

---

**TP 02 – Use a SAT solver, approximation**


---

Programming language of your choice (to be validated with your TP instructor).

**Exercise 1.***Sudoku*

Sudoku is a game where the digits from 1 to 9 must be placed in a grid of 9 rows and 9 columns, divided in 9 blocks of  $3 \times 3$  size. The digits 1 to 9 must be placed so that each row, each column and each  $(3 \times 3)$ -block contains all the digits 1 to 9. Given some digits initially placed on the grid, the Sudoku decision problem is to determine whether the grid can be completed or not respecting the rules above.

1. Give an example of a negative instance of the Sudoku decision problem.
2. In asymptotic notation (given a simplified expression as much as possible), what is the complexity of an algorithm for solving Sudoku on a grid of size  $9 \times 9$ ?

We now generalise Sudoku to a grid of size  $n^2 \times n^2$ , which is divided into  $n^2$  blocks of size  $n \times n$ . The game then uses  $n^2$  different symbols to fill the grid's cells.

3. Explain how to model an instance of the generalised Sudoku problem of size  $n^2 \times n^2$  as a propositional formula that is satisfiable if and only if the grid has a solution. Compute its number of variables and clauses as a function of  $n$ .
4. Implement a reduction to SAT of the question 3, for size  $n = 3$  (classic Sudoku), following the structure below :
  - (a) The program will take as input a  $9 \times 9$  grid as a string of 81 characters on the alphabet  $\{1, 2, 3, 4, 5, 6, 7, 8, 9, .\}$ , where  $.$  means an initially empty cell (the rows of the grid are concatenated, see the benchmarks in question 5), and it will produce a formula in DIMACS format (<https://people.sc.fsu.edu/~jburkardt/data/cnf/cnf.html>).
  - (b) The program will call minisat (<http://minisat.se/>) to decide satisfiability, and obtain a model if possible.
  - (c) The program must include a display functionality (in the terminal, it is sufficient) of the starting grid and its solution (if it exists).
5. Experimentally measure the number of grids that your procedure can resolve in 1 second, for each of the following two grid datasets :
  - [https://pageperso.lis-lab.fr/kevin.perrot/benchmarks/benchmark\\_sudoku\\_1.txt](https://pageperso.lis-lab.fr/kevin.perrot/benchmarks/benchmark_sudoku_1.txt)
  - [https://pageperso.lis-lab.fr/kevin.perrot/benchmarks/benchmark\\_sudoku\\_2.txt](https://pageperso.lis-lab.fr/kevin.perrot/benchmarks/benchmark_sudoku_2.txt)
 Source : <https://github.com/t-dillon/tdoku/benchmarks> 17\_clue et forum\_hardest\_1905\_11+.
6. Explain how to model the following additional constraint for  $n^2 \times n^2$  size :  
two horizontally adjacent boxes must not contain consecutive digits.  
Add a function to your code that produces these clauses.

**Exercise 2.***Timetable*

We will model and solve a timetable construction problem that is compatible with the constraints of students and teachers. Let  $n$  be students,  $m$  teachers, and  $\ell$  courses. Each student follows a set of courses, and each teacher gives a set of courses. On the weekly timetable, we have 10 half-days available to schedule the classes. A student cannot follow two courses simultaneously, and a teacher cannot give two courses simultaneously. Given the courses followed by each student, and the courses given by each teacher, is it possible to design a compatible timetable?

1. Give a positive instance and a negative instance of the problem.
2. Write the code to randomly generate an instance of the problem.

For a given natural integer  $k$ , the  **$k$ -Coloring** problem consists of deciding, given an undirected graph  $G = (V, E)$ , whether it is possible to assign to each of its vertices a colour from the set  $\{1, 2, \dots, k\}$  such that if two vertices are connected by an edge, then they have different colours. Mathematically,  $\exists c : V \rightarrow \{1, 2, \dots, k\}, \forall \{u, v\} \in E, c(u) \neq c(v)$ ?

3. How can timetable construction be modelled as a graph colouring problem? (Which graph? How many colours?)
4. Given that the **3-Coloring** problem is NP-complete, prove that the timetable problem is NP-complete.
5. Explain how to reduce **10-Coloring** to **SAT** in polynomial time. That is to say, given a graph  $G$ , how to construct a formula  $\varphi_G$  that is satisfiable if and only if the graph  $G$  is  $k$ -colorable.
  - (a) Which are the variables?
  - (b) Which is the conjunction of constraints (clauses)?
  - (c) Express in CNF.

State the number of variables, as well as the number and size of the clauses, as a function of  $n = |V|$  and  $m = |E|$ . Analyse the computational time of the transformation.

6. Write the code for this reduction from a graph to a formula in DIMACS format.
7. Connect your scheduling problem instance generator to the `minisat` solver, via your reduction from  **$k$ -Coloring** to **SAT**. Produce a solution (if possible).

**Exercise 3.***Wireless network*

A wireless network consists of  $n$  antennas, some of which are connected to each other and others not. The connections are represented by an undirected graph  $G = (V, E)$  with  $|V| = n$  vertices (one vertex per antenna). To perform routing, an administrator wants to build a *network backbone*. This is a subset of antennas connected to all the antennas.

The aim is to minimise the number of antennas required. The associated decision problem is called **Dominating Set**: given a graph  $G = (V, E)$  and an integer  $k$ , does there exist a subset of vertices  $X \subseteq V$  of size  $k$  such that for every  $v \in V$  we have  $(\{v\} \cup \{u \mid \{u, v\} \in E\}) \cap X \neq \emptyset$ ?

1. Show that this problem belongs to the class NP.
2. Is the size of the dominating set difficult to maximise or minimise?

Recall that an independent set (or stable set) is a subset of vertices that are not connected to one another.

3. Show that every maximal independent set is also a minimal dominating set.
4. Knowing that the **Independent Set** problem (given  $G$  et  $k$ , does  $G$  have an independent set of size  $k$ ) is NP-hard, can we deduce that **Dominating Set** is NP-hard?

5. Use `minisat` to solve this problem (reduction to SAT).
  - (a) Which are the variables?
  - (b) Which is the conjunction of constraints (clauses)?
  - (c) Express in CNF.

We will now approximate the problem by trying to find a dominating set whose size is close to the minimum.

6. Implement a greedy algorithm to find a dominating set, where, at each step, it selects the vertex that dominates the greatest number of new vertices.
7. Analyse the complexity of the algorithm in question 6.

We will now prove that the greedy algorithm is a  $(\log \Delta + 2)$ -approximation, where  $\Delta$  is the maximum degree of the graph.

8. What does it mean to be a  $(\log \Delta + 2)$ -approximation?

Let  $G$  be an arbitrary graph of maximum degree  $\Delta$ . Let  $S^*$  be an optimal independent set, and  $S$  the independent set given by the greedy algorithm. Then  $|S^*| \leq |S|$ .

Let  $W = S \setminus S^*$ . Since  $S^*$  is a dominating set, every vertex  $w \in W$  has a neighbour in  $S^*$ , denoted by  $s(w)$ . The set of vertices  $w \in W$  and their neighbours  $s(w)$  in  $S^*$  form a collection of stars. We will analyse these stars.

9. In each star, how many vertices belong to  $S^*$ ?

We will now perform, for each star, an amortised calculation of the number of vertices in  $S$ , which will show that there are (overall) at most  $\log \Delta + 2$  vertices per star.

10. Why will this indeed allow us to reach a conclusion?

Let us consider the execution of the greedy algorithm. When a vertex  $u$  is selected for  $S$  by the greedy algorithm, we will distribute its cost (from 1 to the size of  $S$ ) uniformly among the new vertices dominated by the addition of  $u$  to  $S$ .

11. What are the minimum and maximum costs that a vertex can receive?

We will now focus on a single star, denoting its centre by  $s^*$  and its degree by  $d(s^*)$ .

12. Show that when a first vertex of this star is dominated by a vertex  $u$  added to  $S$  by the greedy algorithm, it receives a cost of at most  $\frac{1}{d(s^*)+1}$ .
13. Show that when a second vertex of this star is dominated by a vertex  $u$  added to  $S$  by the greedy algorithm, it receives a cost of at most  $\frac{1}{d(s^*)}$ .
14. Show that when a third vertex of this star is dominated by a vertex  $u$  added to  $S$  by the greedy algorithm, it receives a cost of at most  $\frac{1}{d(s^*)-1}$ .
15. Deduce a bound on the total cost of the star centred at  $s^*$ .

We denote by  $H(n) = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$  the  $n$ -th harmonic number.

16. By grouping the terms in sets of 1, 2, 4, 8, 16... show that  $H(2^m) \leq m + 1$ .
17. Conclude.