Correction TD IA - 2026

Équipe Pédagogique IA

1 TD5 : Satisfaction de Contraintes (CSP) et Logique (SAT)

Exercice 5 : Modélisation SAT du Sudoku (Exemple 4x4)

Énoncé : Modélisez le problème pour une grille 4×4 (pour simplifier les calculs). Analysez le nombre de variables et de formules.

1	2		
		2	1
2	4		
		4	2

FIGURE 1 – Exemple d'instance Sudoku 4x4 à résoudre

Proposition de Corrigé

1. Définition des Variables

On définit les variables propositionnelles $x_{i,j,k}$ telles que :

$$x_{i,j,k} = \text{Vrai} \iff \text{La case (ligne } i, \text{colonne } j\text{) contient la valeur } k$$

Pour un Sudoku 4x4 : $i, j, k \in \{1..4\}$.

Calcul du nombre de variables :

$$4 \text{ (lignes)} \times 4 \text{ (colonnes)} \times 4 \text{ (valeurs)} = \mathbf{64} \text{ variables.}$$

2. Contraintes (Formules)

A. Contrainte de représentation de l'état initial Si la grille contient un 2 en position (2,3), on ajoute la clause unitaire : $(x_{2,3,2})$.

B. Contrainte : "Une case contient au moins une valeur" Pour chaque case (i, j), il faut qu'au moins l'une des valeurs soit vraie :

$$\forall i, j : (x_{i,j,1} \vee x_{i,j,2} \vee x_{i,j,3} \vee x_{i,j,4})$$

Pour la case de la première ligne $(i = 1)$ et première colonne $(j = 1)$, cela donne :

$$(x_{1,1,1} \vee x_{1,1,2} \vee x_{1,1,3} \vee x_{1,1,4})$$

$\rightarrow$ 1 clause par case $\times$ 16 cases = **16 formules**.

TD IA-2026修正案

IA教学团队

1　TD5：约束满意度（CSP）与逻辑满意度（SAT）

练习5：数独的SAT建模（4x4示例）

陈述：对4×4的网格问题进行建模（为简化计算）。分析变量数量和公式数量。

1	2		
		2	1
2	4		
		4	2

图1 – 4x4数独实例待解示例

修正提案

1. 变量定义

我们定义命题变量$x_{i,j,k}$如下：

$x_{i,j,k}$= 真ﻓﻢ ﻑ 单元格（行i，列j）包含值k

对于4x4数独：i，j，k　$\{1..4\}$。

变量数量的计算：

$$4\text{（行）} \times 4\text{（列）} \times 4\text{（值）} = \mathbf{64}\text{个变量。}$$

2. 约束（公式）

A. 初始状态表示约束若网格在(2,3)位置存在2，则
添加单元条款：$(x_{2,3,2})$。

B. 约束条件：“每个单元格至少包含一个值”对于每个单元格（i，j），必须至少有一个值为真：

$$\forall i, j: (x_{i,j,1} \quad x_{i,j,2} \quad x_{i,j,3} \quad x_{i,j,4})$$

对于第一行（$i=1$）和第一列（$j=1$）的单元格，计算结果为：

$$(x_{1,1,1} \quad x_{1,1,2} \quad x_{1,1,3} \quad x_{1,1,4})$$

ﻻ 每个案例1个条款× 16个案例 =**16个公式**。

C. Contrainte : "Une case contient au plus une valeur" C'est ici qu'on peut modéliser de deux façons.

— <u>**Solution 1 : Approche Logique (Implication)**</u> On dit : "Si la case vaut k, alors elle ne peut pas valoir les autres".
Exemple pour la case (1,1) et la valeur 1 :

$$x_{1,1,1} \implies \neg(x_{1,1,2} \lor x_{1,1,3} \lor x_{1,1,4})$$

$\rightarrow$ 4 formules par case $\times$ 16 cases $=$ **64 formules**.
Total B + C (Solution 1) : $16 + 64 = 80$ formules.

— <u>**Solution 2 : Approche CNF (Paires exclusives - XOR)**</u> Les solveurs préfèrent les clauses simples. On interdit toutes les paires de valeurs pour une même case.
Pour une case donnée, on interdit :

  — $(\neg x_{i,j,1} \lor \neg x_{i,j,2})$
  — $(\neg x_{i,j,1} \lor \neg x_{i,j,3})$ ... et ainsi de suite.

Il y a 6 paires possibles parmi 4 valeurs.
$\rightarrow$ 6 clauses par case $\times$ 16 cases $=$ **96 formules**.
Total B + C (Solution 2) : $16 + 96 = 112$ formules (clauses).

D. Contraintes de validité (Lignes, Colonnes, Carrés) Pour s'assurer que la grille est valide, il suffit d'imposer que **chaque chiffre apparaisse au moins une fois** dans chaque groupe (ligne, colonne, carré).

Raisonnement logique : Puisqu'il y a 4 cases dans une ligne et 4 chiffres possibles, si chaque chiffre apparaît au moins une fois, alors ils apparaissent tous exactement une fois (principe des tiroirs), à condition que les contraintes sur les cases (B et C) soient respectées.

— **Pour les lignes :** Pour chaque ligne i et chaque valeur k, k doit apparaître au moins une fois.
Exemple (Ligne 1 doit contenir un 1) :

$$(x_{1,1,1} \lor x_{1,2,1} \lor x_{1,3,1} \lor x_{1,4,1})$$

$\rightarrow$ 4 lignes $\times$ 4 valeurs $=$ **16 formules**.

— **Pour les colonnes :** Idem.
$\rightarrow$ 4 colonnes $\times$ 4 valeurs $=$ **16 formules**.

— **Pour les carrés :** Idem.
$\rightarrow$ 4 carrés $\times$ 4 valeurs $=$ **16 formules**.

Note : On n'a pas besoin d'ajouter les clauses d'unicité (AMO) pour les lignes/colonnes ici, car elles sont implicites par la combinaison des contraintes précédentes.

Résumé total (Solution CNF standard) :

$$112 \text{ (Cases)} + 48 \text{ (Validité)} + \text{Clauses unitaires (indices)} = \mathbf{160} \text{ clauses} + \text{indices.}$$

C. 约束条件：“每个单元格最多只能包含一个值”这里可以有两种建模方式。

—<u>解决方案1：逻辑方法（蕴含）</u>说：“如果某个单元格的值为k，那么它就不能有其他值。”

以(1,1)坐标点为例，其数值为1：

$$x_{1,1,1} \Rightarrow \neg(x_{1,1,2}\ \ x_{1,1,3}\ \ x_{1,1,4})$$

⇒ 每格4个公式× 16格 =**64个公式**。

B + C（溶液1）：16 + 64 = 80个公式。

—<u>解决方案2：CNF方法（排他对——异或）</u>解决器更倾向于使用简单子句。同一单元格内禁止所有值对的组合。

对于给定的单元格，禁止：

—$(\neg x_{i,j,1}\ \ \neg x_{i,j,2})$

—$(\neg x_{i,j,1}\ \ \neg x_{i,j,3})$ ... 以此类推。

在4个数值中，共有6种可能的配对组合。

⇒ 每个案例6个条款× 16个案例 =**96个公式**。

B + C 总数（方案 2）：16 + 96 = 112 个条款。

D. 有效性约束（行、列、方格）为确保网格有效，只需规定**每个数字在每组（行、列、方格）中至少出现一次**即可。

逻辑推理: 由于每行有4个格子且共有4个可能的数字，若每个数字至少出现一次，那么它们都会恰好出现一次（抽屉原理），前提是满足格子（B和C）的约束条件。

—对于行：对于每一行i和每一个值k，k至少出现一次。示例（第1行必须包含1）：

$$(x_{1,1,1}\ \ x_{1,2,1}\ \ x_{1,3,1}\ \ x_{1,4,1})$$

⇒ 4行× 4值 =**16个公式**。

—**对于列：**同上。

⇒ 4列× 4值 =**16个公式**。

—**对于方块：**同上。

⇒ 4个方格× 4个数值 =**16个公式**。

注意: 无需在此处添加行/列的唯一性约束（AMO），因为前文约束的组合已隐含了这些约束条件。

总体摘要（标准CNF溶液）：

112 （案例）+ 48（有效性）+ 单一条款（索引）=**160**条款 + 索引。

TD 6 : Comparaison SAT vs CSP

Critère	**Approche SAT (4x4)**	**Approche CSP (4x4)**
Variables	**64** variables booléennes ($x_{i,j,k}$)	**16** variables entières ($X_{i,j}$)
Domaines	$\{0, 1\}$ (Vrai/Faux)	$\{1, 2, 3, 4\}$
Contraintes	80 à 112 formules logiques (Clauses CNF, bas niveau)	12 contraintes globales (`AllDifferent`, haut niveau)
Résolution	Solveurs très rapides	Plus compact à modéliser

Conclusion : Le CSP offre une modélisation beaucoup plus compacte ("Haut niveau") grâce aux variables entières, tandis que SAT ("Bas niveau") explose en nombre de variables mais bénéficie de solveurs extrêmement optimisés.

TD 6：SAT与CSP的比较

标准	SAT方法(4x4)	CSP方法(4x4)
变量	**64** variables booléennes ($x_{i,j,k}$)	**16**个整数变量（Xi，j）
多发性骨髓瘤	{0,1}（真/假）	$\{1, 2, 3, 4\}$
约束	80 112种逻辑公式（CNF条款，基础层级）	12个全局约束（AllDifferent，高级）
分辨率	超高速求解器	更便于建模的紧凑型结构

结论：CSP通过整数变量实现了更紧凑的建模（“高层级”），而SAT（“底层级”）虽然变量数量激增，但得益于高度优化的求解器。