Constraint Programming & Data Mining using Choco-solver

1 Constraint Programming Using Choco-solver

In this section, we will delve into a new programming paradigm—declarative programming, specifically *Constraint Programming (CP)*, employing the Java library Choco-solver. Choco-solver is an open-source Java library tailored for CP.

In your local repository, you will find :
— `"HelloWorld.java"` : A "Hello World" program using `Choco Solver`.
— `"Sudoku.java"` : A CP modeling of the `sudoku`$(n \times n)$ problem.
— `"HardSudoku.java"` : An instance of Sudoku to be modeled.
— `"VeryHardSudoku.java"` : Another instance of Sudoku to be modeled.
— `"GTSudoku.java"` : A variant of the problem to be modeled.

Question 1 • Fork the GitHub repository using the following Classroom link : LINK

Question 2 • Import your local repository using an IDE of your choice.

Question 3 • Execute the main method of the `"Sudoku.java"` class.

Question 4 • Take the time to understand the code of the `"Sudoku.java"` class and model/complete the code of the `"HardSudoku.java"`, `"VeryHardSudoku.java"`, and `"GTSudoku.java"` classes. Figures 1, 2, and 3 represent respectively the "Hard," "VeryHard," and "GT" instances of Sudoku.

8								
		3	6					
	7			9		2		
	5				7			
				4	5	7		
			1				3	
		1					6	8
		8	5				1	
	9					4		

Figure 1 – Hard Sudoku instance.

	G			F	8	9	6	4	B	D	5			3	
6	C					4	E	2	7					5	9
			D			G	7	F	E			6			
		4	3	A							6	1	B		
7			5	8	F					B	E	9			G
8				9			4	D			3				2
C	1	3				6			G				F	4	5
9	D	B			G					F			7	A	6
G	B	A			2					7			5	6	D
5	6	F				A			2				8	7	4
D				6			9	5			G				F
3			C	B	5					A	4	G			1
		9	6	G							7	2	C		
			G			B	D	C	5			F			
4	3					8	2	G	F					1	7
	8			5	9	E	A	1	3	2	D			G	

FIGURE 2 – Very Hard Sudoku instance.

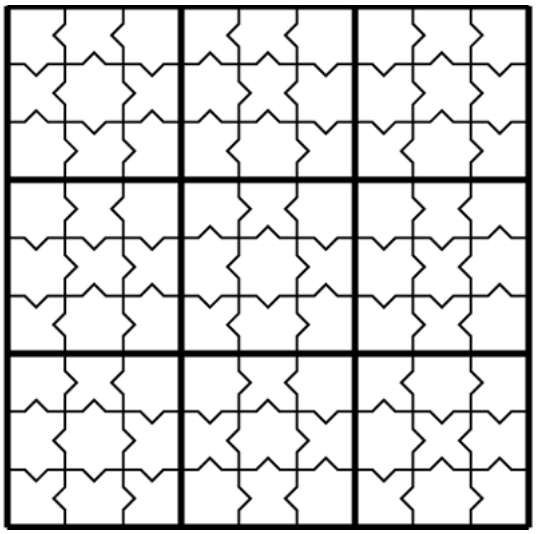

FIGURE 3 – Greater-Than Sudoku instance.

Question 5 • We will now address the optimization aspect by dealing with a variant of Sudoku with a minimizable objective function. Create a file named `SudokuOPT.java` that returns the solution with the smallest sum of values in the grid's main diagonal.

2 Declarative Itemset Mining Using `Choco-mining` module

In this section, we will be working with :

— Choco-Mining is a Java library designed for solving itemset mining problems, built on the Choco-solver framework.

— The SPMF library, an open-source Java-based software and data mining library, specializing in pattern mining (SPMF).

Question 6 • Clone the GitHub repository of Choco-Mining (link).

Question 7 • Open the file `ExampleClosedItemsetMining.java` and perform the following tasks :

1. Review the code in detail.
2. Run the main.
3. Run it on other datasets such as *mushroom* or *chess*.
4. Display the number of resulting patterns.
5. Display the execution time.

Question 8 • Add the frequency constraint : $freq(P) \geq \theta$.

Question 9 • Add a constraint on the size of the returned patterns : $size(P) \geq \alpha$.

Question 10 • Now we are going to replicate the tasks using SPMF. Just run the `.jar` file available in your local repository. The goal is to run LCM for closed itemset enumeration, relaunch with different thresholds for frequency, and also for pattern size.

Question 11 • We will now add a constraint, which we will call `CategoryConstraint` here, to our file 'ExampleClosedItemsetMining.java' that models the following problem : Let's consider a dataset with n items. The dataset contains categories of size $catSize$ (e.g., household products, appliances, etc.). The dataset is organized into $nbCat = n/catSize$ categories, with items at the end that do not belong to any category but do not exceed the number of $catSize$. Figure 4 provides an example of a dataset with 8 items, 2 categories of size 3, and 2 items at the end that are out of category but do not exceed the category size. The question now is to have a constraint model that allows enumerating all closed itemsets composed of items belonging to atleast m categories :

$$\texttt{CategoryConstraint(P)} \equiv \sum_{\texttt{i=1}}^{\texttt{nbCat}} \prod_{\texttt{j=1}}^{\texttt{catSize}} \texttt{P}_\texttt{i} \geq \texttt{m}$$

For example, the dataset in Figure 4 yields the following patterns with $m = 2$: BEF.

Question 12 • How can this `CategoryConstraint` be taken into account in SPMF ?

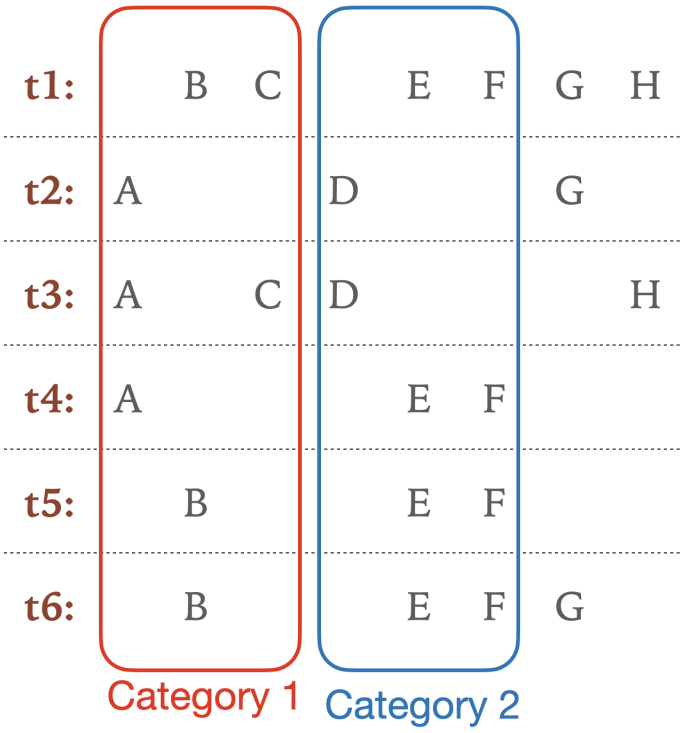

FIGURE 4 – Items categories illustration.