# A Java Library for Itemset Mining with Choco-solver

Charles Vernerey [1][¶] and Samir Loudni [1]

1 TASC, IMT-Atlantique, LS2N-CNRS, Nantes, France ¶ Corresponding author

Summary

While traditional data mining techniques have been used extensively for discovering patterns in databases, they are not always suitable for incorporating user-specified constraints. To overcome this issue, new research has began connecting Data Mining to Constraint Programming (CP). Such fertilization leads to a flexible way to tackle data mining tasks, such as itemset or association rule mining. In this paper, we introduce a new library for solving itemset mining problems with Choco-solver.

Constraint Programming (CP)

Constraint Programming (CP) is a powerful paradigm for solving combinatorial optimization problems (Rossi et al., 2006). It provides a declarative approach to problem-solving by defining a set of variables, domains, and constraints that capture the problem's requirements. CP solvers explore the space of possible solutions systematically, leveraging powerful search algorithms and constraint propagation techniques to efficiently find valid solutions. The flexibility of CP allows for modeling a wide range of problems, including scheduling (Baptiste et al., 2001), resource allocation (Zhang et al., 2013), and planning (Van Beek & Chen, 1999). Its ability to handle complex constraints, discrete variables, and global properties makes it particularly suitable for tackling real-world problems. CP has demonstrated remarkable success in various domains, offering a high-level modeling language and a diverse set of solving techniques. Its integration with other optimization methods and technologies further enhances its applicability and effectiveness. Overall, Constraint Programming is a valuable tool for addressing challenging optimization problems, offering a powerful approach to problem modeling, solving, and decision support.

Itemset Mining

Itemset mining is a fundamental data mining technique that aims to extract meaningful associations and patterns from large datasets (Fournier-Viger et al., 2017). It involves the identification of sets of items (called itemsets or patterns) that frequently co-occur or exhibit significant relationships. By uncovering these itemsets, researchers gain valuable insights into the underlying structure and dependencies within the data. Itemset mining finds applications in various domains, including market basket analysis (Agrawal et al., 1994), bioinformatics (Martinez et al., 2008), and social network analysis (Erlandsson et al., 2016).

CP and Pattern Mining

In recent years, CP has been proven to be effective for modelling and solving itemset mining problems (Guns et al., 2011; Lazaar et al., 2016; Ugarte et al., 2017), and sequence mining problems (Ghosh et al., 2022; Kadıoğlu et al., 2023; Wang et al., 2022; Wang & Kadıoğlu, 2022). The main advantage of using CP rather than specialised approaches for solving pattern mining problems is that the user can easily add custom constraints without having to modify

the underlying system. Multiple user-specified constraints have been proposed in the literature to model and solve several pattern mining problems.

Statement of need

Having a generic prototypical approach that can be parameterized to declaratively and efficiently discover patterns of interest using the available constraint solving tools is crucial to promote the use of CP for itemset mining. Multiple constraints designed for different mining tasks have been proposed in the recent years. However, few alternatives exist that bring all of these constraints together in the same place. A user interested by using constraints in their own project would have to implement them from scratch, which takes time and may lead to bugs. To alleviate the burden of the user, we propose a new CP library that gathers multiple reference constraints for itemset mining in the same place.

Features and Functionality
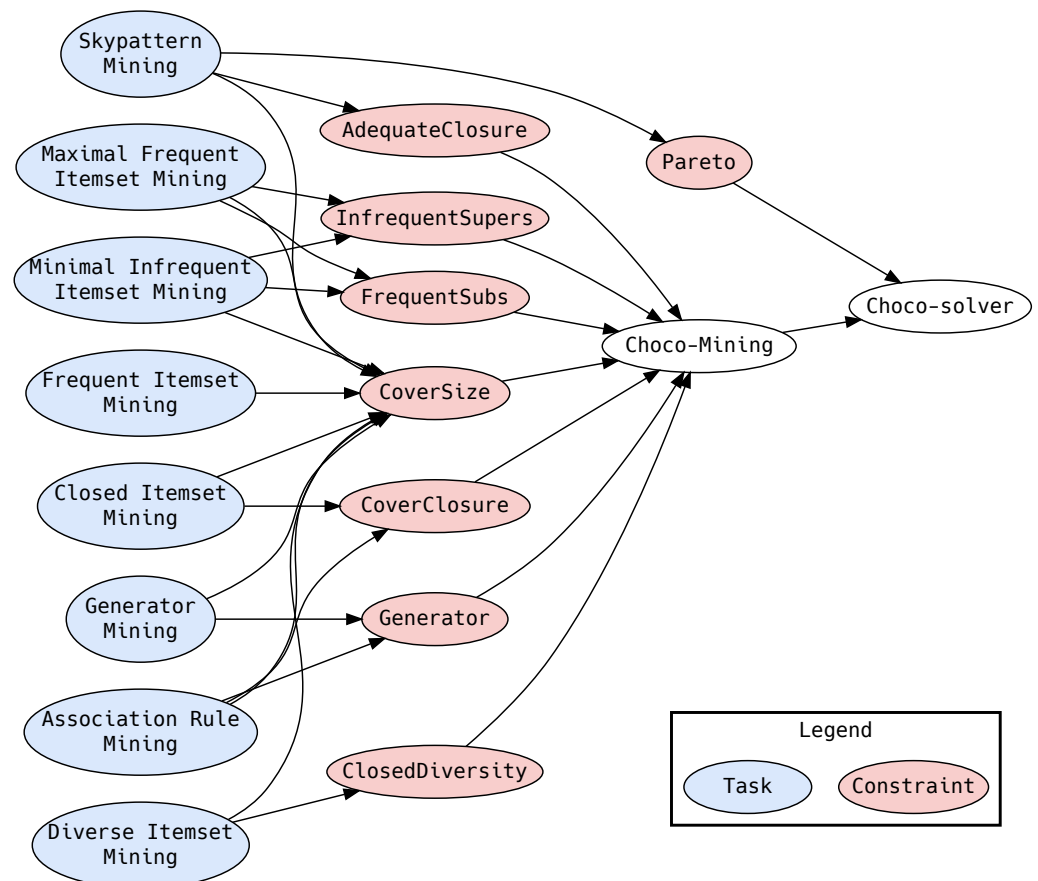
Figure 1: Summary of constraints implemented with Choco-Mining

We propose a new CP library called **Choco-Mining** that is based on Choco-solver (Prud'homme & Fages, 2022). The architecture of the library is illustrated in Figure 1. As we can see, multiple constraints dedicated to different itemset mining tasks are available in the Choco-Mining library. Each constraint takes as input a transactional database D and a vector of Boolean variables x used for representing itemsets, where $x[i]$ represents the presence/absence of the item i in the

searched itemset. These constraints are then used to define the problem at hand in terms of constraint programming. The following constraints are available in Choco-Mining:

- $CoverSize_D(x, f)$ (Schaus et al., 2017): Given an integer variable f that represents the frequency (noted $freq$) of an itemset x, the constraint ensures that $f = freq(x)$.
- $CoverClosure_D(x)$ (Schaus et al., 2017): The constraint ensures that x is closed w.r.t. the frequency, i.e., $\nexists\, y \supset x : freq(x) = freq(y)$.
- $AdequateClosure_{D,M}(x)$ (Vernerey et al., 2022): Given a set of measures M, the constraint ensures that x is closed w.r.t. M, i.e., $\nexists\, y \supset x$ such that for all $m \in M :$ $m(x) = m(y)$.
- $FrequentSubs_{D,s}(x)$ (M. Belaid et al., 2019): Given a frequency threshold s, the constraint ensures that all the subsets of x are frequent, i.e., $\forall y \subset x : freq(y) \geq s$.
- $InfrequentSupers_{D,s}(x)$ (M. Belaid et al., 2019): Given a frequency threshold s, the constraint ensures that all the supersets of x are infrequent, i.e., $\forall y \supset x : freq(y) < s$.
- $Generator_D(x)$ (M.-B. Belaid et al., 2019): The constraint ensures that x is a generator, i.e., $\nexists\, y \subset x : freq(y) = freq(x)$.
- $ClosedDiversity_{D,\mathcal{H},j,s}(x)$ (Hien et al., 2020): Given a history of itemsets $\mathcal{H}$, a diversity threshold j and a minimum frequency threshold s, the constraint ensures that x is a diverse itemset (i.e., $\nexists\, y \in \mathcal{H} : jaccard(x, y) \geq j$), x is closed w.r.t. the frequency and $freq(x) \geq s$.

We can model different problems using these constraints. Figure 1 shows examples of mining tasks (in blue) with the constraints (in red) involved in their modelling:

- Frequent Itemset Mining: Given a threshold s, find all the itemsets x such that $freq(x) \geq s$.
- Closed Itemset Mining: Given a threshold s, find all the itemsets x such that $freq(x) \geq s$ and $\nexists\, y \supset x : freq(x) = freq(y)$.
- Skypattern Mining: Given a set of measures M, find all the itemsets x such that there exists no other itemset y that dominates x. We say that y dominates x iff $\forall\, m \in M : m(y) \geq m(x)$ and $\exists\, m \in M : m(y) > m(x)$.
- Maximal Frequent Itemset Mining: Given a threshold s, find all the itemsets x such that $freq(x) \geq s$ and $\forall\, y \supset x : freq(y) < s$.
- Minimal Infrequent Itemset Mining: Given a threshold s, find all the itemsets x such that $freq(x) < s$ and $\forall\, y \subset x : freq(y) \geq s$.
- Generator Mining: Find all the itemsets x such that $\nexists\, y \subset x : freq(y) = freq(x)$.
- Association Rule Mining: Find all the association rules $x \Rightarrow y$ that respect the constraints specified by the user.
- Diverse Itemset Mining: Given a diversity threshold j and a minimum frequency threshold s, find all the diverse itemsets that are closed w.r.t. the frequency and such that $freq(x) \geq s$.

Running example

We give below an example of CP encoding for the Closed Itemset Mining problem using our library Choco-Mining.

```java
// Read the transactional database
TransactionalDatabase database = new DatReader("data/contextPasquier99.dat").read();
// Create the Choco model
Model model = new Model("Closed Itemset Mining");
/* Array of Boolean variables where x[i] == 1 represents
the fact that i belongs to the itemset */
BoolVar[] x = model.boolVarArray("x", database.getNbItems());
/* Integer variable that represents the frequency of x
with the bounds [1, nbTransactions] */
```

```java
IntVar freq = model.intVar("freq", 1, database.getNbTransactions());
// Integer variable that represents the length of x with the bounds [1, nbItems]
IntVar length = model.intVar("length", 1, database.getNbItems());
// Ensures that length = sum(x)
model.sum(x, "=", length).post();
// Ensures that freq = frequency(x)
ConstraintFactory.coverSize(database, freq, x).post();
// Ensures that x is a closed itemset
ConstraintFactory.coverClosure(database, x).post();
Solver solver = model.getSolver();
// Variable heuristic : select item i such that freq(x U i) is minimal
// Value heuristic : instantiate it first to 0
solver.setSearch(Search.intVarSearch(
        new MinCov(model, database),
        new IntDomainMin(),
        x
));
// Create a list to store all the closed itemsets
List<Pattern> closedPatterns = new LinkedList<>();
while (solver.solve()) {
    int[] itemset = IntStream.range(0, x.length)
            .filter(i -> x[i].getValue() == 1)
            .map(i -> database.getItems()[i])
            .toArray();
    // Add the closed itemset with its frequency to the list
    closedPatterns.add(new Pattern(itemset, new int[]{freq.getValue()}));
}
System.out.println("List of closed itemsets for the dataset contextPasquier99:");
// Print all the closed itemsets with their frequency
for (Pattern closed : closedPatterns) {
    System.out.println(Arrays.toString(closed.getItems()) +
            ", freq=" + closed.getMeasures()[0]);
}
```

The goal is to find all the closed itemsets with a minimum frequency of 1. We start by reading the transactional database using the method `read()` of the `DatReader` instance. Then, we create a model with Choco-solver. Variables `freq` and `length` are created to store respectively the frequency and the length of the itemset. A boolean array of variables `x` represents the itemset, where `x[i] = 1` indicates that item `i` belongs to the itemset. Finally, we post three constraints:

- `model.sum(x, "=", length).post()`: ensures that $length = \sum x$.
- `ConstraintFactory.coverSize(database, freq, x).post()`: ensures that $freq = freq(x)$.
- `ConstraintFactory.coverClosure(database, x).post()`: ensures that x is closed w.r.t. the frequency.

After finding all the solutions, we print them to the user.

Acknowledgements

References

Agrawal, R., Srikant, R., & others. (1994). Fast algorithms for mining association rules. *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, *1215*, 487–499.

Baptiste, P., Le Pape, C., & Nuijten, W. (2001). *Constraint-based scheduling: Applying constraint programming to scheduling problems* (Vol. 39). Springer Science & Business Media.

Belaid, M.-B., Bessiere, C., & Lazaar, N. (2019). Constraint programming for association rules. *Proceedings of the 2019 SIAM International Conference on Data Mining (SDM)*, 127–135. https://doi.org/10.1137/1.9781611975673.15

Belaid, M., Bessiere, C., & Lazaar, N. (2019). Constraint programming for mining borders of frequent itemsets. *Proceedings of IJCAI 2019*, 1064–1070. https://doi.org/10.24963/ijcai.2019/149

Erlandsson, F., Bródka, P., Borg, A., & Johnson, H. (2016). Finding influential users in social media using association rule learning. *Entropy*, *18*(5), 164. https://doi.org/10.3390/e18050164

Fournier-Viger, P., Lin, J. C.-W., Vo, B., Chi, T. T., Zhang, J., & Le, H. B. (2017). A survey of itemset mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *7*(4), e1207. https://doi.org/10.1002/widm.1207

Ghosh, S., Yadav, S., Wang, X., Chakrabarty, B., & Kadıoğlu, S. (2022). Dichotomic pattern mining integrated with constraint reasoning for digital behavior analysis. *Frontiers in Artificial Intelligence*, *5*, 868085. https://doi.org/10.3389/frai.2022.868085

Guns, T., Nijssen, S., & De Raedt, L. (2011). Itemset mining: A constraint programming perspective. *Artificial Intelligence*, *175*(12), 1951–1983. https://doi.org/10.1016/j.artint.2011.05.002

Hien, A., Loudni, S., Aribi, N., Lebbah, Y., Laghzaoui, M., Ouali, A., & Zimmermann, A. (2020). A relaxation-based approach for mining diverse closed patterns. *Proceedings of ECML PKDD 2020*, *12457*, 36–54. https://doi.org/10.1007/978-3-030-67658-2_3

Kadıoğlu, S., Wang, X., Hosseininasab, A., & Hoeve, W.-J. van. (2023). Seq2Pat: Sequence-to-pattern generation to bridge pattern mining with machine learning. *AI Magazine*, *44*(1), 54–66. https://doi.org/10.1002/aaai.12081

Lazaar, N., Lebbah, Y., Loudni, S., Maamar, M., Lemière, V., Bessiere, C., & Boizumault, P. (2016). A global constraint for closed frequent pattern mining. *Principles and Practice of Constraint Programming: 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings 22*, 333–349. https://doi.org/10.1007/978-3-319-44953-1_22

Martinez, R., Pasquier, N., & Pasquier, C. (2008). GenMiner: Mining non-redundant association rules from integrated gene expression data and annotations. *Bioinformatics*, *24*(22), 2643–2644. https://doi.org/10.1093/bioinformatics/btn490

Prud'homme, C., & Fages, J.-G. (2022). Choco-solver: A Java library for constraint programming. *Journal of Open Source Software*, *7*(78), 4708. https://doi.org/10.21105/joss.04708

Rossi, F., Van Beek, P., & Walsh, T. (2006). *Handbook of constraint programming*. Elsevier.

Schaus, P., Aoga, J. O. R., & Guns, T. (2017). CoverSize: A global constraint for frequency-based itemset mining. *Proceedings of the 23rd CP 2017*, 529–546. https://doi.org/10.1007/978-3-319-66158-2_34

Ugarte, W., Boizumault, P., Crémilleux, B., Lepailleur, A., Loudni, S., Plantevit, M., Raïssi, C., & Soulet, A. (2017). Skypattern mining: From pattern condensed representations

to dynamic constraint satisfaction problems. *Artificial Intelligence*, *244*, 48–69. https://doi.org/10.1016/j.artint.2015.04.003

Van Beek, P., & Chen, X. (1999). CPlan: A constraint programming approach to planning. *AAAI/IAAI*, 585–590.

Vernerey, C., Loudni, S., Aribi, N., & Lebbah, Y. (2022). Threshold-free pattern mining meets multi-objective optimization: Application to association rules. *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 1880–1886. https://doi.org/10.24963/ijcai.2022/261

Wang, X., Hosseininasab, A., Colunga, P., Kadıoğlu, S., & Hoeve, W.-J. van. (2022). Seq2Pat: Sequence-to-pattern generation for constraint-based sequential pattern mining. *Proceedings of the AAAI Conference on Artificial Intelligence*, *36*(11), 12665–12671. https://doi.org/10.1609/aaai.v36i11.21542

Wang, X., & Kadıoğlu, S. (2022). Dichotomic pattern mining with applications to intent prediction from semi-structured clickstream datasets. *The AAAI-22 Workshop on Knowledge Discovery from Unstructured Data in Financial Services*. https://arxiv.org/abs/2201.09178

Zhang, L., Zhuang, Y., & Zhu, W. (2013). Constraint programming based virtual cloud resources allocation model. *International Journal of Hybrid Information Technology*, *6*(6), 333–344. https://doi.org/10.14257/ijhit.2013.6.6.30