Widened KRIMP: Better Performance Through Diverse Parallelism

Oliver Sampson and Michael R. Berthold

Chair for Bioinformatics and Information Mining
Department of Computer and Information Science
University of Konstanz, Germany

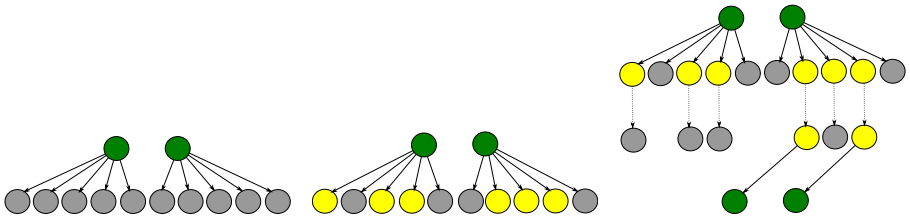
Abstract. We demonstrate that the previously introduced Widening framework is applicable to state-of-the-art Machine Learning algorithms. Using KRIMP, an itemset mining algorithm, we show that parallelizing the search finds better solutions in nearly the same time as the original, sequential/greedy algorithm. We also introduce Reverse Standard Candidate Order (RSCO) as a candidate ordering heuristic for KRIMP.

1 Introduction

Research into parallelism in Machine Learning has primarily focused on reducing the execution time of existing algorithms, e.g., parallelized K-MEANS [23,17,14,26] and DBSCAN [11,4,7]. There have been some exceptions, such as *metalearning* and *ensemble methods* [9], which have employed heterogeneous algorithms in parallel, and [3], which describes the application to simple examples. Recent work [2,15] describes *Widening*, a framework for employing parallel resources to increase accuracy. With Widening, measures of diversity are used to guarantee the parallel search paths' exploration of disparate regions within a solution space, thereby stepping around the common greedy algorithmic tendency to find local optima. Thus far, work has concentrated on a proof-of-concept and demonstrative application to algorithms for solving the SET COVER PROBLEM and the creation of Decision Trees. This document describes the same approach, but with a state-of-the-art algorithm, KRIMP [24].

KRIMP finds "interesting" itemsets from a transactional database via the Minimum Description Length (MDL) principle [21]. The authors summarize the method as "the best set of patterns [being] the set of patterns that describes the data best," where the best set of itemsets is the set that provides the highest compression using MDL. The algorithm not only provides a solution to the problem of pattern explosion, thereby greatly reducing the set of itemsets used to generate association rules, but provides exceptional performance in other applications such as classification [24].

This paper demonstrates that it is possible to apply Widening to find even more interesting sets of itemsets than those found by the standard KRIMP algorithm.



(a) Two models with five refinements.

(b) Three selected refinements from each group of five.

(c) Two selections (yellow) from the group of six, ready for the next iteration (green).

Fig. 1: Refine and Select with $l = 3$ and $k = 2$.

2 Widening

Given the set of all models, $\mathcal{M}$, that describe the solution space for a typical greedy machine learning algorithm, $m(\cdot) \in \mathcal{M}$ is a model which describes a portion of the solution space. It is iteratively refined by a *refinement operator*, $r(\cdot)$, based on a subset, x, from a training dataset $\mathcal{T}$, i.e., $m'(\cdot) = r(m(\cdot), x), x \subseteq \mathcal{T}$. The derivation of a Decision Tree is one example of this process [15].

In contrast to that above, in the Widening framework a set of models, $M \subseteq \mathcal{M}$, is the result of a refinement operator based on data from $\mathcal{T}$ and a *diversity metric*, Δ, which describes some minimum difference between the resulting models, $\{m'_1, \cdots, m'_l\}$, i.e., $M = r_\Delta(m) = \{m'_1, \cdots, m'_l\}$. For clarity, the data elements from the training data are eliminated from the notation.

A *selection operator*, $s_{top-k}(\cdot)$ is employed to select the best k models at each step [15]. $M_{i+1} = s_{top-k}(r_\Delta(M_i))$. The results of the selection operation are further refined until some stop condition is met. This iterative refine-and-select process, as depicted in Figure 1, is conceptually similar to a *beam search* [19].

3 The KRIMP Algorithm

In the area of Itemset Mining, KRIMP finds the set of "most interesting" itemsets in a transaction database based on MDL, i.e., KRIMP defines the best model, m, as the model that maximizes the compression of a transaction database, $\mathcal{D}$, encoded with that model and the compression of the model itself [21,24].

Given a database $\mathcal{D}$ composed of transactions $t \in \mathcal{D}$, KRIMP finds the subset of itemsets, X, from the set of all itemsets, $\mathcal{X}$, that maximally compresses $\mathcal{D}$. KRIMP calculates the size of the encoded database using the codelengths of *prefix-free codes*, which are related to the frequency of the appearance of an itemset, $x \in X$, in the database and to the Shannon entropy: $L(x) = -\log_2 P(x)$, where $L(x)$ is the codelength measured in bits of an item or itemset in the database, and $P(x)$ is the relative frequency of the item's or itemset's appearance in the database [24].

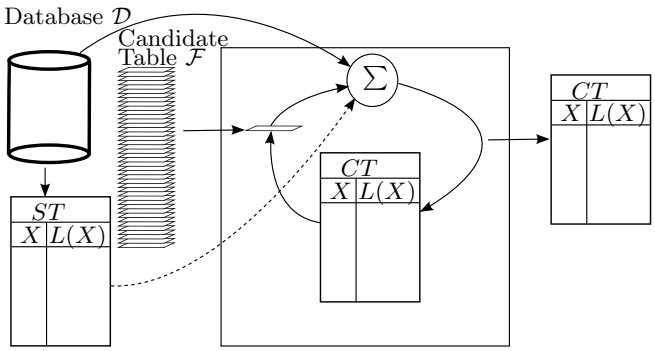


Fig. 2: The KRIMP algorithm.

KRIMP begins with the generation of the Standard Code Table, ST, which is merely a code table comprised of only the individual items from the set of all single items, $\mathcal{I}$. The codelength of a given transaction, $L(t)$ is the sum of its compositional codelengths.

KRIMP then iterates through a list of candidate itemsets, $\mathcal{F}$, generated by an algorithm external to KRIMP, such as AFOPT [18] or APRIORI [1]. Each of the candidate itemsets from $\mathcal{F}$ is temporarily inserted into the code table CT, where all relative frequencies are determined and the compression evaluated. If it provides better compression, it is kept as part of CT and if not, it is discarded [24]. A general flow diagram is depicted in Figure 2.

The size of the encoded database, $L(\mathcal{D}|CT)$, is the sum of the encoded lengths of all transactions. The size of the encoded Code Table, $L(CT|\mathcal{D})$, is the size of each code plus the lengths of the encoded itemsets, for which the single items from ST are used. The compressed MDL size of the database is the size of the encoded database plus the size of the encoded code table. $L(\mathcal{D}, CT) = L(\mathcal{D}|CT) + L(CT|\mathcal{D})$ [24]

Both $\mathcal{F}$ and CT are ordered heuristically to maximize compression. $\mathcal{F}$ is ordered according to the *Standard Candidate Order*, which orders primarily by the itemsets' *support* in descending order, secondarily by cardinality in descending order, and tertiarily by lexicographical order, as a tie-breaker. The rationale is that itemsets with larger support are likelier to cover more transactions and are evaluated first. Itemsets with the same support are sorted secondarily by cardinality, because larger itemsets cover more items in each transaction, reducing the number of itemsets or items required to cover a transaction [24].

CT is ordered using the *Standard Cover Order*, which orders primarily by descending cardinality, secondarily by descending support, and tertiarily lexicographically, again as a tie-breaker. The rationale is that larger itemsets are preferred for their ability to cover more of each transaction. Of those, the ones with a larger support are more likely to cover more transactions in the database, thereby providing shorter codes [24].

KRIMP also includes a post-processing step for each iteration called PRUNING. If the relative frequency of any itemset in CT decreases as a result of adding a new itemset, CT is re-evaluated with each itemset in CT singly removed. If any of the itemsets' temporary absence from CT enables a better compression, it is discarded. Results in [24] indicate that PRUNING improves overall compression performance marginally, but can dramatically reduce the number of itemsets providing that level of compression.

4 Widenend Krimp

A given path through a KRIMP solution space is based on two things: 1) the order with which the candidate itemsets from $\mathcal{F}$ are evaluated, because the acceptance of a particular itemset into CT influences which itemsets are accepted in later iterations, and 2) the order in which the itemsets in CT are used to cover the database. Varying either of these two heuristics' orderings varies the solution path through the solution space and introduces diversity from the other paths taken.

For use as Δ in the refining function, $r_{\Delta}(\cdot)$, two *explicit* measures and one implicit measure of diversity are investigated here. Explicit measures *p-dispersion-min-sum* and *p-dispersion-sum* select maximally diverse subsets of candidates from the candidate table. Implicit method, *Directed Placement*, is investigated with respect to the ordering of the itemsets evaluated for covering the transactions in CT.

p-dispersion-min-sum maximizes the sum of minimum distances between pairs of members of the selected subset [20].

Definition 1 ***p-dispersion-min-sum.***[1] *Given a set $\mathcal{F} = \{F_1, \cdots, F_n\}$ of n itemsets and l, where $l \in \mathbb{N}$ and $l \leq n$, and a distance measure $Jaccard(F_i, F_j) : F_i, F_j \in \mathcal{F}$ between items F_i and F_j, the l-diversity problem is to select the set $F : F \subseteq \mathcal{F}$, such that*

$$F^* = \max_{\substack{F \subseteq \mathcal{F} \\ |F|=l}} f(F), \text{where } f(F) = \sum_{i=1}^{l} \min_{1 \leq j \leq l, i \neq j} Jaccard(F_i, F_j), F_i, F_j \in F [20][16] \tag{1}$$

p-dispersion-sum maximizes the distance between all members of the selected subset.

Definition 2 ***p-dispersion-sum.*** *Given a set $\mathcal{F} = \{F_1, \cdots, F_n\}$ of n itemsets and l, where $l \in \mathbb{N}$ and $l \leq n$, and a distance measure $Jaccard(F_i, F_j) : F_i, F_j \in$*

[1] The canonical names from the literature, p-dispersion-min-sum and p-dispersion-sum, are maintained here, even though in this context, they should be called "l-dispersion-min-sum" and "l-dispersion-sum."

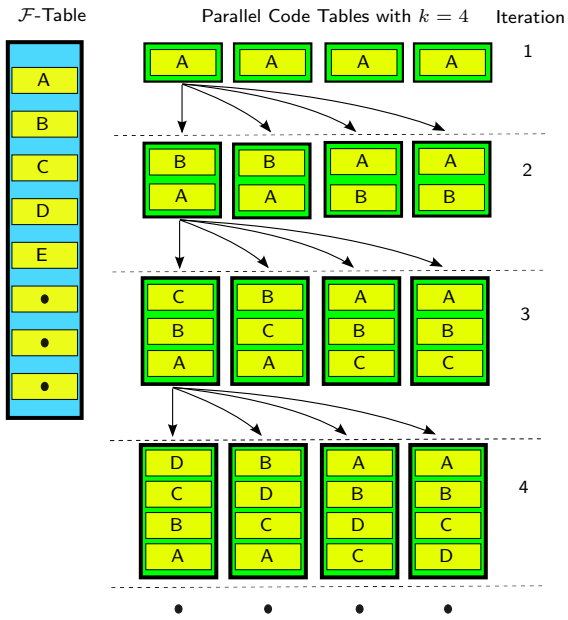


Fig. 3: At each iteration, the next $F \in \mathcal{F}$ is inserted into each of the parallel code tables at a depth $\frac{i}{l}$. Refinement is shown only for one table in each iteration.

$\mathcal{F}$ between itemsets F_i and F_j, the l-diversity problem is to select the set $F : F \subseteq \mathcal{F}$, such that

$$F^* = \max_{\substack{F \subseteq \mathcal{F} \\ |F|=l}} f(S), \textit{where } f(F) = \frac{1}{l(l-1)} \sum_{i=1}^{l} \sum_{j>1}^{l} Jaccard(F_i, F_j) \textit{[20,12]} \qquad (2)$$

p-dispersion-sum has the side-effect of pushing the selected members to the boundaries of the original set. This results in selected sets that are less diverse and representative of the dataset than those that are selected by p-dispersion-min-sum [20].

The Directed Placement diversity heuristic functions by inserting the next candidate itemset, $F \in \mathcal{F}$ at a position with different fractional depths into l parallel instances of CT. The depth inserted into CT is a function of l, where the depth is $\frac{i}{l}|CT| : i = 1, \ldots, l$. Because the role of each itemset in the covering algorithm is dependent on its position in CT, positioning F at different depths explores diverse solution paths. This method of diversity is implicit, because the diversity between different CT tables is not measured directly. See Figure 3.

An additional heuristic ordering of $\mathcal{F}$ called *Reverse Standard Candidate Order* (RSCO) is introduced here. It orders the candidate itemsets primarily by cardinality in ascending order, secondarily by support in descending order, and tertiarily by lexicographical order as a tie-breaker. In combination with the Standard Cover Order heuristic for covering transactions, RSCO attempts to

mimic the PRUNING subalgorithm; candidate itemsets with larger cardinality are examined later but are inserted before the smaller itemsets already in CT. With Standard Cover Order, small itemsets whose potential ability to efficiently cover transactions are "shadowed" by larger itemsets and have a lower relative frequency used for the compression calculation. In contrast, using RSCO, smaller itemsets that may have a beneficial effect, yet show up too late in the list to be considered with Standard Candidate Order, can still be evaluated.

5 Experimental Results

Compression of $\mathcal{D}$ with CT, $L(\mathcal{D}, CT)$, is compared to a baseline compression of $\mathcal{D}$ encoded using only ST, $L(\mathcal{D}, ST)$. This paper follows the convention of [24] in using the compression ratio measured in percentages, where lower is better. $L\% = L(\mathcal{D}, CT)/L(\mathcal{D}, ST) * 100$. [24]

The notation $|CT \setminus \mathcal{I}|$ indicates the number of non-singleton itemsets used. For a given compression level, a smaller number of itemsets is considered more interesting. KRIMP optimizes for both $L\%$ and $|CT \setminus \mathcal{I}|$ by evaluating $L\%$ first, and then bettering $|CT \setminus \mathcal{I}|$ with the PRUNING subalgorithm.

All experiments were conducted in KNIME [6] and used APRIORI [1,8] with a *minsupport* of 1 to generate the set of *closed itemsets*. The datasets used were the LUCS-KDD-DN [10] discretized versions of the Breast Cancer Wisconsin (Original) [25] (*Breast*) and Pima Indians Diabetes Data Set [22] (*Pima*) datasets available from the UCI-ML Data Repository [5].

Evaluations in Sections 5.1 and 5.2 compare three methods, KRIMP$_{Greedy}$, KRIMP$_{RSCO}$ and KRIMP$_{Diverse}$. KRIMP$_{Greedy}$ refers to the baseline "standard" KRIMP implementation used within KNIME; KRIMP$_{RSCO}$ refers to the implementation in KNIME, using RSCO for ordering $\mathcal{F}$ rather than Standard Candidate Order, because the results with RSCO for the *Breast* and *Pima* datasets were actually better than Standard Candidate Order;[2] and KRIMP$_{Diverse}$ refers to KRIMP with a method of diversity being evaluated. KRIMP has two performance metrics for a model; solution pairs are shown in the form of $\langle L\%, |CT \setminus \mathcal{I}| \rangle$. Results are shown with and without PRUNING for KRIMP$_{Greedy}$ and KRIMP$_{RSCO}$. All experiments with KRIMP$_{Diverse}$ were performed without PRUNING because we felt it would introduce another variable of diversity for which we were not controlling.

Experimental solution pairs are also shown at the position found $\langle l, k \rangle$, where l is the number of refined models and k is the number of models selected according to compression performance.

5.1 Diverse Candidate Selection

Both of these methods of subset selection are performed with replacement, because early experiments without replacement on $\mathcal{F}$ candidate tables generated

[2] This is just an artifact for these two datasets. Preliminary results not shown here demonstrate that RSCO does indeed perform better than SCO for some datasets, albeit not consistently across all datasets tested.

	Heuristic		Pruning	$L\%$	$\lvert CT \setminus \mathcal{I} \rvert$
	$\mathcal{F}$ (Candidate Table)	CT (Code Table)			
KRIMP$_{Greedy}$	Standard Candidate Order	Standard Cover Order	no	18.11	29
			yes	17.61	28
KRIMP$_{RSCO}$	RSCO	Standard Cover Order	no	17.86	28
			yes	17.82	26
KRIMP$_{Diverse}$	p-dispersion-min-sum + RSCO	Std. Cover Order	no	17.97	28
	p-dispersion-sum + RSCO	Std. Cover Order		19.42	34
	RSCO	Directed Placement		**17.39**	**26**

Table 1: *Breast* Dataset Results Summary.

from closed itemsets demonstrated that there were simply not enough candidate itemsets for evaluation to generate solutions sets of reasonable performance for larger values of k. With replacement, in order for the algorithm to come to completion, the first element in $\mathcal{F}$ is removed after each iteration, ensuring the algorithm's completion after $|\mathcal{F}|$ iterations. This method naturally entails a dependency on the initial ordering of $\mathcal{F}$.

A summary of the results for the *Breast* dataset using the p-dispersion-min-sum diversity metric for $\mathcal{F}$ candidate selection can be found in Table 1. The experiments were run with all combinations of $l \in \{5, 10, 20, 30, 40, 50\}$ and $k \in \{1, 5, 10, 15\}$. The best solution for KRIMP$_{Diverse}$ with p-dispersion-min-sum, $\langle 17.97\%, 28\rangle$ was found at $\langle l, k\rangle = \langle 50, 10\rangle$, which was better than KRIMP$_{Greedy}$ without PRUNING $\langle 18.11\%, 29\rangle$, but not better than KRIMP$_{Greedy}$ with PRUNING $\langle 17.61\%, 28\rangle$. KRIMP$_{RSCO}$ with PRUNING performed even better at $\langle 17.82\%, 26\rangle$.

KRIMP$_{Diverse}$ with p-dispersion-sum was run with all combinations of $l \in \{5, 10, 20, 30, 40, 50\}$ and $k \in \{1, 5, 10, 20, 50\}$ and as expected, the best solution pair $\langle 19.42\%, 34\rangle$ was not nearly as good as that with p-dispersion-min-sum, and was found in an even larger search space of $\langle l, k\rangle = \langle 50, 30\rangle$.

The experiments with KRIMP$_{Diverse}$ with p-dispersion-min-sum were run over a smaller search space when compared to KRIMP$_{Diverse}$ with p-dispersion-sum after recognizing that the results had already reached the goal of beating one of the KRIMP$_{Greedy}$ scores.

Due to run-time constraints (See Section 6.) experiments were not performed with diversity-based candidate selection on the *Pima* dataset.

5.2 Diverse Cover Order

The results of the Directed Placement heuristic with RSCO as the Candidate Selection heuristic are also summarized in Tables 1 and 2. Experiments were performed on the *Breast* dataset with all combinations of $l \in \{5, 10, 20, 30, 40, 50\}$ and $k \in \{1, 5, 10, 15\}$. The heuristic found a solution $\langle 17.39\%, 26\rangle$ outperforming the best KRIMP variant. Additionally, the solution was found in a much smaller search space, when compared to Diverse Candidate Selection, with the best solution found first at $\langle l, k\rangle = \langle 10, 10\rangle$.

	Heuristic		Pruning	$L\%$	$\lvert CT \setminus \mathcal{I} \rvert$
	$\mathcal{F}$ (Candidate Table)	CT (Code Table)			
KRIMP$_{Greedy}$	Standard Candidate Order	Standard Cover Order	no	35.6	66
			yes	34.4	53
KRIMP$_{RSCO}$	RSCO	Standard Cover Order	no	34.3	63
			yes	33.7	**49**
KRIMP$_{Diverse}$	RSCO	Directed Placement	no	**32.9**	56

Table 2: *Pima* Dataset Results Summary.

The *Pima* dataset showed an interesting property in that paths to better solutions were sensitive to increasing k, the number of selected models to be refined in the next step, but not in l, the number of refinements in each step. Better results, $\langle 33.2\%, 61 \rangle$, than KRIMP$_{Greedy}$ and KRIMP$_{RSCO}$ both without PRUNING were found immediately at $\langle l, k \rangle = \langle 5, 5 \rangle$. In fact, all solutions found with the Directed Placement heuristic with RSCO showed better $L\%$ than either KRIMP$_{Greedy}$ or KRIMP$_{RSCO}$ with or without PRUNING. The best solution found for the *Pima* dataset was $\langle 32.9\%, 56 \rangle$ at $\langle l, k \rangle = \langle 5, 50 \rangle$. This result has significantly better compression and yields nearly the number of itemsets as KRIMP$_{Greedy}$, but not nearly as good as KRIMP$_{RSCO}$.

6 Discussion and Future Work

In general, absolute timing values are not necessary for timing comparisons. To a first order of approximation, KRIMP runs in $O(|\mathcal{F}| \times |\mathcal{D}| \times \theta)$ where θ is a factor describing the average length of CT during the entire execution of the algorithm. (It should be noted that the authors of [24] saw a performance improvement in execution speed after implementing the PRUNING subalgorithm, because of a smaller value of θ.) Accounting for application of a diversity measure and the use of a performance measurement for selection, KRIMP runs in $O((|\mathcal{F}|+\Delta+\Psi)\times |\mathcal{D}| \times \theta)$, where $O(\Delta)$ is the measure of the complexity of the diversity heuristic, and $O(\Psi)$ is a measure of the complexity of the performance measurement.

Although p-dispersion-min-sum was able to find comparable results to the standard KRIMP implementation (better than KRIMP$_{Greedy}$ without PRUNING), the computational cost is significant. Selecting a subset of p diverse elements from a larger set is a variation of the p-dispersion problem and is $\mathcal{NP}$-hard [13]. Moreover, a comparison of this metric to p-dispersion-sum demonstrates what could be a pitfall for applying p-dispersion-sum: a much wider solution space had to be searched, $\langle l, k \rangle = \langle 50, 10 \rangle$ versus $\langle l, k \rangle = \langle 50, 30 \rangle$. Although at least one of these diversity measures fulfills the desire to show that widened data mining can find better solutions than the traditional greedy algorithm, it is insufficient for a requirement of finding better solutions in the same or less time than the traditional greedy algorithm, which is the ultimate goal of Widening.

Directed Placement, however, was able to significantly improve on the solution found by standard KRIMP in [24]. For the *Breast* dataset, the results were even better than the results found with KRIMP_{RSCO}. Directed Placement also showed a partially better solution with the *Pima* dataset. In comparison to the other diversity metrics presented here, Directed Placement has a much smaller overhead for generating diverse solution paths. It must be noted, however, that the claim of "better solutions in the same or faster time" in this case is not strictly accurate. For large values of $|\mathcal{F}|$ and $|\mathcal{D}|$, the influence of $O(\Delta)$ for the Directed Placement diversity heuristic becomes negligible. The evaluation of the models for selection, $O(\Psi)$ is also negligible for KRIMP, because it is merely a comparison of the best $L\%$. Additionally, Directed Placement provided the best Widened result in a significantly smaller search region than the other diversity heuristics.

Ideally, a Widened algorithm is able to find a better or the best solution in the same time as the traditional greedy algorithm. The pitfall of the methods described here is that both a performance evaluation (Ψ) and a synchronized comparison of results from the parallel workers are required. This would be avoided with a *communication-less* [15] approach where the parallel workers would be able to refine and select without requiring a synchronized comparison step. Additionally, although the better solutions found by Widened KRIMP meet the definition of "better," further research into how well the smaller sets perform as classifiers or in other KRIMP applications is necessary. The effects of including the PRUNING subalgorithm on the dataset compression, and the corresponding solution space paths also require further investigation, as does the magnitude and interplay between l and k for different datasets.

7 Conclusion

In this paper we have validated Widening for the first time using a state-of-the-art algorithm for itemset mining, KRIMP, and shown that it is possible to use the novel approach of Widening to find significantly better solutions than that of the traditional greedy algorithm by searching diverse regions of a solution space in parallel. We have also introduced RSCO, a new Candidate Table ordering heuristic for KRIMP that can provide even better results for some datasets.

References

1. Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases*, volume 1215, pages 487–499, 1994.
2. Zaenal Akbar, Violeta N. Ivanova, and Michael R. Berthold. Parallel data mining revisited. Better, not faster. In *Proceedings of the 11th International Symposium on Intelligent Data Analysis*, pages 23–34, 2012.
3. Selim G. Akl. Parallel real-time computation: Sometimes quantity means quality. In *Parallel Architectures, Algorithms and Networks, 2000. I-SPAN 2000. Proceedings. International Symposium on*, pages 2–11. IEEE, 2000.

4. Domenica Arlia and Massimo Coppola. Experiments in parallel clustering with DBSCAN. In *Euro-Par 2001 Parallel Processing*, pages 326–331. Springer, 2001.
5. Kevin Bache and Moshe Lichman. UCI Machine Learning Repository, 2013.
6. Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Christoph Sieb, Kilian Thiel, and Bernd Wiswedel. KNIME: The Konstanz Information Miner. In Christine Preisach, Hans Burkhardt, Lars Schmidt-Thieme, and Reinhold Decker, editors, *Data Analysis, Machine Learning and Applications - Proceedings of the 31st Annual Conference of the Gesellschaft für Klassifikation e.V. (GfKL 2007)*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 319–326, Berlin, Germany, 2007. Springer.
7. Christian Böhm, Robert Noll, Claudia Plant, Bianca Wackersreuther, and Andrew Zherdin. Data mining using graphics processing units. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems I*, pages 63–90. Springer, 2009.
8. Christian Borgelt and Rudolf Kruse. Induction of association rules: Apriori implementation. In *Compstat*, pages 395–400. Springer, 2002.
9. Philip Chan and Salvatore J. Stolfo. Experiments on multistrategy learning by meta-learning. In *In Proceedings of the Second International Conference on Information and Knowledge Management*, pages 314–323, 1993.
10. Frans Coenen. LUCS-KDD DN software, 2003.
11. Inderjit S. Dhillon and Dharmendra S. Modha. A data-clustering algorithm on distributed memory multiprocessors. In *Large-Scale Parallel Data Mining*, pages 245–260. Springer, 2000.
12. Marina Drosou and Evaggelia Pitoura. Comparing diversity heuristics. Technical report, Technical Report 2009-05. Computer Science Department, University of Ioannina, 2009.
13. Erhan Erkut. The discrete p-dispersion problem. *European Journal of Operational Research*, 46(1):48–60, 1990.
14. Reza Farivar, Daniel Rebolledo, Ellick Chan, and Roy Campbell. A parallel implementation of k-means clustering on GPUs. In *Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pages 340–345, 2008.
15. Violeta N. Ivanova and Michael R. Berthold. Diversity-driven widening. In *Proceedings of the 12th International Symposium on Intelligent Data Analysis (IDA 2013)*, 2013.
16. Paul Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 1901.
17. Sanpawat Kantabutra and Alva L. Couch. Parallel k-means clustering algorithm on nows. *NECTEC Technical journal*, 1(6):243–247, 2000.
18. Guimei Liu, Hongjun Lu, Jeffrey Xu Yu, Wang Wei, and Xiangye Xiao. AFOPT: An efficient implementation of pattern growth approach. In *Proceedings of the ICDM workshop on frequent itemset mining implementations*, 2003.
19. Bruce T. Lowerre. *The HARPY speech recognition system.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1976.
20. Thorsten Meinl. *Maximum-Score Diversity Selection.* PhD thesis, University of Konstanz, July 2010.
21. Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.
22. Jack W. Smith, J.E. Everhart, W.C. Dickson, W.C. Knowler, and R.S. Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In

Proceedings of the symposium on computer applications and medical care, volume 261, page 265, 1988.

23. Kilian Stoffel and Abdelkader Belkoniene. Parallel k/h-means clustering for large data sets. In Patrick Amestoy, Philippe Berger, Michel Daydé, Daniel Ruiz, Iain Duff, Valérie Frayssé, and Luc Giraud, editors, *Euro-Par99 Parallel Processing*, volume 1685 of *Lecture Notes in Computer Science*, pages 1451–1454. Springer Berlin Heidelberg, 1999.
24. Jilles Vreeken, Matthijs van Leeuwen, and Arno Siebes. Krimp: Mining itemsets that compress. *Data Mining and Knowledge Discovery*, 23(1):169–214, #jul# 2011.
25. William H. Wolberg and Olvi L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the national academy of sciences*, 87(23):9193–9196, 1990.
26. Weizhong Zhao, Huifang Ma, and Qing He. Parallel k-Means Clustering Based on MapReduce. In MartinGilje Jaatun, Gansen Zhao, and Chunming Rong, editors, *Cloud Computing*, volume 5931 of *Lecture Notes in Computer Science*, pages 674–679. Springer Berlin Heidelberg, 2009.