Table 7: A summary of common notations.

Symbol	Description
D, d	The dataset, and its dimensionality
x_i	A data point in D
c_j	The centroid of cluster S_j
$lb(i,j)$	The lower bound of x_i to cluster c_j
$lb(i)$	The Lower bound of x_i to its second nearest cluster
$ub(i)$	The Upper bound of x_i to its nearest cluster
$a(i)$, $a'(i)$	The indices of x_i's new and previous cluster
p	The pivot point of node N
$\|c_j\|$	The L2-norm of centroid c_j
$\delta(j)$	The centroid drift of c_j

APPENDIX

A. SUPPLEMENTARY MATERIALS

Algorithm 2: Selective Running of Knob Configurations

```
Output: two ground truth files: g1 and g2
for every test dataset D do
    F ← MetaFeatureExtraction(D, k, T);
    Run partial sequential methods;
    write(F, identifier of the optimal knob m_o, g1);
    Test the index method m_i with Ball-tree;
    if m_i is slower than m_o then
        write(F, not using index 1, g2);
    else
        Test index-single and index-multiple with m_o;
        if index method is the fastest then
            write(F, point traveseal 2, g2);
        else
            if index-single is the fastest then
                write(F, single traveseal 3, g2);
            else
                write(F, multiple traveseal 4, g2);
return g1 and g2;
```

A.1 Common Notations

Table 7 lists the common notations used in this paper.

A.2 Space Analysis of Index Methods

Recall Definition 1, building an extra data structure will increase the space overhead, the size of a node N is comparably small – each node is composed of two vectors (p, sv), four floats (r, ψ, num, h), and two pointers to child nodes (left and right) or a set of points in leaf nodes (the number is limited as the capacity f). Thus, we estimate the space of leaf nodes as $2d+4+f$, and internal nodes as $2d+6$. Then the overall space cost (number of floats) of all the nodes will be $\frac{n}{f}\cdot(2d+4+f)+q\cdot(2d+6) = n+\frac{n}{f}\cdot(2d+4)+q\cdot(2d+6)$, where $\frac{n}{f}$ and $q=\frac{n}{f}\cdot(1-2^{1-log_2\frac{n}{f}})$ are the numbers of leaf nodes and internal nodes.

Such an estimation is based on an ideal case where each leaf node has f points and the index is a balanced Ball-tree with a height $log_2\frac{n}{f}$, and we skip the summation for geometric sequence here. In Figure 10 and 18, we can find that the footprint of Ball-tree is comparable with those space-saving sequential methods in the most datasets. For the datasets that have high footprint, most of them are high-dimension, which means that they will have more leaf nodes that cover less points than those datasets with lower dimensions. Note that one advantage of Ball-tree is that its footprint is fixed once it is built, and will not change with the increasing of k, while most sequential methods will need much more space.

A.3 More Experiment Results

Figure 16 to 19 show a detailed comparison between sequential methods when setting $k=10$, which mainly enhances Section 7.2.2. We can observe similar trends with $k=100$, and the main difference is that smaller k has fewer data accesses and lower pruning ratio. Table 9 and 10 present the time of assignment and refinement which partially decompose the results the overall speedup in Table 6.

A.4 Reproducibility

Our code is available at Github:

`https://github.com/tgbnhy/fast-kmeans`

We show how to run our experiments with simple commands and operations in the description of the repository. Readers can download the datasets in Table 2 online by clicking the links shown there, and put into local folders, then compile the code using Maven (`http://maven.apache.org/`). Clustering tasks can be conducted locally by executing the given exemplar commands by terminals. All the algorithms will be run, and the results will be shown in the terminal to monitor the running progress. After all the algorithms end, detailed results on each comparison metric, such as speedup and the number of data accesses, will be written into the log files for further plotting. We also show readers how to interpret the evaluation results based on the terminal and logs files.

A.5 Future Opportunities

More Meta-Features. In our evaluations, we used multiple measures without extra costs. The main reason that we do not use other features that can be extracted using data profiling [18] or meta-feature extraction [59] is that, they need much time to extract, such as the model-based method by training decision trees. Since a higher precision will recommend a better algorithm and may greatly reduce the clustering time, it is promising to apply these features to improve the precision with fast meta-feature extraction.

New ML Models to be Adopted. We have tested multiple classical models to predict the optimal algorithm, while the loss function generally computes exact matches and does not consider the importance of ranked lists of algorithms. Hence, designing a specific machine learning model with a loss function like MRR, which we used to evaluate the accuracy, will be crucial to further improve the prediction accuracy. Furthermore, deep neural networks (DNN) have also shown significant successes in various classification tasks, and it will be interesting to design a specific network structure to input a sampled dataset as features directly to predict the optimal algorithm. It is worth noting that this is out of the scope of this work and our evaluation framework is orthogonal to the choice of ML model adopted in the learning part.

Discovering New Configuration Knobs. In this evaluation, we have compared all existing fast k-means clustering

Table 8: Evaluation of knob configuration: training time and prediction time.

Efficiency	BDT	**Basic features**					**+ Tree-features**					**+ Leaf-features**				
		DT	RF	SVM	kNN	RC	DT	RF	SVM	kNN	RC	DT	RF	SVM	kNN	RC
Training (ms)	-	1.63	577	3.74	1.73	4.40	1.89	573	4.35	1.39	4.54	2.53	598	5.42	1.43	4.36
Prediction (μs)	-	3.98	251	6.28	36.4	5.33	6.09	259	7.14	39	6.72	7.15	271	10.2	37.7	8.57
S-Training (ms)	-	1.98	660	29	1.85	4.90	2.81	764	53.3	2.33	4.55	4.75	897	54.3	1.87	4.68
S-Prediction (μs)	-	1.53	97	10.7	27.9	1.39	1.24	105	13.3	32.00	1.82	**1.35**	97.2	14.6	34.5	1.90

algorithms and `UniK` with our new settings. It is worth mentioning that the predicted algorithm is only the fastest among the group that we tested, and k-means can be further accelerated in the future. As shown in Algorithm 1, we have listed multiple knobs, and the knob configuration space Θ is large, which means our tested algorithms are only a tiny proportion. The discovery of new configurations based on Algorithm 1 will enable us to combine various optimizations that have never been tried and tested. Such new configurations will form new algorithms that can be potentially fast for a certain group of clustering tasks.

Applying ML to Other Database Algorithm Selection Problems. In the database field, efficiency evaluations of existing algorithms have been an essential guide for users due to the diversity of tasks and datasets. Even though several clear insights are given for choosing the right decision in the end of evaluations, they essentially form an elementary and inaccurate decision tree for referring. Auto-tuning based on the evaluation logs and ML models can interpret the superiority of algorithms in a more accurate way than humans, and can directly predict the optimal algorithm. Hence, it will be interesting to apply the auto-tuning methodology presented in this paper to those evaluation studies that have made a good comparison but still manually selects algorithms.

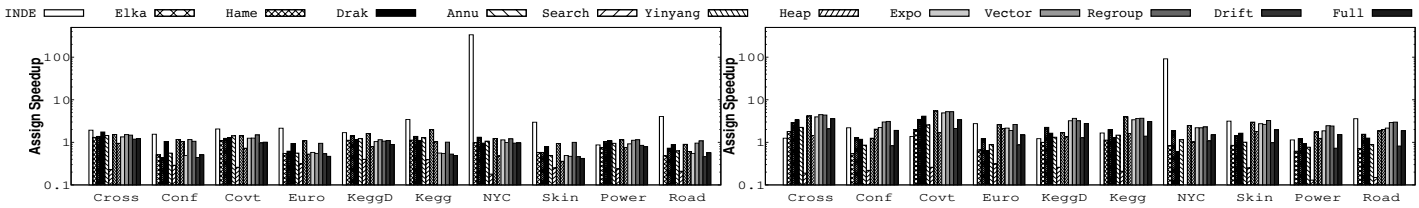

Figure 16: Overall assignment speedup in various datasets when setting k as 10 and 100, respectively.

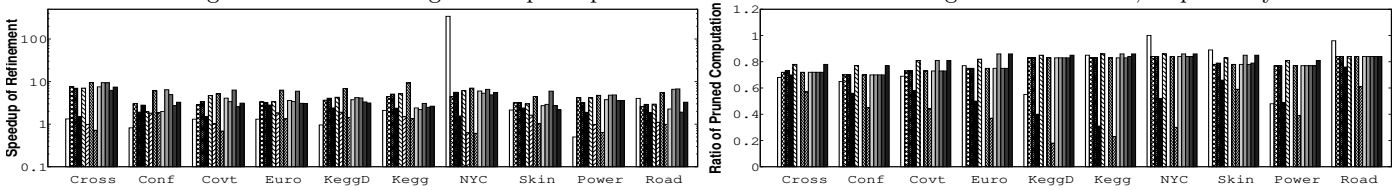

Figure 17: Refinement's speedup and distance calculation pruning ratio ($k = 10$).

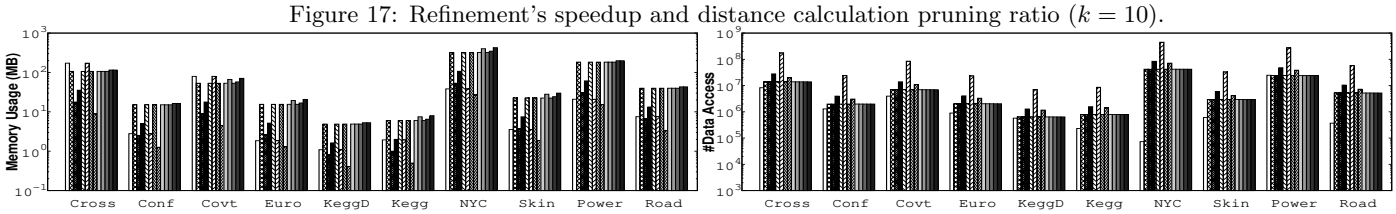

Figure 18: Statistics on the footprint of bound (index) and data accesses ($k = 10$).

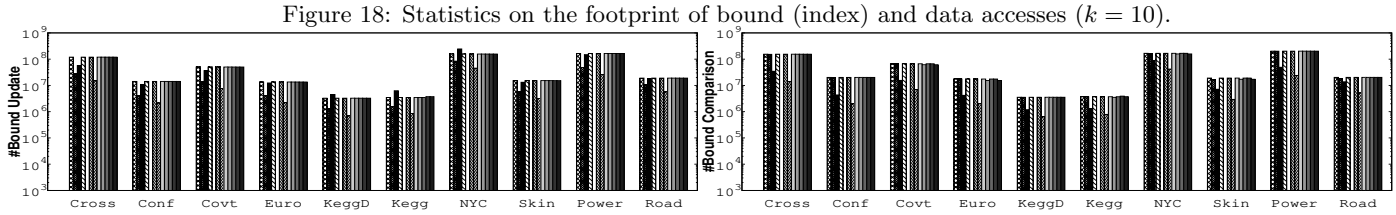

Figure 19: Statistics on the bound accesses and updates ($k = 10$).

Table 9: The assignment speedup over the running time (second) of Lloyd's algorithm (the gray column).

Data	$k = 10$					$k = 100$					$k = 1000$				
	Lloyd	×Speedup				Lloyd	×Speedup				Lloyd	×Speedup			
		SEQU	INDE	UniK	UTune		SEQU	INDE	UniK	UTune		SEQU	INDE	UniK	UTune
Cross	219.7	1.45	1.86	1.16	1.56	1381	2.73	2.20	3.10	4.54	13325	3.30	1.82	4.00	7.68
Conf	1.73	1.09	1.44	0.62	1.09	7.42	1.37	1.69	2	2.26	48.17	2.76	1.46	2.79	2.86
Covt	0.48	1.72	2.35	1.53	2.35	2.09	5.55	1.44	5.50	5.60	10.13	7.46	1.04	6.65	7.46
Euro	12.72	1.15	1.46	0.32	1.31	103.82	2.94	2.61	1.67	3.64	379.46	2.56	0.61	0.50	2.99
KeggD	0.35	2.57	4.11	2.69	3.74	1.04	2.47	1.21	2.00	5.52	8.29	6.56	1.23	2.06	7.51
Kegg	0.37	1.69	2.89	1.2	2.89	2.25	2.51	3.38	5.38	5.80	18.23	6.69	0.94	5.81	6.69
NYC	12.26	1.16	397.3	25.1	397.3	75.6	3.83	158.8	50.10	158.8	221.7	1.65	10.9	7.36	13.0
Skin	0.44	1.03	2.68	2.10	2.68	2.63	4.13	2.73	2.39	3.8	20.86	2.65	1.38	2.53	3.45
Power	5.31	1.26	0.90	0.77	1.26	29.79	2.26	1.06	2.38	2.46	220.19	2.15	0.97	2.24	2.4
Road	4.28	1.09	9.01	2.57	9.01	17.54	2.38	3.92	3.11	4.52	126.84	2.32	1.58	1.90	2.75

Table 10: The refinement speedup over the running time (second) of Lloyd's algorithm (the gray column).

Data	$k = 10$					$k = 100$					$k = 1000$				
	Lloyd	×Speedup				Lloyd	×Speedup				Lloyd	×Speedup			
		SEQU	INDE	UniK	UTune		SEQU	INDE	UniK	UTune		SEQU	INDE	UniK	UTune
Cross	42.6	8.03	1.34	5.56	8.74	83.0	7.34	1.68	12.8	16.6	205	4.77	1.26	11.3	18.8
Conf	0.72	6.60	6.60	3.54	6.60	1.58	1.37	1.27	11.5	10.1	2.58	7.59	1.55	11.1	8.40
Covt	0.17	7.46	1.39	7.54	1.39	0.44	1.17	1.17	12.76	8.90	0.25	8.13	0.97	9.85	8.13
Euro	2.62	7.90	1.25	2.69	3.19	6.79	11.41	2.08	3.49	16.9	2.44	6.87	1.65	2.48	15.3
KeggD	0.10	9.95	2.28	10.7	17.1	0.12	8.21	1.16	2.01	15.2	0.21	8.08	1.63	1.97	15.6
Kegg	0.12	10.2	2.57	6.76	2.57	0.25	15.8	1.78	34.6	22.0	0.41	5.15	0.83	15.4	5.15
NYC	3.06	6.87	361	348	361	8.76	14.2	122	162.7	122	8.05	4.61	15.3	21.1	24.2
Skin	0.12	6.79	2.19	8.90	2.19	0.29	8.81	1.88	9.70	15.2	0.55	8.10	1.34	8.71	13.4
Power	1.07	5.60	0.44	3.52	5.60	3.12	6.12	0.74	7.43	6.31	3.71	3.92	0.55	6.00	6.4
Road	1.74	5.88	7.65	13.0	7.65	3.63	6.58	2.50	8.76	16.2	5.96	7.48	1.61	15.2	15.3