kDCI: on using direct count up to the third iteration

Claudio Lucchese
HPC Lab of ISTI-CNR
Pisa, Italy.
claudio.lucchese@isti.cnr.it

Salvatore Orlando
Università Ca' Foscari
Venezia, Italy.
orlando@dsi.unive.it

Raffaele Perego
HPC Lab of ISTI-CNR
Pisa, Italy.
raffaere.perego@isti.cnr.it

1. Problem Statement and Solution

In Apriori-like algorithms, one of the most consuming operation during the frequent itemset mining process is the candidate search. At each iteration k the whole dataset $\mathcal{D}$ has to be scanned, and for each transaction t in the database, every of its subsets of length k is generated and searched within the candidates. If a candidate is matched, it means that the transaction subsumes the candidate, and therefore its support can be incremented by one.

This search is very time demanding even if appropriate data structures are used to gain a logarithmic cost. In [3, 2] we introduced a *direct count* technique which allows constant time searches for candidates of length 2. Given the set of n frequent single items, candidates of length 2 are stored using an upper triangular matrix $n \times n$ DC_2 with $\binom{n}{2}$ cells, such that $DC_2(i,j)$ stored the support of the 2-itemset $\{ij\}$. As shown in [1] the direct count procedure can be extended to the third iteration using an $n \times n \times n$ matrix DC_3 with $\binom{n}{3}$ cells, where $DC_3(i,j,l)$ is the support of the 3-itemset $\{ijl\}$.

We thus introduced such technique in the last version of kDCI, which is level-wise hybrid algorithm. kDCI stores the dataset with an horizontal format to disk during the first iterations. After some iteration the dataset may become small enough (thanks to anti-monotone frequency pruning) to be stored in the main memory in a vertical format, and after that the algorithm goes on performing tid-lists intersections to retrieve itemsets supports, and searches among candidates are not needed anymore. Usually the dataset happens to be small enough at most at the fourth iteration.

2. Experiments and Conclusion

The experiments show that the improvement given by this optimization is sensible in some cases. The time needed for the third iteration is halved. Note that the time spent during the firsts iteration is significant for the global effectiveness of the algorithm on most datasets. In fact, in the test performed, the total time was reduced from 19 sec. to 14 sec., which means an overall speed up of about 20%.

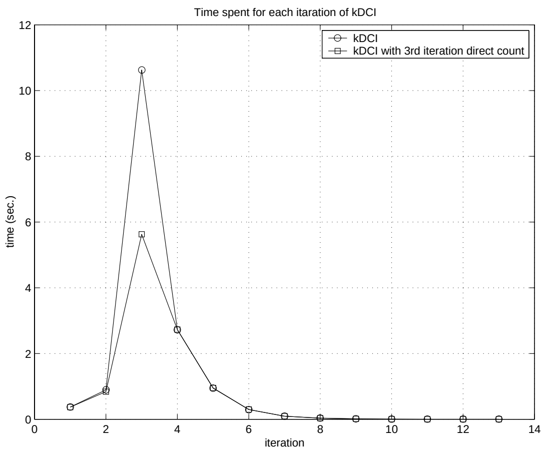

Figure 1. The dataset used was T10I4D100K with a minimum absolute support of 10.

We acknowledge the authors C.Targa, A.Prado and A.Plastino of [1], who showed the effectiveness of such optimization.

References

[1] C.Targa, A.Prado, and A.Plastino. Improving direct counting for frequent set mining. Technical report, Instituto de Computação, UFF RT-02/09 2003.

[2] Claudio Lucchese, Salvatore Orlando, Paolo Palmerini, Raffaele Perego, and Fabrizio Silvestri. kdci: a multistrategy algorithm for mining frequent sets. In *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, November 2003.

[3] S. Orlando, P. Palmerini, R. Perego, and F. Silvestri. Adaptive and resource-aware mining of frequent sets. In *Proc. The 2002 IEEE International Conference on Data Mining (ICDM02)*, page 338345, 2002.