Exercice B003 *Problèmes `SAT` et `MAX-SAT`*

Cet énoncé est accompagné d'un ou plusieurs codes compagnons en OCaml fournissant certaines des fonctions mentionnées dans l'énoncé : ils sont à compléter en y implémentant les fonctions demandées.

On s'intéresse au problème SAT pour une formule en forme normale conjonctive. On se fixe un ensemble fini $\mathcal{V} = \{v_0, v_1, \ldots, v_{n-1}\}$ de variables propositionnelles.

Un littéral l_i est une variable propositionnelle v_i ou la négation d'une variable propositionnelle $\neg v_i$. On représente un littéral en OCaml par un type énuméré : le littéral v_i est représenté par `V i`, et le littéral $\neg v_i$ par `NV i`. Une *clause* $c = l_0 \vee l_1 \vee \cdots \vee l_{|c|-1}$ est une disjonction de littéraux, que l'on représente en OCaml par un tableau de littéraux. On ne considérera dans cet exercice que des formules sous forme normale conjonctive, c'est-à-dire sous forme de conjonctions de clauses $c_0 \wedge c_1 \wedge \cdots \wedge c_{m-1}$. On représente une telle formule en OCaml par une liste de clauses, soit une liste de tableaux de littéraux. On n'impose rien sur les clauses : une clause peut être vide, et un même littéral peut s'y trouver plusieurs fois. De même, une formule peut n'être formée d'aucune clause, elle est alors notée $\top$ et est considérée comme une tautologie. Une *valuation* $v : \mathcal{V} \to \{V, F\}$ est représentée en OCaml par un tableau de booléens.

Un programme OCaml à compléter vous est fourni. La fonction `initialise : int -> valuation` permet d'initialiser une valuation aléatoire.

1. Implémenter la fonction `evalue : clause -> valuation -> bool` qui vérifie si une clause est satisfaite par une valuation.

Etant donnée une formule f constituée de m clauses $c_0 \wedge c_1 \wedge \ldots c_{m-1}$ définies sur un ensemble de n variables, la fonction `random_sat` a pour objectif de trouver une valuation qui satisfait la formule, s'il en existe une et qu'elle y arrive. Cette fonction doit effectuer au plus k tentatives et renvoyer un résultat de type `valuation option` avec une valuation qui satisfait la formule passée en paramètre si elle en toruve une, et la valeur `None` sinon.

L'idée de ce programme est d'effectuer une assignation aléatoire des variables, puis de vérifier que chaque clause est satisfaite. Si une clause n'est pas satisfaite, on modifie aléatoirement la valeur associée à un littéral de cette clause, qui devient ainsi satisfaite, puis on recommence.

2. Si ce programme renvoie `None`, peut-on conclure quela formule f donnée en entrée n'est pas satisfiable ? De quel type d'algorithme probabiliste s'agit-il ?
3. Proposer un jeu de 5 tests élémentaires permettant de tester la correction de ce programme.
4. Ce programme est-il correct par rapport à sa spécification ? Si cela s'avère nécessaire, corriger son code pour qu'il remplisse ses objectifs.

On s'intéresse maintenant au problème MAX-SAT, qui consiste, toujours pour une frmule en forme normale conjonctive comme ci-dessus, à trouver le plus grand nombre de clauses de cette formule simultanément satisfiable. Un algorithme d'approximation probabiliste et naïf pour obtenir une solution approchée consiste à générer aléatoirement k valuations et retenir celle qui maximise le nombre de clauses satisfaites.

5. Implémenter cette approche en OCaml et vérifier sur quelques exemples. Quelle est sa complexité dans le meilleur et le pire cas ?
6. Sous l'hypothèse P $\neq$ NP, peut-il exister un algorithme de complexité polynomiale pour résoudre MAX-SAT ? Justifier.