ is a path-closed set.

The ideal scenario would be a division into disjoint, closed sets, that is a closed partitioning. In practice we will likely have to settle for a less ideal setup where the subsets partially overlap but each model is still reachable in at least one set. The amount of overlap obviously directly affects potential redundancy when we use this kind of approximate partitioning to widen our search. Figure 3 illustrates this approach.

3.4 Path-based widening

Instead of relying on explicitly defined (approximate) partitions, we can also inject a segmentation of the search space implicitly through the use of individual selection operators. They ensure that a particular parallel worker takes only those models into consideration for which it is responsible. In its simplest incarnation this will simply consider the partition assignment and we are back at Partition-Based Widening. However, the more interesting case is an incarnation where each parallel worker prefers different models, in effect using an individual preference schema to pick the model m' to continue working with.

Definition 12 Given a family of models $\mathcal{M}$, a refinement operator r, and a selection operator s, a linearly ordered set of models $P_s = \{m_1, \ldots, m_n\}$ is called a ***selection path*** iff $\forall 1 \leq i \leq n-1 : s(r(m_i)) = m_{i+1}$ and m_1 is a base model.

The idea of path-based Widening is now that either the refinement operator or the selection operator or both are individualized in such a way that they prefer a specific subset of models. Then we can focus on particular selection operators, resulting in different selection path partitions, depending on the choices they make:

Definition 13 Given a family of models $\mathcal{M}$, a ***path-based subset*** based on selection operator s is the subset $M_s \subseteq \mathcal{M}$ of models that are selection-path-reachable via s, that is, $M_s = \{m : \exists P_s = \{m_1, \ldots, m\} \wedge m_1 \text{ is a base model}\}$.

In an ideal (perfect) scenario, the overlap between partitions induced by the selection operators is empty but the union of those partitions is the full model space:

Definition 14 Given a family of models $\mathcal{M}$, a set of selection operators $s_1, \ldots, s_k$ is called a ***perfect selection operator set*** iff $M_{s_1} \cup \cdots \cup M_{s_k} = \mathcal{M}$ and $\forall i, j = 1, \ldots, k : i \neq j \Rightarrow M_{s_i} \cap M_{s_j} = \emptyset$.

Finding a set of selection operators that are overlap free will be hard to achieve in reality. However, the framework introduced above allows us to put the implications of the actual implementations we use into context.

3.5 Diversity-driven widening

Until now we have discussed approaches to ensure our Widening methods explore different portions of the search space but we have completely ignored our goal to explore the search space as broadly as possible.

Using partitions (either directly or path-indirectly) does not guarantee that we find the top k solutions. We will see later that this does not appear to have a big effect. Moreover, beam search (aka, Top-k Widening) does not guarantee the discovery of the globally k best solutions either, since we cannot rule out that the path to any one of those solutions does not go through an intermediate solution that is not part of the top k at that step.

Of more importance is a bigger drawback of all beam search style searches, namely their focus on a narrow portion of the search space. Similar to issues known in genetic algorithms (and often addressed explicitly in this context, for an example see Goldberg and Richardson 1987), these search methods tend to exploit areas around local optima rather than globally exploring the search space and potentially also finding the global optimum. Our partitioning-based approaches described above do not encourage a global exploration either, because the partitioning incorporates no further constraints, such as enforcing *diversity* of the models in separate partitions. The result could be that different partitions actually contain fairly similar models resulting in all of our workers finding similar solutions.

Hence partitioning would benefit from an additional diversity constraint, making sure we are not only following the path (or beam) of (locally) best models but truly exploring the overall model space to get closer to discovering the globally best model. Note that this turns the model selection step into a multi-objective problem as we will now be required to balance the quality of the solutions vs. the additional desired diversity of the set of selected solutions.

3.6 Ideal diversity-driven widening

As before, let us first discuss an ideal setup for the solution of this problem, which would lead to a diverse set of final models adhering to a given diversity constraint. For model quality we assume again a given performance metric. In order to measure the diversity of a set of models we have many options: the obvious choices are optimizing the average pair-wise similarity, the sum over all pair-wise similarities, the minimum or the maximum. We will not dive into more detail here but instead refer to Meinl et al. (2011) where an extensive overview is given. For the purpose of this paper, it is enough to assume that we are given a function Ψ measuring the model quality and a function

Δ, which measures the diversity of a set of models. Given those two functions, we can now define what the goal of diverse Top-k should be:

Definition 15 Let $M \subset \mathcal{M}$ be a subset of models, Ψ and Δ a model quality resp. diversity function (for k models), and k the width-parameter, then the function $s_{\text{topdiv}-k}$ is called ***best-diverse-k selection operator*** if $\forall M' \subseteq M \wedge |M'| = k : \Psi(s_{\text{topdiv}-k}(M)) \geq \Psi(M') \vee \Delta(s_{\text{topdiv}-k}(M)) \geq \Delta(M')$,

That is, there is no other subset of size k that performs better, and, at the same time, is more diverse. As with all multi-objective optimization problems, there is potentially more than one such solution: the entire Pareto front contains non-dominated alternatives. Picking the subset of high-quality and diverse models from a given set of models that fulfill a diversity criterion is an art in itself and many algorithms exist as the optimal solution is (often: NP) hard to find. But even for our purpose here—widening of the greedy approach—we face a challenge as for a simple approach such as a Top-k Widening we need to pick diverse subsets at every step, requiring extensive communication between all workers to find the best, yet diverse subset.

3.7 Randomized diversity-driven widening

In many cases, actually determining model diversity is infeasible. We can approximate this by a disjoint set of final models using an entirely random, though closed partitioning. We base this on the assumption of a *randomized partitioning* as follows:

Definition 16 Given a family of models $\mathcal{M}$ and a random number assignment $h : \mathcal{M} \to \mathbb{N}$ the partition of $\mathcal{M}$ into k subsets $M_i = \{m \mid h(m)\%k = i\}$, $i = 1, \ldots, k$ is called a ***randomized partitioning***.

(Note the strong similarity to hash functions and that by definition, it automatically follows that the sets are disjoint and their union equals the entire model family.)

It would then be easy to force each worker to consider only models that are part of its partition, however, devising a random partitioning that also guarantees each partition's being a path-closed set will be tough and will require taking into account some structural properties of the models—which violates the assumption of our partitioning being random. The two goals of closeness under refinement and the desire to have model-independent, randomized partitioning are obviously conflicting. Note that in order to ensure finding the same solution as Top-k Widening **each** worker needs to investigate the top k solutions of its subset as well.

Note that we can use randomized partitioning to create estimates regarding the optimal performance. Assuming that we do indeed sample the underlying model space uniformly, we can use the quality measures of the models in our sample to derive an estimate for the overall quality distribution. This requires some simplifying assumptions on the underlying model space, and it will be interesting to find out how much this affects those estimates for real world model families. However, Randomized Widening itself also makes fairly strong assumptions on the underlying randomization. We assume that we partition our model space in such a way that, given the training data, the chance of each worker having the best model as part of its partition is equally good. In order to actively control this, we will have to inject diversity constraints directly.

3.8 Summary

In this section we have introduced the Widening framework and a number of definitions that allow us to formalize actual implementations. In particular we make the distinction between explicit partitions of the model space and how partitions can be closed under refinement or just weakly closed when the selection operator has been applied as well. We relaxed these definitions to allow for joined starting points, focusing on the reachability of all models within at least one partition. Afterwards we introduced the notion of path-based Widening which relies on the selection operator to implicitly segment the model space. We concluded by formalizing the injection of an explicit diversity constraint and its easiest implementation: randomized partitioning. The latter does not guarantee closeness but we will see in the next section how a fairly low communication overhead can be invested to make this issue disappear in practice.

4 Practical considerations and experimental insights

In the previous section we have described the Widening framework in detail but have skipped any considerations on how this can actually be implemented. The original claim of "better, not faster" (Akbar et al. 2012) implies "not slower" so we have to not only consider how to ensure the diversity of the search but also pay attention to the runtime. It is obvious that straightforward approaches such as Top-k Widening require additional communication between the parallel workers and if each worker has a runtime similar to the classic greedy search, this will result in a significant slow down of the overall search.

In the following we summarize a series of results from the past couple of years that provide some insight into practical ways to inject diversity into the model search and how to reduce the communication overhead. We will first compare those approaches qualitatively and subsequently discuss quantitative results. We end the section with a summary of lessons learned and suggestions on what to consider when implementing Widened versions of other machine learning algorithms.

4.1 Top-*k* widening

As mentioned above, the most obvious approach to Widening implements a beam search and simply follows the path of the (at each step) k best models. Called *Top-k Widening*, this approach implicitly induces a *weakly closed partitioning* since each worker may produce duplicate models during refinement but in the selection phase, duplicates are eliminated. Top-k doesn't enforce diversity, so the risk of following a local optimum is high. In addition to the effort to pick the k best solutions, communication overhead between workers is $\mathcal{O}(k)$.

Nevertheless, Akbar et al. (2012) already showed with some simple experiments on a set covering benchmark that two expected effects do materialize: Investing lots of additional work into exploring more than just one search path leads to moderate improvements in quality and a reduction in variation (e.g. it becomes more likely

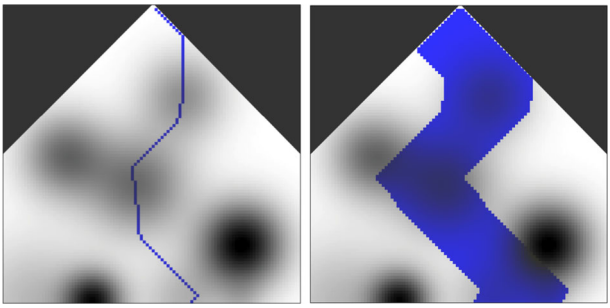

Fig. 4 Illustration of the search paths of a greedy search and Top-*k* Widening aka Beam Search. Darker shades indicate areas of better model quality

that we are choosing a better model). Using set covering seems a bit of a stretch but a number of rule induction algorithms make use of variations of the "sequential covering" approach used here.

Figure 4 illustrates the difference between the greedy search and Top-*k* Widening. The initial model is on top and continuous refinement/selection steps progress downwards. The blue line/bar shows the search individual resp. beam path. Areas with higher model quality have darker shading. The illustration nicely shows how Top-*k* Widening broadens the greedy search path.

4.2 Injecting diversity

Generally, diversity-driven Widening can be implemented using methods that explicitly measure diversity between models and use that to enforce sufficiently different paths through the search space, and methods that produce diverse solutions more indirectly, e.g. through injecting preferences for specific model/model parts or aspects of the data or through sheer randomness. One simple approach assumes that similar models have similar scores and therefore selecting some good and some not so good models is sufficient to cover the search space more broadly, an approach also often taken in reinforcement learning and evolutionary algorithms.

While the ideal case of a perfectly partitioned search space, where the probability of finding the best model is the same for each partition, requires no communication between parallel workers, in reality we can hardly search efficiently without communication. This is especially true when diversity between the explored models should be taken into account explicitly.

4.2.1 Explicit diversity: diverse top-*k*

In order to enforce a certain degree of diversity among the selected models, the Top-*k* algorithm discussed earlier can be modified to take model similarity into account. A simple implementation constrains the Selection operator the following way: First the best model is selected, then the next best model is chosen that has a similarity less than some threshold θ compared to the best model. All subsequently selected models must

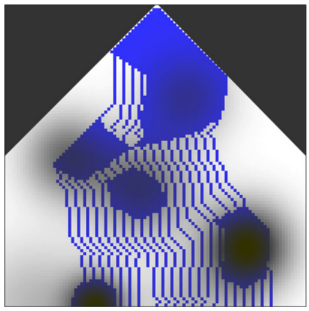

Fig. 5 Illustration of the search paths of Diverse Top-*k* Widening. The influence of the diversity threshold is clearly visible

also have a similarity not greater than θ compared to all previously selected models. Such a selection scheme is called *Diverse Top-k* selection (Felner et al. 2003). More sophisticated diversity selection schemes exist, such as for instance the Score Erosion method described by Meinl et al. (2011). In order to decrease the number of models that need to be transferred between workers, Diverse Top-*k* can be implemented with a local pre-selection, where each worker performs Diverse Top-*k* selection on its refinements with local θ_l, then models are collected centrally and another round of Diverse Top-*k* is performed with a global θ_g to select k models. While Diverse Top-*k* guarantees diversity, it requires a sorting step (or at least k-best picking) before the selection, which makes this method rather slow. Furthermore, Diverse Top-*k* does not improve upon the communication behavior of Top-*k* and has high memory consumption. This is because the sorting requires all refinements to be available at the same time, as opposed to Top-*k* where each model only needs to be seen once. Similar to Top-*k*, *Diverse Top-k* also induces a *weakly closed partitioning* since in the diversity selection phase, duplicates are eliminated. Figure 5 illustrates this type of search. Depending on the diversity threshold, the search is more spread out than Top-*k* Widening.

4.2.2 Explicit diversity: data- versus model-driven diverse top-*k*

Ivanova and Berthold (2013) investigated a data-driven approach to Explicit Diversity Driven Widening on the same datasets as used for the set covering experiments by injecting worker-specific, pre-determined preferences for the data to be covered. Depending on the strength of these preferences this results only in different ways to perform tie breaking (which occurs frequently when following the greedy set covering algorithm) or it can result in stronger preferences to cover specific patterns earlier during the search.

An alternative to injecting diversity via data coverage is *Model-Driven Diversity* where we use properties of the models to determine which ones to refine and/or select. Ivanova and Berthold (2013) also describe such an approach where diversity is enforced through a similarity function on the models by using the Jacard distance of the resulting set covers (the elements already covered by the intermediate models).

Fillbrunn and Berthold (2015) apply this type of approach to a hierarchical clustering and a number of benchmarks. In order to inject diversity, a tree-based distance

method between hierarchical clusterings was used, allowing for the selection strategy to pick a diverse subset of models at each step.

A very different approach for Widened construction of Bayesian Networks is explored by Sampson et al. (2018). Instead of weighting specific model parts, each network is assigned a code based on Fiedler vectors. These codes are then hashed and each parallel worker only considers networks with its specific code. The big challenge here is to find a suitable coding scheme so that most models are reachable within the same code-induced partition that a specific worker is focusing on.

The latter two approaches are *path-based* approaches to Widening but are both not closed, i.e. neither of them guarantee that every possible model is, in fact, reachable. But they require zero communication overhead between the workers and—depending on the way the path is defined—therefore require only moderate extra work to enforce model-driven diversity.

4.2.3 Implicit diversity

Explicit diversity results in either rather high complexity in terms of communication overhead or, for the model-driven alternative, has no communication overhead but requires extremely careful adjustment about the types of models pursued to ensure useful diversity and completeness of the search itself. One way of reducing this complexity is based on the assumption that similar models also have a similar score, a core assumption in induction learning. Then, standard diversity selection approaches can be used based on a measure for model similarity.

4.2.4 Implicit diversity: hashed bucket selector

Approaches like roulette wheel selection (Bäck 1996) can also be applied in Widening, but have two disadvantages: They require multiple communication phases and they are not *elitist*, meaning that the best model is not always selected. A better selection method is the *Bucket Selector* first introduced by Fillbrunn et al. (2017), where each parallel worker is responsible for models with a certain hash code. This hash code determines the partition a model is in. To refine and select a set of models, each parallel worker first refines its locally held model(s) and assigns them one of k partitions using a hash function that is shared among all workers. Afterwards the locally best model in each partition is sent to the responsible worker, which then selects the overall best model from its partition. Given a refinement function r, a hash function h, and a scoring function ψ, for the set of models $M_{i+1} \subseteq r(M)$ selected for the next iteration the following formula holds:

$$M_{i+1} = \{m_i \mid i \in 1, \ldots, k, \\ \forall m_j \in r(M) : h(m_i) = h(m_j) \implies \psi(m_i) \geq \psi(m_j)\}.$$

Since models can be exchanged between workers directly in a single communication phase, this represents a *m:n*-communication scheme. Compared to *m:1:m*-communication it has obvious advantages: There is no central worker doing

processing while others wait, the maximum number of models transferred between two workers is $k - 1$ and fewer ($k[k - 1]$ versus $[k + 1][k - 1]$) models have to be transmitted in total.

As opposed to selection schemes like roulette wheel selection, the selection probability of a model processed by the same worker is not proportional to its score, but rather to its rank among the other models. The ith-best model has a selection probability of

$$P(m_i) = \left(1 - \frac{1}{k}\right)^{i-1},$$

as there are $i - 1$ better models and each of them has a chance of $1 - \frac{1}{k}$ to fall into the same partition, assuming that the values of the hash function are distributed evenly between 1 and k.

The bucket selection has a runtime complexity of only $\mathcal{O}(n)$ as opposed to Top-k's $\mathcal{O}(n \log k)$ or Diverse Top-k's $\mathcal{O}(n \log n + nk)$. For each model the hash function needs to be evaluated and the model score compared to the currently best model in its partition. If the score is better, the top model is replaced, otherwise nothing happens. This also means that similar to Top-k, each model only has to be seen once, making the process much more memory efficient than Diverse Top-k and roulette wheel selection, where all models need to be held in memory at the same time to allow sorting or calculation of statistics such as the sum of the scores.

The *Hashed Bucket Selector* induces a *weakly closed partitioning* as each worker only works on models in its partition but may produce refinements that are not part of its partition. If the refinement operator only produces models with the same hash value as the parent model, than no communication overhead occurs, each worker performs a greedy search in its own partition and the induced partitioning is closed. In the more general case, communication between workers is in $\mathcal{O}(n^2)$ in addition to the potentially costly evaluation of the hash function. However, compared to Top-k the communication occurs directly between workers (decentralized) and no single worker has to do work while the others wait.

One caveat is that the hash function of the bucket selector has a strong influence on the diversity of selected models. One extreme is a locality sensitive hash function that is more likely to assign similar models the same hash value, so that models "close" to each other in model space often appear in the same partition. Using an arbitrary hash function that only ensures that its values are spread evenly and that identical models get the same hash value would be ideal but is hard to realize in reality. This would be close to the ideal scenario of diversity-driven Widening mentioned in the previous section, albeit with moderate communication overhead.

Figure 6 illustrates this type of search as well. No diversity threshold is needed as is the case with Diverse Top-k—visible here as the less equally distributed spread.

4.2.5 Implicit diversity: random bucket selector

One issue with the Hashed Bucket Selector is the need to compute a hash code in order to assign models to buckets. Designing good hash codes is non trivial and computing

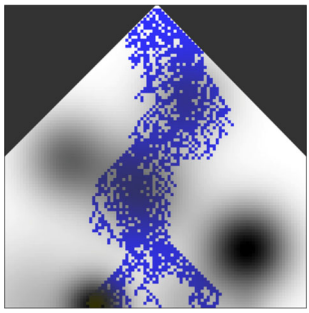

Fig. 6 Illustration of the search paths induced by the Hashed Bucket Selector. The influence of the hash code driven distribution becomes visible as well

them can be a costly operation. One variant of this method—avoiding the calculation of a hash code—is to simply assign a random code to each model when it is generated. This is obviously a fast operation and it guarantees all desired randomness properties, but the big drawback is that we can no longer guarantee that the same model generated through different paths (by the same or different workers, even!) has the same code. This works if duplicates are of little or no concern, e.g. because the model space is a tree. While this does not change the general runtime complexity, it avoids the frequent evaluations of the hash function at the cost of potentially exploring the same model in several buckets. Therefore the result is not even a *weakly closed partitioning* since partition assignments are generated randomly at every step, allowing the same model to appear in different partitions throughout the search. Communication between workers is in $\mathcal{O}(n^2)$, similar to the Hash Bucket Selector but without the need for computation of a hash function. Diversity is close to ideal as exploration of the space is truly random. The only caveat is that model duplications artificially inflate the search space as different workers can work on the same models.

4.3 Experimental insights

More results are reported in earlier publications—in this section we focus on an interesting subset comparing the approaches described in this section and providing validation for some of our hypotheses and interesting insights into their behavior. We conducted experiments for widened agglomerative hierarchical clustering on three datasets: Breast Cancer (569 instances, 32 attributes (Wolberg and Mangasarian 1990)), Seeds (210 instances, 7 attributes (Charytanowicz et al. 2010)), and Ruspini (75 instances, 2 attributes (Ruspini 1970)) from the UCI Machine Learning Repository. Due to their smaller size only the latter two are suitable for Diverse Top-k while the Breast Cancer dataset is too large. The number of refinements in hierarchical agglomerative clustering scales with the number of data points and the Diverse Top-k selector's sorting step constitutes a massive performance bottleneck. We therefore report results for the Seeds and Ruspini datasets for Greedy, Top-k, Diverse Top-k, and the Bucket Selector; we do not run Diverse Top-k for the Breast Cancer dataset.

Each experiment is run 100 times, each time drawing 90% of the original dataset for training. The results of these 100 runs are visualized as box plots of the cophenetic correlation coefficient (Sokal and Rohlf 1962) obtained by clustering the random sub-

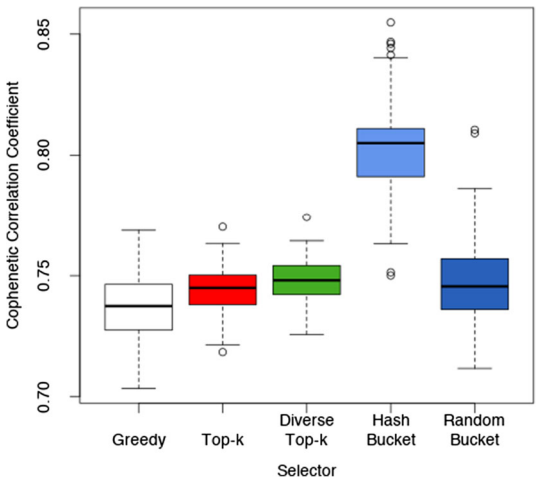

Fig. 7 Results of widening hierarchical agglomerative clustering on the Seeds dataset with $k = 100$

set with each of the selectors mentioned above. The cophenetic correlation coefficient measures how faithfully the dendrogram structure represents the distance between points $d(\mathbf{x}_i, \mathbf{x}_j)$ in the underlying dataset and is determined by the correlation of the elements of the original distance matrix used to infer the dendrogram and the distance induced by the dendrogram, where the distance $t(\mathbf{x}_i, \mathbf{x}_j)$ between two data points is the height of the node at which the two are merged into the same cluster:

$$c = \frac{\sum_{i<j} \left(d(\mathbf{x}_i, \mathbf{x}_j) - \overline{d}\right) \left(t(\mathbf{x}_i, \mathbf{x}_j) - \overline{t}\right)}{\sqrt{\left(\sum_{i<j} \left(d(\mathbf{x}_i, \mathbf{x}_j) - \overline{d}\right)^2\right) \left(\sum_{i<j} \left(t(\mathbf{x}_i, \mathbf{x}_j) - \overline{t}\right)^2\right)}}$$

with $\overline{d}$ resp. $\overline{t}$ being the average distance in the data space resp. dendrogram.

Figure 7 shows the results of clustering the Seeds dataset with 100 dendrograms created in parallel. Here Top-k achieves slightly better results than the greedy algorithm and Diverse Top-k is again slightly better. The Random Bucket Selector is just as good as Top-k. Obviously in this case determining a model's partition using hashing (and thus avoiding exploration of redundant solutions) significantly improves the median cophenetic correlation coefficient from around 0.75 to more than 0.8. On the other hand, the fact that the Hashed Bucket Selector outperforms Diverse Top-k is due to the explicit diversity injection which Diverse Top-k requires. This emphasizes that finding a good similarity metric is critical for this method whereas the Hashed Bucket Selector only relies on a good hashing mechanism.

Figure 8 displays the results for the Ruspini dataset. With 75 observations and two dimensions, this dataset is rather simple, seemingly offering less room for improvement over the greedy algorithm. Again the Hashed Bucket Selector achieves the best results, albeit by only a small margin. Top-k and Diverse Top-k are on the same level, while greedy reaches slightly lower cophenetic correlation coefficients. Similar to the Seeds dataset, the Random Bucket Selector is slightly better than Top-k.

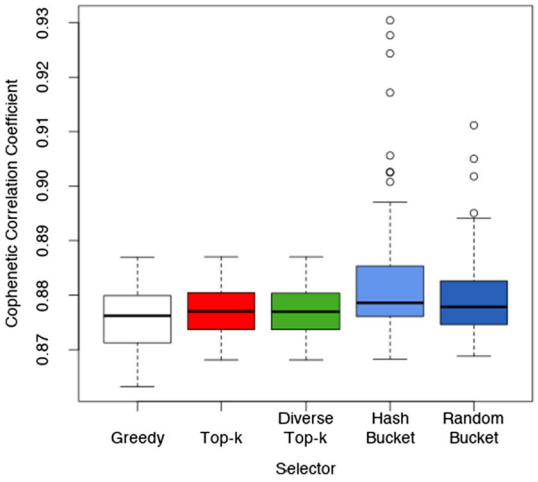

Fig. 8 Results of widening hierarchical agglomerative clustering on the Ruspini dataset with $k = 100$

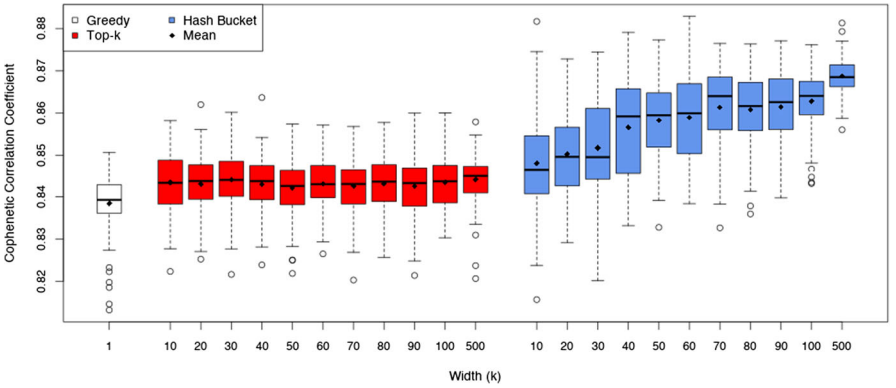

Fig. 9 Results of widening hierarchical agglomerative clustering on the Breast Cancer dataset

Figure 9 shows more in-depth results for widened hierarchical agglomerative clustering on the Breast Cancer dataset with the Greedy, Top-k, and Bucket Selector, the latter two of which were tested with varying k. As can be seen here, Top-k achieves slightly better results than the greedy algorithm, but cannot improve much further with increasing k. The mean cophenetic correlation coefficient remains at around 0.843, even at $k = 500$, while the greedy selector achieves 0.84. The Hashed Bucket Selector steadily increases its model quality with higher k, achieving as much as 0.869.

Note that the Top-k selector used in this experiment explicitly removes duplicates, suggesting that it indeed selects close but very similar models. The implicit diversity of the Hashed Bucket Selector, on the other hand, makes it spread more broadly across the model space, finding better models in other areas.

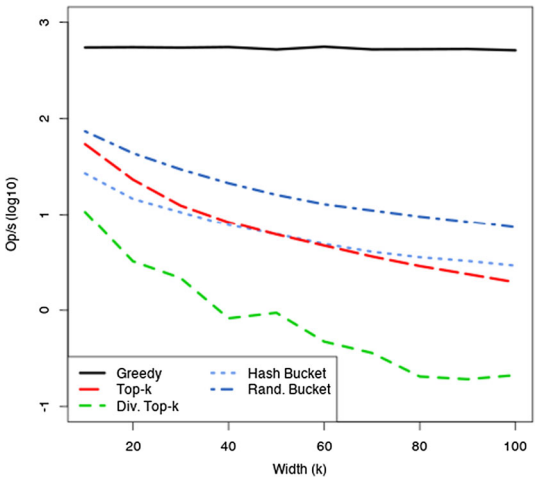

Fig. 10 Average operations per second of a single local run for different selectors

4.4 Runtime observations

In order to validate our runtime assessments above, we evaluated the runtime of the previously mentioned selectors in two ways: measuring the time for executing the local part of the selector alone and measuring the total time the algorithms need for clustering an artificial dataset. All benchmarks were run using the Java Microbenchmark Harness (JMH)[2] on a machine with an AMD EPYC 7351P 16-Core CPU and 128 GB of DDR4 main memory.

Figure 10 shows the operations per second of the individual selectors in isolation, selecting k models out of 435,000 for different values of k. As expected the $\log k$ component of the Top-k's runtime complexity of $\mathcal{O}(n \log k)$ (n being the number of refinements from which to select) makes its runtime increase faster with k than then Bucket Selector's. Initially the Hashed Bucket Selector has a higher runtime and therefore lower throughput, which is a result of the rather time consuming calculation of the hash function. At around $k = 1200$ Top-k takes longer for selection than the Hashed Bucket Selector. The Random Bucket Selector does not have to calculate any hash functions and therefore offers a better throughput from the start. The throughput of the Diverse Top-k selector is lower than any of the others. Not only does this selector need to sort the 435,000 models by score, it also has a selection process with a complexity of $\mathcal{O}(n \log n + nk)$, as every selected model needs to be compared to the previously selected ones. None of the selectors achieves a throughput as high as the greedy selector, even though the theoretical complexity of the Hashed and Random Bucket Selector is $\mathcal{O}(n)$ as well. In practice, the selection of multiple models is still more complex and time consuming.

The operations per second executing a complete clustering run with a dataset consisting of 50 10-dimensional random data points on 8 parallel threads is shown in Fig. 11. Again the greedy selector is constant and high, as it does not depend on k. All other selectors are affected by the size of k due to the fact that a higher value of

[2] https://openjdk.java.net/projects/code-tools/jmh/.

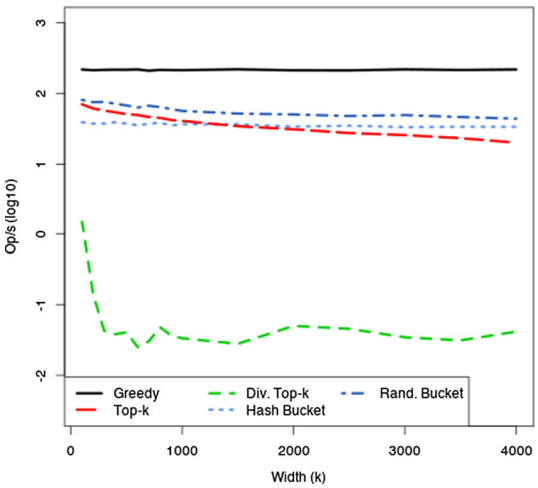

Fig. 11 Average operations per second of a full clustering run for different selectors

k also means more refinements from which to select. This is the reason why the two bucket selectors' throughput decreases linearly. Top-*k* and Diverse Top-*k* also have a *k*-term in their runtime complexity and therefore take even longer to execute when *k* increases. Diverse Top-*k* is again significantly slower than all other selectors.

4.5 Lessons learned

So what do we take away from the above? As expected, simple *Top-k Widening*, which essentially conducts a beam search in the model space, can improve model quality and at the same time reduce the variance. However, the computational overhead is substantial and deviates far from the goal of "...in the same time".

We have reviewed different approaches to divide the search space by either using a method to divide the model space into partitions (*partition-based Widening*) or injecting a preference for different search paths (*path-based Widening*). In theory these approaches can completely eliminate any communication overhead but in practice it is non trivial to find suitable mechanics by either adjusting refinement- and/or selection operators or assigning models to specific partitions. In both cases guaranteeing closeness of the search is not easy.

Adding diversity to Widening can help to ensure better coverage of the entire search space and avoid getting stuck in local optima. We have seen with *Diverse Top-k* that this simple approach can indeed improve model quality but the computational cost is high. Adding diversity implicitly by e.g. hash or hash-like codes can also work and eliminate communication overhead completely at the cost of losing closeness of the search itself.

A promising compromise is presented by the *Hashed Bucket Selector*, which allows injection of diversity through a hash code and guarantee closeness of the search but with much more reasonable computational overhead than (*Diverse*) *Top-k*. If the risk of duplicate model exploration is manageable, a simplified version of this approach, the

Random Bucket Selector allows the avoidance of potentially complex computations of hash codes at the risk of exploring the same models in multiple partitions.

A few more insights resulted from unpublished (since negative or unimpressive) results when we attempted to widen decision tree induction. There is an obvious way to partition the search space—at least to a certain degree: use different attributes for the first split. Then each worker focuses only on refining trees with a different root node. This approach has two problems: some of the workers will be forced to focus on a clearly suboptimal part of the search space, especially when the data contains many useless attributes. A less obvious issue also surprised us initially when we investigated more sophisticated ways for injecting diversity into decision tree widening: Decision trees are rather order independent. Even if a useless attribute is used early on, the algorithms will simply introduce the more important attribute later on in many of the branches in parallel. The resulting tree will become a bit bigger but will still have essentially the same performance. One can eliminate this effect a bit by enforcing a maximum tree size on all workers, but that just adds another parameter to the mix. However these results hint towards an explanation for the occasional superiority of data-driven diversity over model-driven diversity. Two very different models may still perform pretty much the same tasks on the underlying data. So comparing models (which in many cases means designing hash codes) needs to be done carefully to ensure that one compares **what** the model does rather than **how**.

5 Related work

Akl (2002) already introduced the idea of increasing model quality by adding quantity, however it focuses on a broad range of applications ranging from cryptography to game playing. Focusing on machine learning algorithms allows us to formalize a number of constraints based on the underlying model search space.

As mentioned above, considerable research has been conducted aiming to parallelize machine learning algorithms. We will summarize those (very briefly) in the following subsection. In addition, methods that improve the accuracy found by standard learning algorithms are also of relevance to the Widening approach described here. We—again only briefly—summarize the main aspects in the following subsection. Lastly many approaches for improving standard greedy search algorithms exist. We will also take a brief look at the most relevant aspects of this area of research. We are purposefully skipping much of the work that has surfaced under the "Big Data" umbrella as this is not the focus of the methods discussed here.

5.1 Machine learning algorithm parallelization

A wealth of related work exists around the parallelization of machine learning algorithms. Most of it aims at speeding up sequential algorithms to enable them to be applied to larger datasets. Sometimes fairly generic libraries are provided but others rely on rather specific parallel programming frameworks.

5.1.1 Speed-up through parallelization

For the vast majority of parallelizations of machine learning algorithms, the aim is to improve efficiency. We will cover only closely related work, as extensive reports and surveys exist already (Kumar et al. 2001; Kargupta and Chan 2000; Zaki and Ho 2000; Zaki and Pan 2002).

Decision tree parallelization has been broadly explored. One of the earliest distributed decision tree algorithms, SPRINT (Shafer et al. 1996), has served as the basis for many subsequent parallel decision tree approaches. The most prominent example is probably described by Zaki et al. (1998), where SPRINT was parallelized for SMP machines in a data parallel fashion. Other decision tree parallelizations are described by Darlington et al. (1997) using task parallelism and Srivastava et al. (1999) and Kufrin (1997) presenting hybrid approaches. The focus of all of these parallelization approaches is to accelerate the learning process and not the generation of better models.

Another well-researched area is parallel association rule mining algorithms. Extensive surveys exist in this area (e.g. Zaki 1999). Many of these methods are parallel variants of well-established sequential algorithms with the classic Apriori method (Agrawal 1994) serving as the basis (Agrawal and Shafer 1996; Shintani and Kitsuregawa 1996; Parthasarathy et al. 2001; Han et al. 2000). Approaches based on the sequential version of the Eclat algorithm were adopted in MaxEclat, Clique, and MaxClique (Zaki et al. 1997). These algorithms utilize the structural properties of frequent item-sets to speed up the search but do not attempt to improve results.

Parallelism in clustering algorithms has been used for both efficient cluster discovery and more efficient distance computations. Typically, the approaches utilize task and SPMD parallelism and master-slave architectures. Partitioning clustering algorithms are parallelized mostly using message-passing models, examples are presented by Judd et al. (1998); Dhillon and Modha (2000); Kantabutra and Couch (2000). Hierarchical clustering tends to be more costly. Two noteworthy examples are from Olson (1995), where a hypercube network of processors is used to reduce the computation of the minimum spanning tree and PBIRCH (Garg et al. 2006), where each slave computes its own structure based on part of the data, whereupon the slaves then exchange results. Again, the goal of all of these approaches is to save time, not improve model quality.

5.1.2 Specific frameworks

Dean and Ghemawat (2008) introduced MapReduce as a paradigm for processing parallel tasks across very large datasets, allowing for massive scalability across a compute cluster. It is based on a decomposition of the algorithm into a map and a reduce step. Due to that, MapReduce requires specialized versions of the machine learning algorithms to be developed. For example, Zhao et al. (2009) present parallel k-means clustering and Verma et al. (2009) propose the scaling of genetic algorithms using MapReduce. Many other papers focus on single MapReduce implementations of other machine learning algorithms. Nowadays MapReduce has lost attention due to the requirement to split algorithms into two disjunct phases (Ma and Gu 2010 discuss other

issues in more detail) but it is interesting to note that it fits the Widening framework described here extremely well: The refinement operator represents the map-step and diversity-driven selection can be done during the reduce step.

More recently, Spark has taken over when it comes to implementations of machine learning algorithms on large clusters (Meng et al. 2016) but here the focus is also primarily on working with larger data sets, not model quality improvement. Chu et al. (2006) present another approach to parallelize algorithms by using a summation representation of the algorithms; it is applied to locally weighted linear regression, k-means, Naive Bayes, support vector machines, expectation maximization, and others.

5.2 Model quality improvement

A number of papers also concentrate on improving the accuracy of the models. Some attempt to improve the greedy algorithm by making less greedy choices, others learn more models to be used in concert (ensembles) or in a randomized fashion (meta heuristics).

5.2.1 Look ahead strategies

Similar to the aspect of *Deepening* as discussed by Akbar et al. (2012), a number of other approaches exist which try to reduce the effect of greedy local optimum picking by taking into account how future decisions affect the overall model quality. Sarkar et al. (1994); Murthy and Salzberg (1995); Elomaa and Malinen (2003); Esmeir and Markovitch (2004) describe a few such approaches for decision tree induction, which improve the split point choice by investigating how the split criterion behaves for the given choices considering a certain number of additional splits. This type of look ahead strategy is very hard to parallelize in such a way as to ensure that overall computation time remains constant compared to the normal greedy method.

5.2.2 Ensemble learning

Ensembles use multiple models to obtain better predictive performance than could be obtained from any of the constituent models. Among the most important examples are bootstrap aggregating or bagging (Breiman 1996) and boosting (Schapire 1990). These methods are natural candidates for parallelization (Yu and Skillicorn 2001; Lazarevic and Obradovic 2002; Dai et al. 2005). However, a high degree of accuracy comes at the price of interpretability as these methods do not result in a single interpretable model, which is contrary to the goal of Widening.

Breiman (2001)'s random forests are particularly interesting in this context as some of the mechanisms to ensure a diverse ensemble, in particular data and feature subsetting, strongly resemble data-driven diversity for Widening. But the main difference remains: Ensembles aim to combine multiple models to improve performance, Widening aims to find one better model.

5.2.3 Meta heuristics

Learners based on stochastic learning algorithms, such as genetic algorithms are naturally parallelizable. Parallelization can be achieved by way of independent parallel execution of independent copies of a genetic algorithm, followed by selecting the best of the obtained results. This results in improved accuracy (Talia 2002). Examples include GA-MINER (Flockhart and Radcliffe 1996), REGAL (Giordana and Neri 1995), and G-NET (Giordana et al. 1997). This is probably the most similar approach to Widening as discussed here. However, Widening aims at exploring the search space in a structured way as opposed to the randomized nature of these other methods.

In genetic algorithms, selection operators such as roulette wheel selection (Bäck 1996), stochastic universal sampling (Baker 1989), or tournament selection (Bäck 1996) introduce diversity by selecting not only the best, but also some not so well-performing models for a population, without explicitly calculating similarities. While in genetic algorithms these worse models contribute to diversity mostly during recombination, the goal in Widening is that some of them will eventually lead to better solutions in other areas of the model space.

There are also similarities to multistart machine learning (Dick et al. 2014). However, whereas they restart new greedy searches with different random seeds and leave those runs independent, Widening aims to implicitly or explicitly force those different runs to focus on diverse areas of the search space.

5.2.4 Monte Carlo tree search

Related to meta heuristics but with a background in simulation, a recent approach to use Monte Carlo tree search to find a diverse set of patterns by Bosc et al. (2018) bears a couple of similarities to Widening. Although the original article proposes this concept for the sequential discovery of increasingly better patterns (hence the "anytime" in the paper's title), the concept has similarities to Widening in that it aims to better explore the model space. However, the sequential nature of this approach is essential in order to enforce diversity by analyzing the previously known solutions. Widening aims to do the same without knowing what the previously discovered solutions look like. Nevertheless, it was refreshing to see that the authors employ a similar visual representation of searching through model space and discovering diverse solutions.

5.2.5 Federated learning

An approach that aims less at performance improvement but tries to preserve the privacy of data should be mentioned here as well. Federated Learning (a good overview is provided by Kairouz et al. (2019)) learns distributed models on distributed data and later only aggregates those learned models but does not share the actual data that was used for training. The goal here is different to Widening, of course, but some aspects of the Widening framework can be applied nicely as this is an interesting analogy to data-driven diversity.

5.3 Greedy search algorithm improvement

Finally, there is a wealth of literature focusing on the improvement of (greedy) search algorithms in general. Approaches such as beam search are clearly relevant here, but, as we will see, these types of improvements do not facilitate the complete exploration of the model search space and therefore tend to still focus on local optima. The look ahead strategy (mentioned above) has been utilized to improve greedy heuristics in general as well (Sarkar et al. 1994). Shell et al. (1994) present an approach for incorporating diversity within the cost function used to select intermediate solutions. Harvey and Ginsberg (1995) use the observation that, in most cases, failing to find the optimal solution can be explained by a small number of erroneous decisions along the current path. Therefore, their "improved" search first explores the left-most child at a given depth as suggested by the current heuristic function. If no solution is found, all leaf nodes are tried that differ in just one decision point from the heuristic function choice. The algorithm iteratively increases the deviation from the greedy heuristic. The Widening proposed here performs a similar search for alternatives, but in parallel. Felner et al. (2003) explore the idea of adding diversity by a simple k-best first search and shows empirically that that is superior to the greedy best first search heuristic.

5.4 Parallel local search

Local search algorithms are a class of approximate algorithms widely used for combinatorial optimization problems. In order to improve the efficiency of the local search, different types of parallel local search are distinguished. Codognet et al. (2018) present a survey of parallel local searches. It considers different types of local search algorithms (simulated annealing, tabu search and others) and presents existing approaches to parallelizing them. In contrast to the approaches discussed here, the parallel local search algorithms presented have the goal of improving the efficiency for obtaining a good enough solution, while not necessarily looking for the optimal or a better solution, or even trying to escape local optima. The survey classifies the algorithms in two main classes—parallel single walk and parallel multiple walk. In the single walk algorithms one or many steps of the single walk are performed in parallel. At each step, the neighborhood is distributed among parallel workers, (the authors introduce the concept of *distributed neighborhood*, for which the goal is improving the speed-up, i.e. efficiency, which is not our goal). In the framework discussed here, we aim to split the entire workspace via Widening, and to avoid communication between the parallel workers. Our goal is to escape local optima and thus alleviate the problem with machine learning heuristics. In contrast, the distributed neighborhood that is investigated by parallel single walk algorithms is isomorphic to a traditional neighborhood, and thus the same quality solution is discovered.

The notion of multiple walks as a way to parallelize local search is slightly closer to our notion of Widening. In this approach multiple walks are made through the search space simultaneously. The multiple walks that are discussed in this work are non-interactive and interactive, i.e. with and without communication. For the multiple walks without interaction, the parallelization is achieved by simply starting the algorithms

multiple times. The different paths of the walks are realized by different parameters or starting points. While this does lead to higher probability of finding a good solution—and it does indeed improve the efficiency if the goal is finding a good enough solution—this approach is very simplistic and does not take into account the goal to explore the entire search space, as we discussed earlier in the section on Diversity-Driven Widening. The interactive approaches are similar to the ones with diversity measures that require communication which, as we pointed out earlier, results in problems in terms of computational effort with increasing parallelism.

5.5 Communication reduction

Many approaches exist to reduce communication overhead when parallelizations focus on distributing the discovery of one best solution. They range from black board techniques that utilize a central information sharing resource to synchronize their activities to more specific solutions that limit communication overhead such as described by Hamadi et al. (2009) for parallel SAT solving. Quite a few of the approaches discussed above also touch upon this issue. However, no general attempt for machine learning algorithms has been made.

6 Summary

In this article we have summarized half a decade of earlier work and presented an updated and expanded formalization of the concept of Widening. We also discuss an aggregate of earlier results, highlighting potential pitfalls and providing an intuition for different approaches to realize Widening in practice. Diversity-driven widening can outperform naive approaches like Top-k and the distinction between explicit and implicit diversity can have a large effect on communication between parallel workers. We highlight the Bucket Selector as a promising approach for efficient, practical implementations of Widening across a broad spectrum of model families which has limited communication overhead yet still delivers promising performance improvements.

Developing widening mechanisms for other machine learning models will help our understanding of practical ways to implement widening—simply parallelizing existing sequential algorithms is unlikely to be a promising alternative. Instead it will be necessary to reconsider the original goal, which was infeasible to implement, and hence resulted in the employed heuristic. The general framework presented here can then be used to formalize and guide realizations of other approaches to widening the search for better models. Widened search algorithms could potentially also be extended to provide an estimate of the distance to the optimal solution, thus allowing us to estimate how much additional time or resources would be needed to get within a certain distance to the best model.

Clearly, more practical work needs to be done to investigate different methods for different models in order to apply these techniques in practice. We believe that the concepts presented here provide a solid foundation for this exciting area of research.

Acknowledgements We thank Zaenal Akbar, Violeta Ivanova, Oliver Sampson, and Leonard Wörteler for some related work and help with implementations. Thanks also to the reviewers of earlier incarnations of this paper for their continuous constructive (and sometimes entertaining) feedback.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Agrawal R (1994) Fast algorithms for mining association rules. In: the Proceedings of 20th international conference on very large databases (VLDB), pp 487–499

Agrawal R, Shafer JC (1996) Parallel mining of association rules. IEEE Trans Knowl Data Eng 8(6):962–969

Akbar Z, Ivanova VN, Berthold MR (2012) Parallel data mining revisited. better, not faster. In: Hollmén J, Klawonn F, Tucker A (eds) Advances in intelligent data analysis XI. Springer Berlin Heidelberg, Berlin, pp 23–34

Akl SG (2002) Parallel real-time computation: sometimes quantity means quality. Comput Inform 21:455–487

Bäck T (1996) Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press, Oxford

Baker J (1989) Reducing bias and inefficiency in the selection algorithm. In: Proceedings of the second international conference on genetic algorithms, pp 14–21

Bosc G, Boulicaut JF, Raissi C, Kaytoue M (2018) Anytime discovery of a diverse set of patterns with Monte Carlo tree search. Data Min Knowl Disc 32:604–650

Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

Charytanowicz M, Niewczas J, Kulczycki P, Kowalski PA, Łukasik S, Zak S (2010) Complete gradient clustering algorithm for features analysis of x-ray images. In: Information technologies in biomedicine. Springer, pp 15–24

Chu CT, Kim SK, Lin YA, yu Y, Bradski G, Ng A, Olukotun K (2006) Map-reduce for machine learning on multicore. Adv Neural Inf Process Syst 19:281–288

Codognet P, Munera D, Diaz D, Abreu S (2018) Parallel local search. Springer, Cham, pp 381–417

Dai J, Lee J, Wang MC (2005) Efficient parallel data mining for massive datasets: Parallel random forests classifier. In: Proceedings of the international conference on parallel and distributed processing techniques and applications, pp 1142–1148

Darlington J, Yk Guo, Sutiwaraphun J, To HW (1997) Parallel induction algorithms for data mining. In: Liu X, Cohen P, Berthold M (eds) Advances in intelligent data analysis reasoning about data. Springer Berlin Heidelberg, Berlin, pp 437–445

Dean J, Ghemawat S (2008) MapReduce: simplified data processing on large clusters. Commun ACM 51(1):137–150

Dhillon IS, Modha DS (2000) A data-clustering algorithm on distributed memory multiprocessors. In: Zaki MJ, Ho CT (eds) Large-scale parallel data mining. Springer Berlin Heidelberg, Berlin, pp 245–260

Dick T, Wong E, Dann C (2014) How many random restarts are enough. Tech. rep., Carnegie Mellon University

Elomaa T, Malinen T (2003) On lookahead heuristics in decision tree learning. In: Zhong N, Raś ZW, Tsumoto S, Suzuki E (eds) Foundations of intelligent systems. Springer Berlin Heidelberg, Berlin, pp 445–453

Esmeir S, Markovitch S (2004) Lookahead-based algorithms for anytime induction of decision trees. In: Proceedings of the international conference on machine learning, pp 257–264
Felner A, Kraus S, Korf RE (2003) KBFS: K-best-first search. Ann Math Artif Intell 39:19–39
Fillbrunn A (2019) Effektives widening mit hashbasierter partitionierung des hypothesenraums. PhD thesis, University of Konstanz
Fillbrunn A, Berthold MR (2015) Diversity-driven widening of hierarchical agglomerative clustering. In: Fromont E, De Bie T, van Leeuwen M (eds) Advances in intelligent data analysis XIV. Springer, Cham, pp 84–94
Fillbrunn A, Wörteler L, Grossniklaus M, Berthold MR (2017) Bucket selection: a model-independent diverse selection strategy for widening. In: International symposium on intelligent data analysis. Springer, pp 87–98
Flockhart IW, Radcliffe NJ (1996) A genetic algorithm-based approach to data mining. In: Proceedings of the second international conference on knowledge discovery in databases, pp 299–302
Garg A, Mangla A, Gupta N, Bhatnagar V (2006) Pbirch: a scalable parallel clustering algorithm for incremental data. In: Proceedings of the international database engineering and applications symposium, IDEAS, pp 315–316
Giordana A, Neri F (1995) Search-intensive concept induction. Evol Comput 3(4):375–419
Giordana A, Anglano C, Giordana A, Bello GL, Saitta L (1997) A network genetic algorithm for concept learning. In: Proceedings of the 7th international conference on genetic algorithms, pp 434–441
Goldberg DE, Richardson J (1987) Genetic algorithms with sharing for multimodal function optimization. In: Proceedings of the second international conference on genetic algorithms on genetic algorithms and their application, L. Erlbaum Associates Inc., Hillsdale, NJ, USA, pp 41–49
Hamadi Y, Jabbour S, Sais L (2009) Control-based clause sharing in parallel sat solving. In: Proceedings of the 21st international joint conference on artificial intelligence, pp 499–504
Han EH, Karypis G, Kumar V (2000) Scalable parallel data mining for association rules. IEEE Trans Knowl Data Eng 12:337–352
Harvey WD, Ginsberg ML (1995) Limited discrepancy search. In: Proceedings of the 14th international joint conference on artificial intelligence, pp 607–613
Ivanova V, Berthold MR (2013) Diversity-driven widening. In: Proceedings of the 12th international symposium on intelligent data analysis(IDA 2013)
Judd D, McKinley PK, Jain AK (1998) Large-scale parallel data clustering. IEEE Trans Pattern Anal Mach Intell 4(8):871–876
Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, Bonawitz K, Charles Z, Cormode G, Cummings R, D'Oliveira RGL, Rouayheb SE, Evans D, Gardner J, Garrett Z, Gascón A, Ghazi B, Gibbons PB, Gruteser M, Harchaoui Z, He C, He L, Huo Z, Hutchinson B, Hsu J, Jaggi M, Javidi T, Joshi G, Khodak M, Konečný J, Korolova A, Koushanfar F, Koyejo S, Lepoint T, Liu Y, Mittal P, Mohri M, Nock R, Özgür A, Pagh R, Raykova M, Qi H, Ramage D, Raskar R, Song D, Song W, Stich SU, Sun Z, Suresh AT, Tramèr F, Vepakomma P, Wang J, Xiong L, Xu Z, Yang Q, Yu FX, Yu H, Zhao S (2019) Advances and open problems in federated learning. arXiv:1912.04977
Kantabutra S, Couch AL (2000) Parallel k-means clustering algorithm on NOWs. NOCTEC Tech J 1:243–247
Kargupta H, Chan P (2000) Advances in distributed and parallel knowledge discovery. AAAI/MIT Press, Cambridge
Kufrin R (1997) Decision trees on parallel processors. In: Geller J, Kitano H, Suttner CB (eds) Parallel processing for artificial intelligence 3. Machine intelligence and pattern recognition, vol 20. North-Holland, Amsterdam, pp 279–306
Kumar V, Ranka S, Singh V (2001) Special issue on high-performance data mining. Academic Press, London
Lazarevic A, Obradovic Z (2002) Boosting algorithms for parallel and distributed learning. Distrib Parallel Databases 11:203–229
Ma Z, Gu L (2010) The limitation of MapReduce: A probing case and a lightweight solution. In: In Proceedings of the 1st international conference on cloud computing, GRIDs, and virtualization, pp 68–73
Meinl T, Ostermann C, Berthold MR (2011) Maximum-score diversity selection for early drug discovery. J Chem Inf Model 51(2):237–247

Meng X, Bradley JK, Yavuz B, Sparks ER, Venkataraman S, Liu D, Freeman J, Tsai DB, Amde M, Owen S, Xin D, Xin R, Franklin MJ, Zadeh R, Zaharia M, Talwalkar A (2016) MLlib: machine learning in apache spark. J Mach Learn Res 17(34):1–7
Mitchell TM (1997) Machine learning. McGraw-Hill Education, New York
Murthy S, Salzberg S (1995) Lookahead and pathology in decision tree induction. In: IJCAI(2)
Olson CF (1995) Parallel algorithms for hierarchical clustering. JPC 21
Parthasarathy S, Zaki MJ, Ogihara M, Li W (2001) Parallel data mining for association rules on shared-memory multiprocessors. Knowl Inf Syst 3:1–29
Ruspini EH (1970) Numerical methods for fuzzy clustering. Inf Sci 2(3):319–350
Sampson OR, Borgelt C, Berthold MR (2018) Communication-free widened learning of Bayesian network classifiers using hashed Fiedler vectors. In: Duivesteijn W, Siebes A, Ukkonen A (eds) Advances in Intelligent Data Analysis XVII, Springer International Publishing, no. 11191 in Lecture Notes in Computer Science, pp 264–277
Sarkar U, Chakrabarti P, Ghose S, Desarkar S (1994) Improving greedy algorithms by lookahead-search. J Algorithms 16(1):1–23
Schapire RE (1990) The strength of weak learnability. Mach Learn 5:197–227
Shafer J, Agrawal R, Mehta M (1996) SPRINT: a scalable parallel classifier for data mining. In: Proceedings of the 22nd VLDB conference, pp 544–555
Shell P, Rubio JAH, Barro GQ (1994) Improving search through diversity. In: Proceedings of the 12th national conference on artificial intelligence, pp 1323–1328
Shintani T, Kitsuregawa M (1996) Hash based parallel algorithms for mining association rules. In: In Proceedings of 4th international conference on parallel and distributed information systems, pp 19–30
Siebes A (2012) Queries for data analysis. In: Proceedings of the 11th international conference on advances in intelligent data analysis, pp 7–22
Sokal RR, Rohlf FJ (1962) The comparison of dendrograms by objective methods. Taxon 11(2):33–40
Srivastava A, Han EH, Kumar V, Singh V (1999) Parallel formulations of decision-tree classification algorithms. DMKD 3(3):237–261
Talia D (2002) Parallelism in knowledge discovery techniques. In: 6th International conference on applied parallel computing advanced scientific computing, vol 2367, pp 127–138
Verma A, Llorà X, Goldberg DE, Campbell RH (2009) Scaling genetic algorithms using MapReduce. In: Intelligent systems design and applications, pp 13–18
Wolberg WH, Mangasarian OL (1990) Multisurface method of pattern separation for medical diagnosis applied to breast cytology. Proc Nat Acad Sci 87(23):9193–9196
Yu C, Skillicorn D (2001) Parallelizing boosting and bagging. Queen's University, Kingston, Canada, Technical Report
Zaki M (1999) Parallel and distributed association mining: a survey. IEEE Concurr 7(4):14–25
Zaki M, Ho C (2000) Large-scale parallel data mining. Springer, Berlin
Zaki MJ, Pan Y (2002) Introduction: recent developments in parallel and distributed data mining. Distrib Parallel Databases 11(2):123–127
Zaki MJ, Parthasarathy S, Ogihara M, Li W (1997) Parallel algorithms for discovery of association rules. DMKD 1(4):343–373
Zaki MJ, Ho CT, Agrawal R (1998) Parallel classification on SMP systems. In: The 1st workshop on high performance data mining
Zhao W, Ma H, He Q (2009) Parallel K-means clustering based on MapReduce. In: Proceedings of the 1st international conference on cloud computing, pp 674–679