

DM3: Conception et utilisation d'un SAT-solver

TODO

19 mai 2025

1 Conception du SAT-solver

Q10.

Soit n le nombre d'opérateurs.

Dans la configuration $(\dots((a_0|a_1)|a_2)|\dots)|a_n$, il y a bien n opérateurs.

De plus, la complexité de l'algorithme dans cette configuration est $C_n = C_{n-1} + \Omega(n)$.

Il existe donc un $n_0 \in \mathbb{N}$ tel que $\forall n > n_0$:

$$C_n \geq C_{n-1} + An \implies C_n \geq C_{n_0} + A \sum_{k=n_0+1}^n k$$

Donc $C_n - C_{n_0} = \Omega(n^2)$. Or C_{n_0} est une constante.

Donc dans le pire cas, la complexité est au moins de l'ordre de $\Omega(n^2)$.

Pour améliorer cette fonction, on pourrait par exemple utiliser un arbre rouge-noir.

Q11. (Bonus)

Dans la nouvelle fonction, on a

$$C_n = \mathcal{O}(n) + \mathcal{O}(n \log(n))$$
$$C_n = \boxed{\mathcal{O}(n \log(n))}$$

Q19.

La complexité de l'algorithme dans la configuration $\sim (\dots(\sim T)\dots)$ est

$$C_n = C_{n-2} + \Theta(n)$$
$$C_n = \boxed{\Theta(n^2)}$$

Q20. (Bonus)

Dans la nouvelle fonction, on simplifie les enfants avant le nœud.

La complexité est

$$C_n = C_i + C_{n-1-i} + \Theta(1), \quad i \in \llbracket 0, n-1 \rrbracket$$
$$C_n = \boxed{\Theta(n)}$$

Q25.

Nous avons tenté de choisir la variable en fonction de la fréquence d'apparition : celle qui apparaît le plus est choisie en premier et est mise à \top . Nous l'avons implémenté de cette manière :

```
1 (* Renvoie la variable dont la fréquence d'apparition dans f est la plus élevée. *)
2 let find_var (f: formule) : string =
3   let rec find_var_aux (f: formule) (dico: (string, int) dico) (m: string*int): (string*int) * (string, int)
4     dico =
5       match f with
6       | Top | Bot -> m, dico
7       | Var(q) ->
8         begin
9           match get dico q with
10          | None when snd m < 1 -> (q, 1), set dico q 1
11          | None -> m, set dico q 1
12          | Some v when snd m <= v -> (q, v+1), set dico q (v + 1)
13          | Some v -> m, set dico q (v + 1)
14        end
15       | Not(f') -> find_var_aux f' dico m
16       | And (f1, f2) | Or(f1, f2) -> let m', d' = find_var_aux f2 dico m in
17         find_var_aux f1 d' m'
18   in fst(fst(find_var_aux f None ("", 0)))
19
20 (* quine en utilisant find_var (lent) *)
21 let rec quine2 (f: formule) : sat_result =
22   if f = Top then Some []
23   else if f = Bot then None
24   else match find_var f with
25   | "" -> quine (simpl f)
26   | x -> let f' = simpl (subst f x Top) in
27     match quine f' with
28     | Some v -> Some ((x, true)::v)
29     | None -> let f'' = simpl (subst f x Bot) in
30       match quine f'' with
31       | None -> None
32       | Some v' -> Some ((x, false)::v')
```

Cette méthode est particulièrement longue et inefficace. Pour voir comment est implémentée la structure de dictionnaire, le fichier est disponible via [ce lien](#).

Q26. (Bonus)

Le satsolver spécialisé en FNC a été implémenté dans `fnc_solver.ml`. De plus, la propagation unitaire a été rajoutée.

En termes d'efficacité, le satsolver en FNC est meilleur que le simple. En effet, dans la résolution du problème à n dames (cf. seconde partie), le satsolver en FNC permet de résoudre plus de dames et en moins de temps.

2 Résolution de problèmes

Q31.

La formule $\bigwedge_{1 \leq i < j \leq n} (\neg a_i \vee \neg a_j)$ est sous FNC.

Pour n variables différentes, la formule contient $n(n-1)$ variables.

Calculs de mémoire (Q29 - Q38)

On note $T(n)$ la taille des formules en nombre de caractères. et $p = 2(\lfloor \log_{10}(n) \rfloor + 1) + 3$ la taille maximale d'une variable X_{i_j} .

au_moins_une :

La formule est de la forme : $(X_{0_0} \mid \dots \mid X_{n_n})$.

$$T(n) \leq n(p+3)$$

au_plus_une :

La formule est de la forme : $((\sim X_{0_0} \mid \sim X_{0_1}) \& \dots \& (\sim X_{n_n} \mid X_{n_n}))$

Il y a $\frac{n(n-1)}{2}$ clauses, chaque clause est de taille $2p+7$ et entre chaque clause il y a 3 caractères.

$$T(n) \leq \frac{n(n-1)}{2}(2p+10) = n(n-1)(p+5)$$

Contrainte sur une ligne :

La formule est de la forme : (au_moins_une & au_plus_une)

$$T(n) \leq 5 + n(n-1)(p+5) + n(p+3)$$

Contrainte sur toutes les lignes :

La formule est de la forme : (contrainte_une_ligne & ... & contrainte_une_ligne)

$$T(n) \leq n(3 + 5 + n(p+3) + n(n-1)(p+5))$$

Contrainte sur toutes les colonnes :

La formule est de la forme : (au_plus_une & ... & au_plus_une)

$$T(n) \leq n(3 + 5 + n(n-1)(p+5))$$

Contrainte sur toutes les diagonales :

La formule est de la forme : (au_plus_une & ... & au_plus_une) avec $2(n-1)(n-2)$ au_plus_une.

$$T(n) \leq 2n(n-1)^2(n-2)(p+5)$$

Q38.

La taille de la formule de `gen_formule_n_dames` est

$$\begin{aligned} C_n &= C_{ligne}(n) + C_{col}(n) + C_{diag}(n) + \Theta(1) \\ &= nE_x(n) + nE_p(n) + 2 \sum_{i=-n+2}^{n-2} E_p(|n-i|) + \Theta(n) \end{aligned}$$

Avec E_x la fonction `exactement_une` et E_p `au_plus_une`.

Or $nE_x(n) = \Theta(n^3)$, $nE_p(n) = \Theta(n^3)$ et $2 \sum_{i=-n+2}^{n-2} E_p(|n-i|) \leq 4nE_p(n) = \mathcal{O}(n^3)$, donc

$$\begin{aligned} C_n &= \Theta(n^3) + \Theta(n^3) + \mathcal{O}(n^3) + \Theta(n) \\ C_n &= \boxed{\mathcal{O}(n^3)} \end{aligned}$$

Q40.

Pour le problème à 5 dames, on obtient

```
1 fred@mp2:~/DM3$ problemes/n_dames 5
2 Fichier '5_dames.txt' généré.
3 Taille du fichier : 3471 octets.
4 fred@mp2:~/DM3$ satsolver/fnc_solver '5_dames.txt'
5 La formule est sous FNC.
6 La formule est satisfiable en assignant 1 aux variables suivantes et 0 aux autres :
7 X_0_4
8 X_1_2
9 X_2_0
10 X_3_3
11 X_4_1
12 Temps d'exécution : 0.001377 s
```

Soit encore

	0	1	2	3	4
0			X		
1					X
2		X			
3				X	
4	X				

Pour le problème à 8 dames, on obtient

```

1 fred@mp2:~/DM3$ satsolver/fnc_solver '8_dames.txt'
2 La formule est sous FNC.
3 La formule est satisfiable en assignant 1 aux variables suivantes et 0 aux autres :
4 X_0_7
5 X_1_3
6 X_2_0
7 X_3_2
8 X_6_6
9 X_5_1
10 X_4_5
11 X_7_4
12 Temps d'exécution : 0.005287 s

```

Soit encore

	0	1	2	3	4	5	6	7
0			X					
1						X		
2				X				
3		X						
4								X
5					X			
6							X	
7	X							

Et pour le problème à 3 dames, on obtient

```

1 fred@mp2:~/DM3$ satsolver/fnc_solver '3_dames.txt'
2 La formule est sous FNC.
3 La formule est insatisfiable.
4 Temps d'exécution : 0.000985 s

```

3 Problème des cinq maisons

3.1 Description

Le problème des cinq maisons a les contraintes suivantes :

1. φ_1 : Chaque caractéristique (« anglais », « poisson rouge », ...) se retrouve dans exactement une des cinq maisons
2. φ_2 : Chaque maison doit avoir exactement une caractéristique de chaque catégorie (nationalité, boisson, couleur, ...). Par exemple, la maison 1 doit avoir exactement une couleur.
3. φ_3 : Contraintes de l'énoncé

3.2 Nomenclature

Les variables utilisées dans les formules sont de la forme **numéro_caractéristique**. Par exemple :

- **1_anglais** représente « l'anglais habite dans la maison 1 »
- **5_poisson_rouge** représente « la personne habitant la maison 5 a pour animal de compagnie un poisson rouge »
- **3_yop** représente « la personne habitant la maison 3 boit du yop »
- etc.

Dans la suite de cette section, notons C l'ensemble des caractéristiques ($C = \{\text{anglais, lait, escalade, ...}\}$), \mathcal{C} l'ensemble des catégories de caractéristiques (nationalité, couleur, ...) (donc $C = \bigsqcup_{c \in \mathcal{C}} c$) et utilisons des indices de maison dans $\llbracket 1, 5 \rrbracket$.

3.3 Modélisation de la contrainte 1

Pour modéliser la contrainte « Chaque caractéristique se retrouve dans exactement une des cinq maisons », on peut d'abord modéliser la contrainte « La caractéristique c se retrouve dans exactement une maison », en utilisant la formule suivante :

$$\bigvee_{j=1}^5 \left(j_c \wedge \bigwedge_{\substack{i=1 \\ i \neq j}}^5 \neg i_c \right)$$

(i.e. « Soit cette caractéristique est dans la maison 1 et aucune autre, soit dans la 2 et aucune autre, etc. »)

Ainsi, on peut modéliser la contrainte sur toutes les caractéristiques en faisant un « et » logique :

$$\varphi_1 \equiv \bigwedge_{c \in C} \bigvee_{j=1}^5 \left(j_c \wedge \bigwedge_{\substack{i=1 \\ i \neq j}}^5 \neg i_c \right)$$

3.4 Modélisation de la contrainte 2

Pour modéliser la contrainte « Chaque maison doit avoir exactement une caractéristique de chaque catégorie », on peut d'abord modéliser la contrainte « La maison i a exactement une caractéristique de la catégorie $\mathcal{C} \in \mathcal{C}$ », en utilisant la formule suivante :

$$\bigvee_{c' \in \mathcal{C}} \left(i_c' \wedge \bigwedge_{\substack{c \in \mathcal{C} \\ c \neq c'}} \neg i_c \right)$$

On peut alors modéliser la contrainte sur toutes les catégories de caractéristiques avec un « et » logique :

$$\bigwedge_{\mathcal{C} \in \mathcal{C}} \bigvee_{c' \in \mathcal{C}} \left(i_c' \wedge \bigwedge_{\substack{c \in \mathcal{C} \\ c \neq c'}} \neg i_c \right)$$

Et de même, on modélise la contrainte sur toutes les maisons ainsi :

$$\varphi_2 \equiv \bigwedge_{i=1}^5 \bigwedge_{\mathcal{C} \in \mathcal{C}} \bigvee_{c' \in \mathcal{C}} \left(i_c' \wedge \bigwedge_{\substack{c \in \mathcal{C} \\ c \neq c'}} \neg i_c \right)$$

3.5 Modélisation de la contrainte 3

Les contraintes de l'énoncé se modélisent assez facilement. Par exemple, « L'Anglais vit dans une maison rouge » se modélise ainsi :

$$\bigvee_{i=1}^5 i_anglais \wedge i_rouge$$

« La personne qui fait de l'escalade vit à côté de celle qui a des chats » se modélise avec cette formule :

$$\left(\bigvee_{i=1}^4 i_escalade \wedge (i+1)_chats \right) \vee \left(\bigvee_{i=1}^4 i_chats \wedge (i+1)_escalade \right)$$

Encore plus simple, « La personne qui vit dans la maison du centre boit du lait » se modélise avec la formule suivante :

$$3_lait$$

Il suffit ensuite de faire un « et » logique sur toutes les formules données par les contraintes pour obtenir φ_3 .

3.6 Solution

Le satsolver résoud le problème en 0.4 secondes. À noter que l'ordre des contraintes est important : écrire dans le fichier $\varphi_1 \wedge \varphi_2 \wedge \varphi_3$ prend au moins 10 minutes (peut-être beaucoup plus, on l'a arrêté avant), tandis que $\varphi_3 \wedge \varphi_2 \wedge \varphi_1$ est beaucoup plus rapide.

La solution est la suivante :

Maison	1	2	3	4	5
Couleur	Jaune	Bleu	Rouge	Vert	Blanc
Nationalité	Norvégien	Danois	Anglais	Allemand	Suédois
Animal	Chats	Cheval	Oiseaux	Poisson rouge	Chiens
Boisson	Eau	Thé	Lait	Café	Yop
Sport	Danse	Escalade	Vélo	Karaté	Basket

Ainsi, le poisson rouge est l'animal de compagnie de l'Allemand.

4 Problème du calendrier

4.1 Description

Pour la résolution du problème du calendrier, il faut respecter ces deux règles :

1. Chaque case a une pièce (voire 0 pour certaines cases).
2. Chaque pièce n'est utilisée qu'une seule fois.

Pour représenter le problème, chaque case aura 10 variables, indiquant si la pièce correspondante est dessus.

Elle seront de la forme X_0_0 avec X le nom de la pièce, (0,0) les coordonnées de la case. Exemples : 1_4_5 , T_2_7 . (Les noms des pièces sont I, L, S, b, C, l, s, Z, T et V)

Pour optimiser, les cases qui doivent avoir 0 pièce ne seront ni créées ni utilisées.

4.2 Définition des ensembles

On définit $C \subset \llbracket 0, 7 \rrbracket^2$ l'ensemble des cases qui doivent être remplies, implémentés avec `date` et le tableau `calendrier`.

On définit \mathbb{P} l'ensemble des pièces, implémenté dans `pieces.c` et `pieces.h` (\mathbb{P} est ordonné).

On définit \mathbb{V}_p l'ensemble des positions valides de la pièce $p \in \mathbb{P}$, implémenté avec `piece_valide`.

Pour $p \in \mathbb{P}$ et $v \in \mathbb{V}_p$, on définit N_v par

$$\forall (i, j) \in C, \quad N_v(p_{i,j}) = \begin{cases} p_{i,j} & \text{si } p \text{ dans la position } v \text{ est sur la case } (i, j) \\ \neg p_{i,j} & \text{sinon} \end{cases}$$

$$\text{Exemple pour la pièce } T \text{ de position } v = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

On a $N_v(T_{1,2}) = T_{1,2}$ et $N_v(T_{3,1}) = \neg T_{3,1}$

4.3 Modélisation de la contrainte sur une case avec qu'une pièce

Pour $p_{i,j}$ les variables « p est en (i, j) », la contrainte sur la case $(i, j) \in C$ est

$$\varphi_{i,j} = \left(\bigvee_{p \in \mathbb{P}} p_{i,j} \right) \wedge \bigwedge_{\substack{(p,p') \in \mathbb{P}^2 \\ p < p'}} (\neg p_{i,j} \vee \neg p'_{i,j})$$

Cette formule est sous FNC.

Elle est implémentée dans `contrainte_une_case`.

4.4 Modélisation de la contrainte sur toutes les cases avec qu'une pièce

La contrainte sur toutes les cases est

$$\begin{aligned}\varphi &= \bigwedge_{(i,j) \in C} \varphi_{i,j} \\ &= \bigwedge_{(i,j) \in C} \left(\left(\bigvee_{p \in \mathbb{P}} p_{i,j} \right) \wedge \bigwedge_{\substack{(p,p') \in \mathbb{P}^2 \\ p < p'}} (\neg p_{i,j} \vee \neg p'_{i,j}) \right)\end{aligned}$$

Cette formule est sous FNC.

Elle est implémenter dans `contrainte_toutes_cases`.

4.5 Modélisation de la contrainte sur une pièce

La contrainte sur la pièce $p \in \mathbb{P}$ est

$$\varphi'_p = \bigvee_{v \in \mathbb{V}_p} \left(\bigwedge_{(i,j) \in C} N_v(p_{i,j}) \right)$$

Mais cette formule n'est pas sous FNC, on doit passer par des variables intermédiaires que l'on nomme $Z_{p,v}$.

$$\begin{aligned}\varphi'_p &= \bigvee_{v \in \mathbb{V}_p} \left(\bigwedge_{(i,j) \in C} N_v(p_{i,j}) \right) \\ &= \left(\bigvee_{v \in \mathbb{V}_p} Z_{p,v} \right) \wedge \bigwedge_{v \in \mathbb{V}_p} \left(Z_{p,v} \leftrightarrow \bigwedge_{(i,j) \in C} N_v(p_{i,j}) \right) \\ &= \left(\bigvee_{v \in \mathbb{V}_p} Z_{p,v} \right) \wedge \bigwedge_{v \in \mathbb{V}_p} \left(Z_{p,v} \rightarrow \bigwedge_{(i,j) \in C} N_v(p_{i,j}) \right) \wedge \bigwedge_{v \in \mathbb{V}_p} \left(\neg Z_{p,v} \rightarrow \neg \bigwedge_{(i,j) \in C} N_v(p_{i,j}) \right)\end{aligned}$$

Que l'on peut simplifier en

$$\begin{aligned}\varphi'_p &= \left(\bigvee_{v \in \mathbb{V}_p} Z_{p,v} \right) \wedge \bigwedge_{v \in \mathbb{V}_p} \left(Z_{p,v} \rightarrow \bigwedge_{(i,j) \in C} N_v(p_{i,j}) \right) \\ &= \left(\bigvee_{v \in \mathbb{V}_p} Z_{p,v} \right) \wedge \bigwedge_{v \in \mathbb{V}_p} \left(\neg Z_{p,v} \vee \bigwedge_{(i,j) \in C} N_v(p_{i,j}) \right) \\ &= \left(\bigvee_{v \in \mathbb{V}_p} Z_{p,v} \right) \wedge \bigwedge_{\substack{(i,j) \in C \\ v \in \mathbb{V}_p}} (\neg Z_{p,v} \vee N_v(p_{i,j}))\end{aligned}$$

qui est sous FNC.

Elle est implémentée en 2 parties dans `contrainte_piece_pos` et dans `contrainte_une_piece`.

4.6 Modélisation de la contrainte sur toutes les pièces

La contrainte sur toutes les pièces est

$$\begin{aligned}\varphi' &= \bigwedge_{p \in \mathbb{P}} \varphi'_p \\ &= \bigwedge_{p \in \mathbb{P}} \left(\left(\bigvee_{v \in \mathbb{V}_p} Z_{p,v} \right) \wedge \bigwedge_{\substack{(i,j) \in C \\ v \in \mathbb{V}_p}} (\neg Z_{p,v} \vee N_v(p_{i,j})) \right)\end{aligned}$$

Cette formule est sous FNC.

Elle est implémentée dans `contrainte_toutes_pieces`.

4.7 Formule générale

La formule générale est donc

$$\Phi = \varphi \wedge \varphi'$$

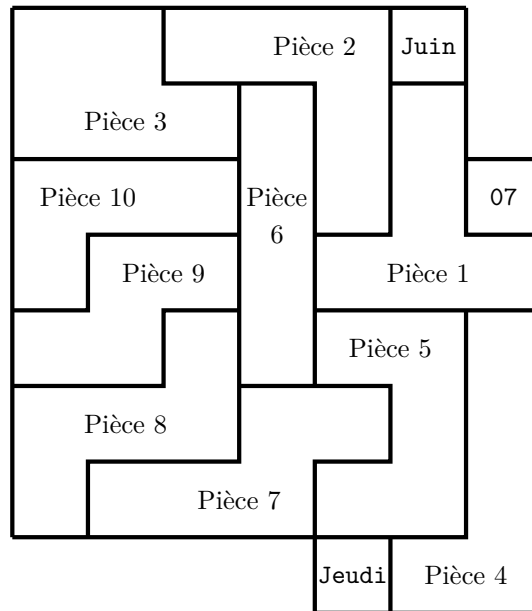
$$= \bigwedge_{(i,j) \in C} \left(\left(\bigvee_{p \in \mathbb{P}} p_{i,j} \right) \wedge \bigwedge_{\substack{(p,p') \in \mathbb{P}^2 \\ p < p'}} (\neg p_{i,j} \vee \neg p'_{i,j}) \right) \wedge \bigwedge_{p \in \mathbb{P}} \left(\left(\bigvee_{v \in \mathbb{V}_p} Z_{p,v} \right) \wedge \bigwedge_{\substack{(i,j) \in C \\ v \in \mathbb{V}_p}} (\neg Z_{p,v} \vee N_v(p_{i,j})) \right)$$

4.8 Solution

Malheureusement, la complexité de cette formule est trop élevée pour les 10 pièces. Donc, en plus d'avoir implémenté le problème dans `calendrier.c`, une version simplifiée est implémentée dans `calendrier_n_pieces.c`.

Ce programme représente le Jeudi 7 Juin avec, en entrée, le nombre de pièces à prendre entre 2 et 10.

L'une des solution attendue pour 10 pièces est



Pour le problème avec 6 pièces, on obtient

```
1 fred@mp2:~/DM3$ problemes/calendrier_n_pieces 6
2 Fichier 'calendrier_6_pieces.txt' généré.
3 Taille du fichier : 182857 octets.
4 fred@mp2:~/DM3$ satsolver/fnc_solver calendrier_6_pieces.txt | problemes/print_tab 8
5 La formule est sous FNC.
6 La formule est satisfiable en assignant 1 aux variables suivantes et 0 aux autres :
7 C_4_4
8 alias_C28
9 C_4_5
10 C_5_5
11 C_6_4
12 C_6_5
13 I_1_3
14 I_2_3
15 I_3_3
16 I_4_3
17 alias_L23
18 L_4_6
19 L_5_6
20 L_6_6
21 L_7_5
22 L_7_6
23 T_1_5
24 T_2_5
25 T_3_4
26 alias_V3
27 V_0_2
```



```

28 V_0_3
29 V_0_4
30 V_1_4
31 V_2_4
32 b_0_0
33 b_0_1
34 alias_b43
35 b_1_0
36 b_1_1
37 b_1_2
38 T_3_5
39 T_3_6
40 alias_I26
41 alias_T31
42 Temps d'exécution : 4.159915 s
43 +---+---+---+---+---+---+
44 |      |      |      |      |
45 +      +---+---+ +---+      +
46 |      |      |      |      |
47 +---+---+---+ + + +      +
48 |      |      |      |      |
49 +      + +---+ +---+      +
50 |      |      |      |      |
51 +      + +---+---+---+      +
52 |      |      |      |      |
53 +      +---+---+ + + +      +
54 |      |      |      |      |
55 +      +---+ + + +      +
56 |      |      |      |      |
57 +      +---+---+ + +      +
58 |      |      |      |      |
59 +---+---+---+---+---+---+

```

Et pour le problème avec 7 et 8 pièces, on obtient

```

1 fred@mp2:~/DM3$ satsolver/fnc_solver calendrier_7_pieces.txt | problemes/print_tab 8
2 La formule est sous FNC.
3 La formule est satisfiable en assignant 1 aux variables suivantes et 0 aux autres :
4 C_4_4
5 ...
6 alias_T48
7 Temps d'exécution : 45.569438 s
8 +---+---+---+---+---+---+
9 |      |      |      |      |
10 +      +---+---+ +---+      +
11 |      |      |      |      |
12 +---+---+---+ + + +      +
13 |      |      |      |      |
14 +      + +---+ +---+      +
15 |      |      |      |      |
16 +      + +---+---+---+      +
17 |      |      |      |      |
18 +      +---+---+ + + +      +
19 |      |      |      |      |
20 + +---+---+ +---+ + + +      +
21 |      |      |      |      |
22 + +---+---+---+---+---+      +
23 |      |      |      |      |
24 +---+---+---+---+---+---+
25 fred@mp2:~/DM3$ satsolver/fnc_solver calendrier_8_pieces.txt | problemes/print_tab 8
26 La formule est sous FNC.
27 La formule est satisfiable en assignant 1 aux variables suivantes et 0 aux autres :
28 C_4_4
29 ...
30 alias_T58
31 Temps d'exécution : 675.462875 s
32 +---+---+---+---+---+---+
33 |      |      |      |      |
34 +      +---+---+ +---+      +
35 |      |      |      |      |
36 +---+---+---+ + + +      +
37 |      |      |      |      |
38 +      + +---+ +---+      +
39 |      |      |      |      |
40 +      +---+ +---+---+---+      +
41 |      |      |      |      |
42 +---+---+ +---+---+ + + +      +
43 |      |      |      |      |
44 + +---+---+ +---+ + + +      +
45 |      |      |      |      |
46 +---+---+---+---+---+---+      +
47 |      |      |      |      |
48 +---+---+---+---+---+---+

```

Mais il n'est pas possible de résoudre avec 9 pièces en un temps convenable.