TP Résolution 2-SAT

Mehdi ZNATA
Ahmed LETAIEF
Fadi NADER

Décembre 2024

Professeur de TD : Mr. Alexandre Dupont Bouillard
Responsable de la matière : Mr. Vincent Limouzy

Contents

1 Question 1 : Détermination du nombre maximal de clauses dans une formule 2-SAT

Introduction

Dans cet exercice, nous cherchons à déterminer combien de clauses distinctes une formule 2-SAT peut contenir lorsqu'elle est définie sur un ensemble de n variables. Une clause 2-SAT est une disjonction de deux littéraux, qui peuvent être positifs (x_i) ou négatifs ($\neg x_i$).

Analyse du problème

Pour résoudre cette question, nous considérons deux ensembles :

- V : Ensemble des littéraux positifs (x_i).
- F : Ensemble des littéraux négatifs ($\neg x_i$).

Nous explorons toutes les combinaisons possibles entre ces ensembles pour construire des clauses.

1. Combinaisons entre les valeurs de V

Chaque littéral de V peut être combiné avec un autre littéral de V (pas lui-même). Le nombre de combinaisons est donné par la formule suivante :

$$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

Ces combinaisons correspondent à des clauses de la forme $(x_i \vee x_j)$.

2. Combinaisons entre les valeurs de V et F

Chaque littéral de V peut être combiné avec un littéral différent de F (pas sa propre négation). Le nombre de combinaisons est également donné par :

$$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

Ces combinaisons correspondent à des clauses de la forme $(x_i \vee \neg x_j)$.

3. Combinaisons entre les valeurs de F et V

Enfin, chaque littéral de F peut être combiné avec un littéral différent de V. Le nombre de combinaisons est encore donné par :

$$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

Ces combinaisons correspondent à des clauses de la forme $(\neg x_i \vee x_j)$.

Résultat

En additionnant ces trois catégories de combinaisons, on obtient le nombre total de clauses possibles :

$$\text{Nombre total de clauses} = 3 \times \sum_{i=1}^{n-1} i = 3 \times \frac{n(n-1)}{2} = \frac{3n(n-1)}{2}$$

Conclusion

Le nombre maximal de clauses dans une formule 2-SAT avec n variables est :

$$\frac{3n(n-1)}{2}$$

2 Question 2 : Exemple d'une formule 2-SAT non satisfiable

Introduction

Dans cet exercice, nous devons fournir un exemple de formule 2-SAT qui n'est pas satisfiable. Une formule 2-SAT est une conjonction de clauses, où chaque clause est une disjonction contenant exactement deux littéraux. Une formule est dite non satisfiable s'il n'existe aucune assignation des variables qui permette d'évaluer toutes les clauses à vrai.

Exemple de formule non satisfiable

Considérons la formule suivante F :

$$F = (b \vee c) \wedge (\neg b \vee c) \wedge (b \vee \neg c) \wedge (\neg b \vee \neg c)$$

Analyse de l'exemple

La formule F est composée des clauses suivantes :

- $(b \vee c)$: Au moins une des variables b ou c doit être vraie.
- $(\neg b \vee c)$: Si b est faux, alors c doit être vrai.
- $(b \vee \neg c)$: Si c est faux, alors b doit être vrai.
- $(\neg b \vee \neg c)$: Si b est vrai, alors c doit être faux, et vice versa.

Vérification de la satisfiabilité

Analysons si une assignation des variables b et c peut satisfaire toutes les clauses :

- **Cas 1 : $b =$ vrai**
  - $(b \vee c)$: Vrai (car $b =$ vrai).
  - $(\neg b \vee c)$: Faux (car $\neg b =$ faux et c peut être faux).
  - $(b \vee \neg c)$: Vrai (car $b =$ vrai).
  - $(\neg b \vee \neg c)$: Faux (car $\neg b =$ faux).
- **Cas 2 : $b =$ faux**
  - $(\neg b \vee c)$: Vrai (car $\neg b =$ vrai).
  - $(b \vee c)$: Vrai si $c =$ vrai.
  - $(b \vee \neg c)$: Vrai si $c =$ faux.
  - $(\neg b \vee \neg c)$: Faux (car c ne peut être à la fois vrai et faux).

  Dans les deux cas, il est impossible de satisfaire toutes les clauses simultanément.

Conclusion

La formule $F = (b \vee c) \wedge (\neg b \vee c) \wedge (b \vee \neg c) \wedge (\neg b \vee \neg c)$ est un exemple clair d'une formule 2-SAT non satisfiable, car il n'existe aucune assignation des variables b et c qui satisfasse toutes les clauses simultanément.

3 Question 3 : Vérification de la satisfiabilité d'une formule 2-SAT

Introduction

L'objectif de cette question est de vérifier si une formule 2-SAT est satisfiable pour une assignation donnée des variables. Une formule est satisfiable si toutes ses clauses peuvent être évaluées à vrai avec une configuration spécifique des littéraux.

Pour répondre à cette question, nous avons implémenté une fonction Python nommée `verifier_sat`. Cette fonction vérifie la satisfiabilité de la formule en procédant clause par clause.

Approche utilisée pour l'implémentation

Étape 1 : Représentation des données

Nous représentons :

- La formule 2-SAT comme une liste 2D, où chaque sous-liste contient deux littéraux. Par exemple :

$$\text{formule} = [[1, 2], [-1, -2]]$$

  Ici :

  - 1 correspond à a, 2 correspond à b.
  - -1 correspond à $\neg a$, -2 correspond à $\neg b$.

- Les valeurs assignées aux variables dans une liste appelée `valeurs`. Par exemple :

$$\text{valeurs} = [\text{None}, 1, 0]$$

  Ici, $1 = \text{vrai}, 0 = \text{faux}$, et `None` est utilisé comme espace réservé pour l'indice 0, car les indices des variables commencent à 1.

Étape 2 : Évaluation des clauses

Chaque clause est évaluée indépendamment en utilisant les règles suivantes :

- Si un littéral est positif (x_i), sa valeur est directement prise dans la liste `valeurs`.
- Si un littéral est négatif ($\neg x_i$), la négation de sa valeur est calculée comme $1 - \texttt{valeurs}[i]$.
- Une clause est vraie si au moins un de ses deux littéraux est évalué à vrai.

Étape 3 : Résultat global

La formule entière est satisfiable si toutes les clauses sont évaluées à vrai. Si une seule clause est fausse, la formule est non satisfiable.

Exemple détaillé

Prenons la formule suivante :

$$F = (a \vee b) \wedge (\neg a \vee \neg b)$$

Représentation sous forme de liste :

$$\text{formule} = [[1, 2], [-1, -2]]$$

Assignons les valeurs suivantes :

$$\text{valeurs} = [\text{None}, 1, 0] \quad (\text{avec } a = \text{vrai}, b = \text{faux}).$$

Étapes d'évaluation :

- Clause $(a \vee b) = (1 \vee 0) = 1 \quad (\text{vrai})$
- Clause $(\neg a \vee \neg b) = (0 \vee 1) = 1 \quad (\text{vrai})$

Puisque toutes les clauses sont vraies, la formule est satisfiable :

$$\texttt{verifier\_sat(formule, valeurs)} \rightarrow \text{True}$$

Conclusion

Nous avons implémenté une méthode efficace pour vérifier la satisfiabilité d'une formule 2-SAT avec une assignation donnée des variables. Cette approche repose sur l'évaluation des clauses et retourne rapidement un résultat. Elle constitue une étape essentielle pour des algorithmes plus complexes de résolution de formules logiques.

4 Question 4 : Transformation d'une formule 2-SAT en graphe orienté

Introduction

L'objectif de cette question est de transformer une formule 2-SAT en un graphe orienté. Cette transformation est essentielle pour appliquer des algorithmes de théorie des graphes à la résolution des formules logiques. Chaque clause de la formule est traduite en un ensemble d'arêtes orientées, représentant les implications logiques entre les littéraux.

Approche utilisée pour l'implémentation

Étape 1 : Représentation de la formule

La formule est représentée sous forme d'une liste 2D, où chaque sous-liste correspond à une clause contenant deux littéraux. Dans votre code, la formule utilisée est la suivante :

$$\text{formule} = [[1, 2], [2, -3], [-2, -4], [2, 4], [4, 1]]$$

Ici :

- $1 = x_1$, $-1 = \neg x_1$, $2 = x_2$, $-2 = \neg x_2$, $3 = x_3$, $-3 = \neg x_3$, $4 = x_4$, $-4 = \neg x_4$.
- Chaque clause est une disjonction ($\vee$) de deux littéraux.

Étape 2 : Transformation en graphe orienté

Chaque clause $(x \vee y)$ est équivalente à deux implications logiques :

$$(x \vee y) \iff (\neg x \to y) \wedge (\neg y \to x)$$

Ainsi, chaque clause est traduite en deux arêtes dans un graphe orienté :

- $-x \to y$
- $-y \to x$

Étape 3 : Construction du graphe

Un graphe orienté est construit en ajoutant les arêtes correspondantes à chaque clause. Les nœuds du graphe représentent les littéraux positifs et négatifs (x_i et $\neg x_i$).

Exemple d'exécution avec votre formule

Formule donnée :

$$\text{formule} = [[1, 2], [2, -3], [-2, -4], [2, 4], [4, 1]]$$

Transformation en graphe orienté :

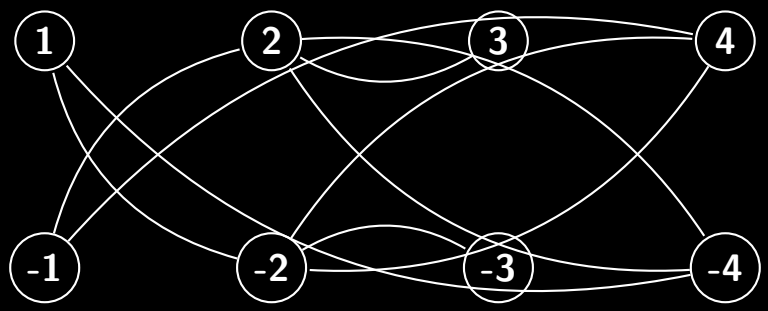
- Pour $(1 \vee 2)$:
$$-1 \to 2, \quad -2 \to 1$$
- Pour $(2 \vee \neg 3)$:
$$-2 \to -3, \quad 3 \to 2$$
- Pour $(\neg 2 \vee \neg 4)$:
$$2 \to -4, \quad 4 \to -2$$
- Pour $(2 \vee 4)$:
$$-2 \to 4, \quad -4 \to 2$$
- Pour $(4 \vee 1)$:
$$-4 \to 1, \quad -1 \to 4$$

Le graphe résultant contient les arêtes suivantes :

$$\begin{aligned} &-1 \to 2, \quad -2 \to 1, \quad -2 \to -3, \quad 3 \to 2, \\ &2 \to -4, \quad 4 \to -2, \quad -2 \to 4, \quad -4 \to 2, \\ &-4 \to 1, \quad -1 \to 4 \end{aligned}$$

Résultat visuel

Le graphe orienté peut être représenté visuellement pour une meilleure compréhension.



Conclusion

La transformation de la formule en graphe orienté permet de capturer les relations logiques entre les littéraux. Le graphe obtenu est une représentation clé pour les étapes ultérieures, comme la recherche des composantes fortement connexes.

5 Question 5 : Vérification de la satisfiabilité avec un graphe orienté

Introduction

Dans cette question, nous cherchons à déterminer si une formule 2-SAT est satisfiable en utilisant le graphe orienté obtenu à partir de la transformation de la formule (Question 4). Pour cela, nous appliquons l'algorithme des composantes fortement connexes (CFC), basé sur l'approche de Tarjan.

Une formule 2-SAT est satisfiable si et seulement si aucun littéral et son opposé ne se trouvent dans la même composante fortement connexe du graphe.

Approche utilisée pour l'implémentation

Étape 1 : Construction du graphe orienté

Le graphe orienté est construit à partir de la formule en suivant les règles expliquées dans la Question 4. Les nœuds représentent les littéraux (x_i) et leurs négations ($\neg x_i$), et les arêtes représentent les implications logiques.

Étape 2 : Recherche des composantes fortement connexes (CFC)

Nous utilisons l'algorithme de Tarjan pour identifier les composantes fortement connexes du graphe. Cet algorithme parcourt le graphe en profondeur et regroupe les nœuds qui appartiennent à une même composante fortement connexe.

Étape 3 : Vérification des conflits logiques

Pour chaque composante fortement connexe, nous vérifions si un littéral et son opposé (x_i et $\neg x_i$) apparaissent dans la même composante. Si c'est le cas, la formule est insatisfiable.

Exemple d'exécution avec votre formule

Formule donnée (tirée de votre code) :

$$\text{formule} = [[1, 2], [2, -3], [-2, -4], [2, 4], [4, 1]]$$

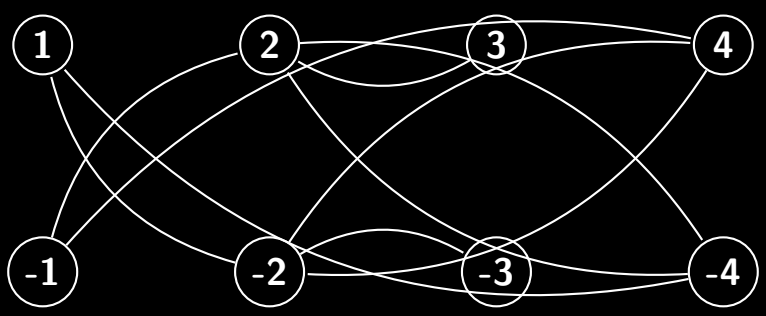
Étape 1 : Construction du graphe

Le graphe orienté est construit comme suit, en utilisant les règles d'implication :

- $-1 \to 2$, $-2 \to 1$
- $-2 \to -3$, $3 \to 2$
- $2 \to -4$, $4 \to -2$
- $-2 \to 4$, $-4 \to 2$
- $-4 \to 1$, $-1 \to 4$

Graphe obtenu

Le graphe résultant peut être représenté comme suit :



Étape 2 : Recherche des composantes fortement connexes

Les composantes fortement connexes obtenues à partir de l'algorithme de Tarjan sont les suivantes :

$$\text{CFC} = \{\{1, -2, 4\}, \{2\}, \{3\}, \{-4\}\}$$

Étape 3 : Vérification de la satisfiabilité

Pour chaque composante, nous vérifions qu'aucun littéral et son opposé ne coexistent :

- Composante $\{1, -2, 4\}$: Pas de conflit.
- Composantes $\{2\}, \{3\}, \{-4\}$: Pas de conflit.

Puisque aucune composante ne contient de conflit, la formule est satisfiable.

Conclusion

La formule formule $= [[1, 2], [2, -3], [-2, -4], [2, 4], [4, 1]]$ est satisfiable, car aucune composante fortement connexe ne contient un littéral et son opposé. Le graphe obtenu permet une vérification efficace de la satisfiabilité.