Commentary on "CoverSize: a global constraint for frequency-based itemset mining"

Pierre Schaus, John O.R. Aoga, Tias Guns

October 8, 2019

This paper is a prime example of a 'best of both worlds' approach.

T.G. had been doing research on pattern mining with constraint programming during his PhD. Pattern mining, more specifically itemset mining, is concerned with extracting the set of all frequently occurring elements (item-sets) from a large collection of sets.

After the PhD, he was investigating whether pattern mining could also be of benefit to constraint programming, e.g. whether the patterns extracted could help in formulating or solving CP problems. One area that looked promising was table constraints, as the data representation of itemset mining was very similar to the Boolean attribute-value representation of a table.

While discussing these ideas informally, P.S. pointed out that they were already working on a brand new table constraint propagator, that operated directly on the Boolean matrix representation of the table constraints, and used bitvector operations inside. This was the 'Compact Table' propagator, for which they subsequently solved that it dominated all previous propagators in nearly all situations. It quickly became clear that this new propagator was in fact very similar to how some specialized itemset mining algorithms implemented the computation of the *cover* relation. However, Compact Table had an additional novelty that could also be of benefit to itemset mining algorithms: the backtrack-friendly reversible sparse bitset datastructure.

With this, we could build a global constraint that performed the cover computation as specialized algorithms do, as Lazaar et al had done recently, but more efficiently. The advantage of putting it in one big constraint is that there is no need to instantiate one Boolean variable per transaction in the solver. These decision variables are functionally defined by the items. Hence when there are no additional constraints on these transaction variables, it is more beneficial to hide them in the global constraint so that the solver does not need to manage them. As there can be thousands to millions of constraints, not having to manage thousands to millions of decision variables would allow scaling to bigger data.

Generality vs. efficiency However, the prime motivation for using constraint programming for itemset mining, rather than specialized algorithms, is the ability to add arbitrary constraints to the problem formulation. We hence wanted to find a more general solution

that still did not require exposing the thousands of transaction variables to the solver. When reviewing the typical constraints expressed on the transaction variables, we realized they basically fall into one of two constraints: 1) constraints on the sum of the transaction variables, that is, the *frequency* of the itemsets, and 2) closedness constraints that can be expressed as a relation between item and transaction variables.

With respect to 1) constraints on the sum of transaction variables, we came to the conclusion that all that was really needed was the value of the frequency. Hence, we could expose an integer variable whose upper and lower-bound was made consistent with the minimal and maximal frequency obtainable by the itemset during search. We called it *CoverSize*. However, propagating from a new upper bound on this variable to the item variables was non-trivial. In fact, our theoretical analysis showed that it is NP-hard to check consistency in general for such a propagator. Still, we found a number of conditions that could be propagated efficiently and these showed to be sufficient in practice.

For the second case, 2) closedness constraints between items and transactions we could not devise a way to export only part of the information. Closedness really requires reasoning both over the set of transactions and items in the data. We hence created a *CoverClosure* constraint that would be similar to *CoverSize* in terms of internal data structure, but that would propagate the closedness relation only. It could hence optionally be combined with CoverSize, or even only on part of the data as is often done when working with a positive and a negative set of transactions for discriminative itemset mining.

Our experiments showed that this approach significantly outperformed other CP-based pattern mining algorithms, and that it was on par with a specialized CP-like algorithm. We also openly showed that even this significant improvement in efficiency for CP-based methods did not allow it to compete with the highly optimized and specialized LCM algorithm. However, it was more generic in that it allowed for any constraints on items and frequency, including disciminative itemset mining settings. Hence, it combined the generality of CP-based itemset mining with the effectiveness of specialized algorithms and the efficiency of backtracking aware data-structures.

Past, present, future Looking back, these CoverSize and CoverClosure constraints could in fact be plugged into almost all of the prior work of T.G. on CP-based pattern mining, and would greatly benefit the scalability of the approaches when doing so.

Looking forward, it remains a clear and well-defined building block for CP-based itemset mining, and for example also used in this year's IJCAI19 paper on "Constraint Programming for Mining Borders of Frequent Itemsets" by Mohamed-Bachir Belaid, Christian Bessiere and Nadjib Lazaar and the CP19 paper on "Learning optimal decision trees using constraint programming" by Hélène Verhaeghe, Siegfried Nijssen, Gilles Pesant, Claude-Guy Quimper and Pierre Schaus.