Projet Squaro

INF 242 - Logique propositionnelle et logique du premier ordre

JACOB Julien et HECKMAN Victor

Sommaire

Introduction

Le projet

Dans ce rapport, nous nous proposons de formaliser les règles du jeu de squaro en un problème logique exprimé sous la forme d'un ensemble de clauses solvables par un SAT-Solveur.

Dans un second temps, nous coderons un SAT-Solveur, le WalkSat (tel que défini dans le sujet) auquel nous ajouterons différentes optimisations et choisirons une méthode déterministe pour sélectionner une partie des variables.

Enfin, nous présenterons des programmes capables de créer une grille de squaro, la traduire en un fichier dimacs correspondant en fonction des clauses établies et donner une solution à la grille grâce au résultat du WalkSat.

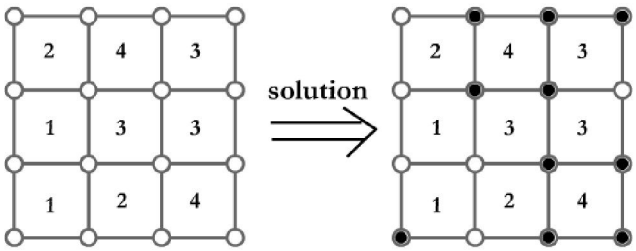
Pour réaliser les programmes du projet, nous avons choisi d'utiliser le langage JavaScript accompagné de fichiers HTML.

Pour lancer les programmes, il suffit d'ouvrir le fichier « index.html » avec un navigateur moderne (Google Chrome ou Mozilla Firefox par exemple).

Le squaro

Le but du Squaro est de remplir les ronds encadrant une valeur de manière à ce que le nombre de ronds cochés soit égal au nombre au centre de la case pour toutes les cases de la grille.

Exemple de la résolution d'une grille de Squaro :



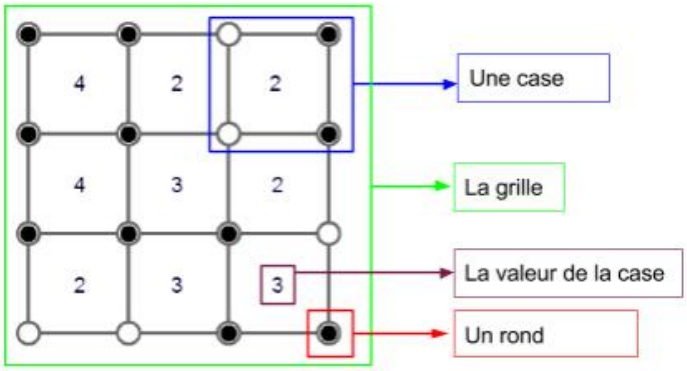
Les règles du squaro sont donc :

1. Chaque case contient un nombre compris entre 0 et 4.
2. Les ronds sont tous vides en début de partie (ils ne peuvent être que remplis ou vide).
3. Le joueur peut cliquer sur un rond vide pour le remplir, ou sur un rond remplit pour le vider.
4. Le joueur remporte la partie quand chaque case contient le même nombre de ronds remplis que le nombre en son centre.

Il est à noter qu'une grille de X cases de hauteur sur Y cases de large comprendra (X+1)*(Y+1) ronds.

Introduction d'un vocabulaire

Afin de pouvoir communiquer efficacement, mais aussi afin de nommer nos futurs objets, variables et fonctions, il est important de définir un vocabulaire standardisé pour notre projet.



Formalisation des règles du squaro en logique

Dans cette partie, nous allons montrer comment formaliser les règles du squaro en logique pour arriver à un problème sous forme normale conjonctive. Nous montrerons deux solutions qui mènent à cette formalisation, l'une s'appuie sur les modèles, l'autre s'appuie sur les contres modèles. Après comparaison, nous choisirons celle qui sera utilisée par la suite dans l'implémentation logicielle du projet. Enfin, un algorithme de transformation de grille de squaro en un ensemble de clauses est donné dans le dernier chapitre de cette partie.

Formalisation à partir des modèles

Supposons une case de valeur y, entourée par 4 ronds : x_1, x_2, x_3, x_4. Pour une valeur donnée, la case du squaro aura un nombre fini de solutions. En utilisant une table de vérité (présentée en annexe), nous obtenons la forme normale disjonctive qui régit la réussite de la case selon sa valeur. Ce sont nos formules propositionnelles.

X1 X2

Y

X3 X4

- Si $y = 0$ alors : $\neg \boldsymbol{x_1} \wedge \neg \boldsymbol{x_2} \wedge \neg \boldsymbol{x_3} \wedge \neg \boldsymbol{x_4}$

- Si $y = 1$ alors : $(\boldsymbol{x_1} \wedge \neg \boldsymbol{x_2} \wedge \neg \boldsymbol{x_3} \wedge \neg \boldsymbol{x_4})$
  $\vee\ (\neg \boldsymbol{x_1} \wedge \boldsymbol{x_2} \wedge \neg \boldsymbol{x_3} \wedge \neg \boldsymbol{x_4})$
  $\vee\ (\neg \boldsymbol{x_1} \wedge \neg \boldsymbol{x_2} \wedge \boldsymbol{x_3} \wedge \neg \boldsymbol{x_4})$
  $\vee\ (\neg \boldsymbol{x_1} \wedge \neg \boldsymbol{x_2} \wedge \neg \boldsymbol{x_3} \wedge \boldsymbol{x_4})$
- Si $y = 2$ alors : $(\boldsymbol{x_1} \wedge \boldsymbol{x_2} \wedge \neg \boldsymbol{x_3} \wedge \neg \boldsymbol{x_4})$
  $\vee\ (\boldsymbol{x_1} \wedge \neg \boldsymbol{x_2} \wedge \boldsymbol{x_3} \wedge \neg \boldsymbol{x_4})$
  $\vee\ (\boldsymbol{x_1} \wedge \neg \boldsymbol{x_2} \wedge \neg \boldsymbol{x_3} \wedge \boldsymbol{x_4})$
  $\vee\ (\neg \boldsymbol{x_1} \wedge \boldsymbol{x_2} \wedge \boldsymbol{x_3} \wedge \neg \boldsymbol{x_4})$
  $\vee\ (\neg \boldsymbol{x_1} \wedge \boldsymbol{x_2} \wedge \neg \boldsymbol{x_3} \wedge \boldsymbol{x_4})$
  $\vee\ (\neg \boldsymbol{x_1} \wedge \neg \boldsymbol{x_2} \wedge \boldsymbol{x_3} \wedge \boldsymbol{x_4})$
- Si $y = 3$ alors : $(\boldsymbol{x_1} \wedge \boldsymbol{x_2} \wedge \boldsymbol{x_3} \wedge \neg \boldsymbol{x_4})$
  $\vee\ (\boldsymbol{x_1} \wedge \neg \boldsymbol{x_2} \wedge \boldsymbol{x_3} \wedge \boldsymbol{x_4})$
  $\vee\ (\neg \boldsymbol{x_1} \wedge \boldsymbol{x_2} \wedge \boldsymbol{x_3} \wedge \boldsymbol{x_4})$
  $\vee\ (\boldsymbol{x_1} \wedge \boldsymbol{x_2} \wedge \neg \boldsymbol{x_3} \wedge \boldsymbol{x_4})$
- Si $y = 4$ alors : $\boldsymbol{x_1} \wedge \boldsymbol{x_2} \wedge\ \boldsymbol{x_3} \wedge\ \boldsymbol{x_4}$

Dans le cas où Y=0 et ou Y=4, leurs FND sont aussi des FNC. Donc :

Pour Y=0, FNC = FND = $\overline{x1}.\overline{x2}.\overline{x3}.\overline{x4}$

Pour Y=4, FNC = FND = $x1.x2.x3.x4$

Dans les cas où Y=1, Y=2 et Y=3, il va nous falloir transformer les FND obtenues précédemment en FNC. Afin d'effectuer cette conversion, nous utilisons l'algorithme de transformation linéaire.

Pour Y=1 :

Nous savons que la somme de monômes qui lui correspond est :

$$(\boldsymbol{x1}.\overline{\boldsymbol{x2}}.\overline{\boldsymbol{x3}}.\overline{\boldsymbol{x4}}) + (\overline{\boldsymbol{x1}}.\boldsymbol{x2}.\overline{\boldsymbol{x3}}.\overline{\boldsymbol{x4}}) + (\overline{\boldsymbol{x1}}.\overline{\boldsymbol{x2}}.\boldsymbol{x3}.\overline{\boldsymbol{x4}}) + (\overline{\boldsymbol{x1}}.\overline{\boldsymbol{x2}}.\overline{\boldsymbol{x3}}.\boldsymbol{x4})$$

En appliquant une transformation linéaire à cette FND, nous obtenons la FNC suivante :

$(\boldsymbol{z1} + \boldsymbol{z2} + \boldsymbol{z3} + \boldsymbol{z4})$<= Clause contenant les variables introduites.

$.(\boldsymbol{z1} + \boldsymbol{x1}).(\boldsymbol{z1} + \overline{\boldsymbol{x2}}).(\boldsymbol{z1} + \overline{\boldsymbol{x3}}).(\boldsymbol{z1} + \overline{\boldsymbol{x4}})$<= transformation de la première clause.

$.(\boldsymbol{z2} + \overline{\boldsymbol{x1}}).(\boldsymbol{z1} + \boldsymbol{x2}).(\boldsymbol{z2} + \overline{\boldsymbol{x3}}).(\boldsymbol{z2} + \overline{\boldsymbol{x4}})$<= transformation de la deuxième clause.

$.(\boldsymbol{z3} + \overline{\boldsymbol{x1}}).(\boldsymbol{z3} + \overline{\boldsymbol{x2}}).(\boldsymbol{z3} + \boldsymbol{x3}).(\boldsymbol{z3} + \overline{\boldsymbol{x4}})$<= transformation de la troisième clause.

$.(\boldsymbol{z4} + \overline{\boldsymbol{x1}}).(\boldsymbol{z4} + \overline{\boldsymbol{x2}}).(\boldsymbol{z4} + \overline{\boldsymbol{x3}}).(\boldsymbol{z4} + \boldsymbol{x4})$<= transformation de la quatrième clause.

Pour Y=2 :

Nous savons que la somme de monômes qui lui correspond est :

$$(x1.x2.\overline{x3}.\overline{x4}) + (x1.\overline{x2}.x3.\overline{x4}) + (x1.\overline{x2}.\overline{x3}.x4)$$

$$+(\overline{x1}.x2.x3.\overline{x4}) + (\overline{x1}.x2.\overline{x3}.x4) + (\overline{x1}.\overline{x2}.x3.x4)$$

En appliquant une transformation linéaire à cette FND, nous obtenons la FNC suivante :

$(z1 + z2 + z3 + z4 + z5 + z6)$<= Clause contenant les variables introduites.

$.(z1 + x1).(z1 + x2).(z1 + \overline{x3}).(z1 + \overline{x4})$<= transformation de la première clause.

$.(z2 + x1).(z2 + \overline{x2}).(z2 + x3).(z2 + \overline{x4})$<= transformation de la deuxième clause.

$.(z3 + x1).(z3 + \overline{x2}).(z3 + \overline{x3}).(z3 + x4)$<= transformation de la troisième clause.

$.(z4 + \overline{x1}).(z4 + x2).(z4 + x3).(z4 + \overline{x4})$<= transformation de la quatrième clause.

$.(z5 + \overline{x1}).(z5 + x2).(z5 + \overline{x3}).(z5 + x4)$<= transformation de la cinquième clause.

$.(z6 + \overline{x1}).(z6 + \overline{x2}).(z6 + x3).(z6 + x4)$<= transformation de la sixième clause.

Pour Y=3 :

Nous savons que la somme de monômes qui lui correspond est :

$$(x1.x2.x3.\overline{x4}) + (x1.x2.\overline{x3}.x4) + (x1.\overline{x2}.x3.x4) + (\overline{x1}.x2.x3.x4)$$

En appliquant une transformation linéaire à cette FND, nous obtenons la FNC suivante :

$(z1 + z2 + z3 + z4)$<= Clause contenant les variables introduites.

$.(z1 + x1).(z1 + x2).(z1 + x3).(z1 + \overline{x4})$<= transformation de la première clause.

$.(z2 + x1).(z2 + x2).(z2 + \overline{x3}).(z2 + x4)$<= transformation de la deuxième clause.

$.(z3 + x1).(z3 + \overline{x2}).(z3 + x3).(z3 + x4)$<= transformation de la troisième clause.

$.(z4 + \overline{x1}).(z4 + x2).(z4 + x3).(z4 + x4)$<= transformation de la quatrième clause.

Formalisations à partir des contres-modèles

Nous procédons comme dans le chapitre précédent, cependant, nous cherchons cette fois les contre-modèles (vérifiés par les tables de vérités en annexes) sous une forme normale disjonctive. Une fois obtenues, nous les transformerons en forme normale conjonctive.

- Si $y = 1$ alors :

$$\begin{aligned}\neg((&x_1 \wedge x_2 \wedge x_3 \wedge x_4)\\ \vee\ &(x_1 \wedge x_2 \wedge \neg x_3 \wedge \neg x_4)\\ \vee\ &(x_1 \wedge \neg x_2 \wedge x_3 \wedge \neg x_4)\\ \vee\ &(x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge x_4)\\ \vee\ &(x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge \neg x_4)\\ \vee\ &(\neg x_1 \wedge x_2 \wedge x_3 \wedge \neg x_4)\\ \vee\ &(\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4)\\ \vee\ &(\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge \neg x_4)\\ \vee\ &(\neg x_1 \wedge \neg x_2 \wedge x_3 \wedge x_4)\\ \vee\ &(\neg x_1 \wedge \neg x_2 \wedge x_3 \wedge \neg x_4)\\ \vee\ &(\neg x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge x_4)\\ \vee\ &(\neg x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge \neg x_4))\end{aligned}$$

- Si $y = 2$ alors :

$$\begin{aligned}\neg((&x_1 \wedge x_2 \wedge x_3 \wedge x_4)\\ \vee\ &(x_1 \wedge x_2 \wedge x_3 \wedge \neg x_4)\\ \vee\ &(x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4)\\ \vee\ &(x_1 \wedge \neg x_2 \wedge x_3 \wedge x_4)\\ \vee\ &(x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge \neg x_4)\\ \vee\ &(\neg x_1 \wedge x_2 \wedge x_3 \wedge x_4)\\ \vee\ &(\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge \neg x_4)\\ \vee\ &(\neg x_1 \wedge \neg x_2 \wedge x_3 \wedge \neg x_4)\\ \vee\ &(\neg x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge x_4)\\ \vee\ &(\neg x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge \neg x_4))\end{aligned}$$

- Si $y = 3$ alors :

$$\begin{aligned}\neg((&x_1 \wedge x_2 \wedge x_3 \wedge x_4)\\ \vee\ &(x_1 \wedge x_2 \wedge x_3 \wedge \neg x_4)\\ \vee\ &(x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4)\\ \vee\ &(x_1 \wedge x_2 \wedge \neg x_3 \wedge \neg x_4)\\ \vee\ &(x_1 \wedge \neg x_2 \wedge x_3 \wedge x_4)\\ \vee\ &(x_1 \wedge \neg x_2 \wedge x_3 \wedge \neg x_4)\\ \vee\ &(x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge x_4)\\ \vee\ &(\neg x_1 \wedge x_2 \wedge x_3 \wedge x_4)\\ \vee\ &(\neg x_1 \wedge x_2 \wedge x_3 \wedge \neg x_4)\\ \vee\ &(\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4)\\ \vee\ &(\neg x_1 \wedge \neg x_2 \wedge x_3 \wedge x_4)\\ \vee\ &(\neg x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge \neg x_4))\end{aligned}$$

Après la transformation en forme normale conjonctive, nous obtenons :

(Ces transformations sont détaillées dans les annexes)

Pour y = 0 : -X1 . -X2 . -X3 . -X4

Pour y = 1 :
(-X1 + -X2 + -X3 + -X4)
. (-X1 + -X2 + X3 + X4)
. (-X1 + X2 + -X3 + X4)
. (-X1 + X2 + X3 + -X4)
. (-X1 + X2 + X3 + X4)
. (X1 + -X2 + -X3 + X4)
. (X1 + -X2 + X3 + -X4)
. (X1 + -X2 + X3 + X4)
. (X1 + X2 + -X3 + -X4)
. (X1 + X2 + -X3 + X4)
. (X1 + X2 + X3 + -X4)
. (X1 + X2 + X3 + X4)

Pour y = 2 :
(-X1 + -X2 + -X3 + -X4)
. (-X1 + -X2 + -X3 + X4)
. (-X1 + -X2 + X3 + -X4)
. (-X1 + X2 + -X3 + -X4)
. (-X1 + X2 + X3 + X4)
. (X1 + -X2 + -X3 + -X4)
. (X1 + -X2 + X3 + X4)
. (X1 + X2 + -X3 + X4)
. (X1 + X2 + X3 + -X4)
. (X1 + X2 + X3 + X4)

Pour y = 3 :
(-X1 + -X2 + -X3 + -X4)
. (-X1 + -X2 + -X3 + X4)
. (-X1 + -X2 + X3 + -X4)
. (-X1 + -X2 + X3 + X4)
. (-X1 + X2 + -X3 + -X4)
. (-X1 + X2 + -X3 + X4)
. (-X1 + X2 + X3 + -X4)
. (X1 + -X2 + -X3 + -X4)
. (X1 + -X2 + -X3 + X4)
. (X1 + -X2 + X3 + -X4)
. (X1 + X2 + -X3 + -X4)
. (X1 + X2 + X3 + X4)

Pour y = 4 : X1 . X2 . X3 . X4

Choix de la formalisation à utiliser

Nous avons pu constater que la formalisation du problème en partant des modèles ajoutait des variables supplémentaires (imposées par l'algorithme de transformation linéaire) et le nombre de clauses engendrées est important (jusqu'à 21 pour y=2). En revanche, La formalisation par contre-modèle n'ajoute aucunes variables et son nombre de clauses maximum (12 atteint pour y=1 et y=3) est bien inférieur à celui de la solution précédente.

C'est donc l'ensemble de clauses formalisées à partir du contre-modèle que nous utiliserons lors de la partie programmation de ce projet.

Algorithme de transformations d'une grille de squaro en un problème SAT

Maintenant que nous pouvons exprimés les clauses qui régissent une case d'une grille de squaro, nous allons voir comment générer de manière procédurale l'ensemble de clauses qui régissent l'ensemble de la grille. Pour ce faire, nous utilisons l'algorithme suivant. Il est à noter que les deux formalisations mentionnées plus haut sont compatibles avec cet algorithme.

Ensemble-de-clauses = ∅
FNC-régissant-la-case = ∅

Pour chacune des 9 **valeurs** {
    FNC-régissant-la-case = la FNC qui lui correspond (voir paragraphe précédent) selon la valeur de la case traitée (0, 1, 2, 3 ou 4).
    FNC-régissant-la-case = FNC-régissant-la-case où les « x1 » ont été remplacés par l'identifiant du rond positionné en haut à gauche de la valeur traitée.
    FNC-régissant-la-case = FNC-régissant-la-case où les « x2 » ont été remplacés par l'identifiant du rond positionné en haut à droite de la valeur traitée.
    FNC-régissant-la-case = FNC-régissant-la-case où les « x3 » ont été remplacés par l'identifiant du rond positionné en bas à gauche de la valeur traitée.
    FNC-régissant-la-case = FNC-régissant-la-case où les « x4 » ont été remplacés par l'identifiant du rond positionné en bas à droite de la valeur traitée.

    Ensemble-de-clauses = **Ensemble-de-clauses** + **FNC-régissant-la-case**
}
// Ici, la résolution de 'Ensemble-de-clauses' donne la solution (si elle existe) de la grille vide.
// La partie suivante permet de tenir compte des ronds déjà remplis.
// Cet algorithme ne prend pas en compte les contraintes sur les ronds vides.

Pour chaque **rond** déjà remplis dans la grille {
    Ensemble-de-clauses = **Ensemble-de-clauses** + la clause contenant uniquement l'identifiant du rond traité.
}
Retourne Ensemble-de-clauses

Représentation d'une grille de squaro de taille 3x3 valeurs.							
Rond 1		Rond 2		Rond 3		Rond 4	
	Valeur 1		**Valeur 2**		**Valeur 3**		
Rond 5		Rond 6		Rond 7		Rond 8	
	Valeur 4		**Valeur 5**		**Valeur 6**		
Rond 9		Rond 10		Rond 11		Rond 12	
	Valeur 7		**Valeur 8**		**Valeur 9**		
Rond 13		Rond 14		Rond 15		Rond 16	

Prendre en compte des ronds imposés

Cette option propose de prendre en compte les ronds qui ont été rempli (ou imposé) par l'utilisateur. Si un rond est rempli, alors la solution proposée par le SAT-Solveur devra contenir la variable correspondant au rond remplis à vrais.

Grâce à cette option, nous pourrons, par exemple, vérifier que si l'utilisateur remplis un des ronds adjacents à une case de valeur 0, le SAT-Solveur retournera : unsolvable.

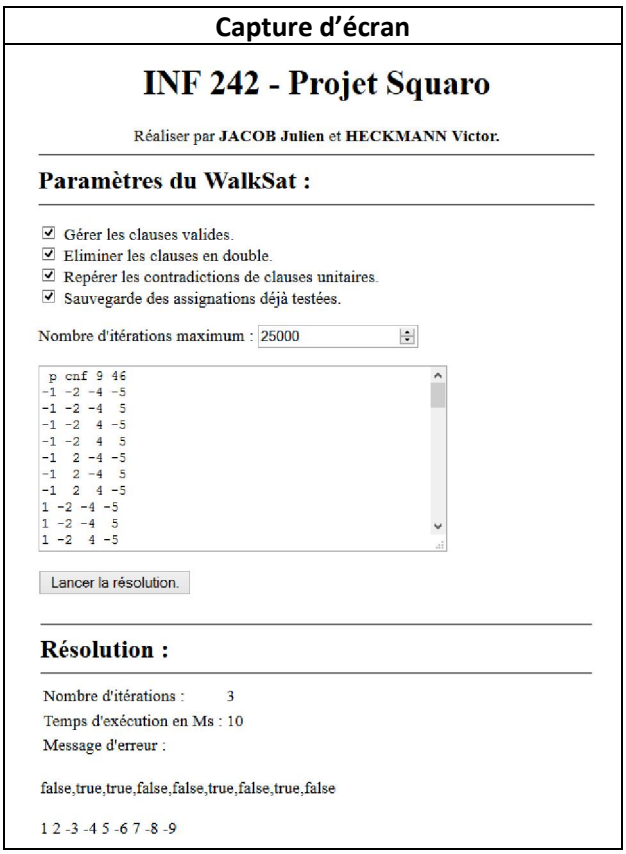
Le WalkSat –SAT-Solveur du projet

Introduction

Le WalkSat est un SAT Solveur qui, pour une assignation créée aléatoirement, prend un ensemble de clause donné en paramètre, il choisit ensuite une clause invalide de cet ensemble de clause par rapport à l'assignation courante et prend une variable soit au hasard de cette clause soit de façon déterministe pour en inverser sa valeur dans l'assignation.

Il répète cet ensemble d'instruction jusqu'à ce qu'il trouve une assignation qui est modèle à l'ensemble de clause ou jusqu'à ce que le nombre d'itération atteint un certain nombre définit au préalable.

Capture d'écran



Choix de variable déterministe utilisé

Pour choisir une variable d'une clause de façon déterministe, un score est calculé pour chaque variable de la clause. Le score d'un variable se définit de la façon suivante :

- Si la valeur de cette variable est égale à **vrai** dans l'assignation courant alors son score sera le nombre de clause où cette variable apparaît négativement moins le nombre de clause où elle apparaît positivement.
- Si sa valeur est égale à **faux**dans l'assignation courant alors son score sera le nombre de clause où cette variable apparaît positivement moins le nombre de clause où elle apparaît négativement.

La variable choisit sera celle qui aura le score le plus élevé.

Les améliorations apportées

Après avoir développé le WalkSAT et implémenté le choix de variable déterministe détaillé ci-dessus, nous lui avons ajouté quelques améliorations.

Gestion des clauses unitaires

La gestion des clauses unitaires intervient plusieurs fois dans les algorithmes utilisés dans le développement du **WalkerTexSATRanger**.

Cette amélioration permet, lors de la création aléatoire de l'assignation, de déterminer la valeur d'une variable si elle apparait dans une clause unitaire.

Elle empêche de prendre une clause unitaire lors du tirage d'une clause aléatoire.

Elle permet aussi de ne pas tirer une variable contenu dans une clause unitaire lors du choix déterministe d'une variable.

Contradiction de clauses unitaires

La détection de contradiction entre deux clauses unitaires permet de savoir de façon très rapide si l'ensemble de clause est satisfaisable ou non lors de la création de l'assignation.

Si la variable d'une clause unitaire '*c*' possède déjà une valeur dans l'assignation et que cette valeur est différente de celle qu'elle devrait avoir pour rendre la clause '*c*' valide, alors l'ensemble est insatisfaisable.

Sauvegarde des assignations

La sauvegarde des assignations déjà testé permet, lors du choix déterministe d'une variable, de renvoyer directement une variable si l'inversement de sa valeur dans l'assignation n'as pas encore été essayé.

Elimination des clauses en doubles

L'élimination des clauses en doubles permet de réduire le nombre d'accès au tableau contenant les clauses, ce qui a pour effet de réduire légèrement le temps d'exécution de certaines fonctions.

Mise en œuvre logiciel

Un objet Squaro

Afin de pouvoir mettre pratique les formalisations et les algorithmes précédents, nous devons disposer d'un moyen pour créer et manipuler des grilles de squaro. Pour ce faire, nous utilisons la programmation orientée objets proposée par le JavaScript afin de créer une classe « ScuaroGrid » qui représente une grille de squaro.

Le constructeur de la classe s'initialise avec les paramètres suivants :

- Largeur en nombre de cases de la grille à générer
- hauteur en nombre de cases de la grille à générer
- une chaine de caractère contenant le nom de l'objet tel qu'il est initialisé (utile pour générer des grilles graphiques pouvant appeler l'objet)
- Booléen indiquant si la grille à générer doit avoir une solution ou si elle doit être totalement aléatoire.

L'objet dispose des méthodes nécessaires pour créer des grilles de tailles variables, solvables ou insolvables. Retourner des informations sur la grille (nombre de cases, sommes et distributions des valeurs, etc). Retourner un fichier dimacs régissant la grille en prenant en compte ou non les ronds remplis par l'utilisateur. Afficher une grille automatiquement colorée (rouge ou vert) selon l'état de la case dans une balise d'un fichier HTML.

Découpage du projet

Trois pages sont destinées à l'utilisateur :

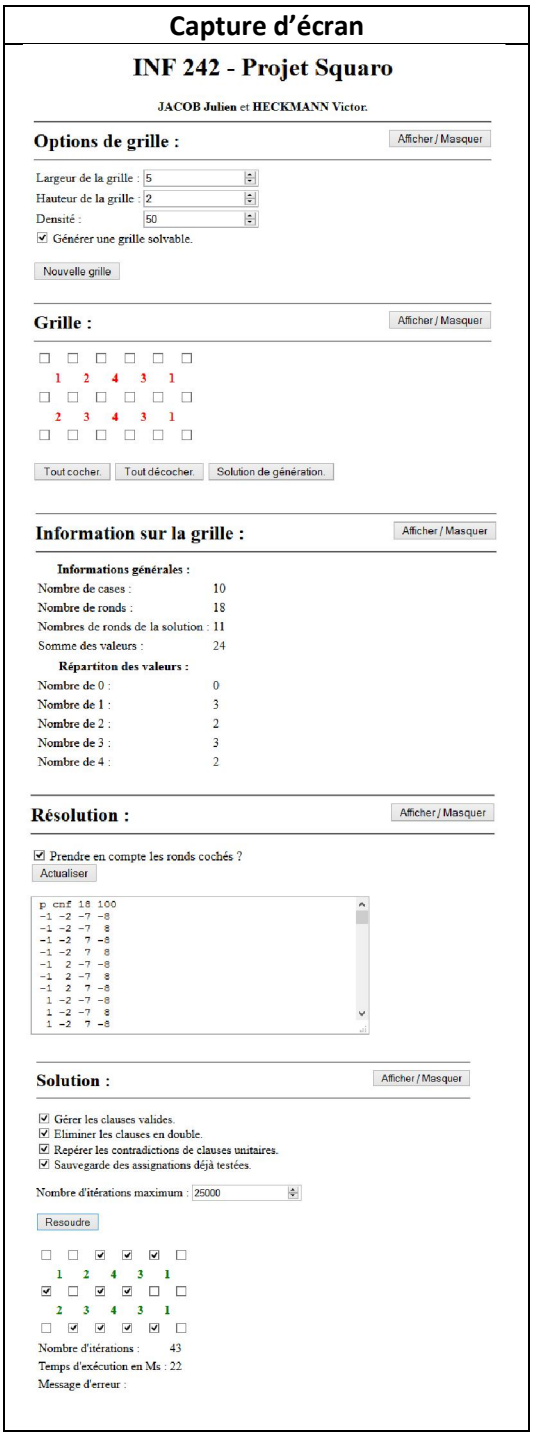
- **Index.html :** Menu vers les autres pages.
- **Analysis.html** : Programme d'analyse d'une grille de squaro.
- **MetaAnalysis.html** : Programme d'analyse d'une série de grilles de squaro.

Les fichiers de codes JavaScript sont contenus dans le dossier : /script

Analyse d'une grille de squaro

Cette partie du programme permet de générer, afficher, manipuler, observer et résoudre une grille de squaro.

Capture d'écran



Générer une grille avec les options suivantes :

- Largeur : Nombre de cases.
- Hauteur : Nombre de cases.
- Densité : Compris entre 0 et 100, plus la valeur est grande, plus la somme des valeurs de la grille sera grande, et inversement.
- Solvable : si oui, la grille aura forcément une solution et le programme pourra l'afficher. Sinon, la grille est totalement aléatoire.

Affichage de la grille :

- Affichage de la grille (les ronds de la grille peuvent être remplis ou vidés par l'utilisateur).
- Les valeurs de la grille s'affichent en vert si la case est correctement remplie.
- L'utilisateur peut : Remplir ou vider tous les ronds en 1 clic.
- L'utilisateur peut : Afficher la solution générée lors de la génération si celle-ci est valide.

Les informations sur la grille :

- Nombre de cases
- Nombre de ronds
- Nombres de ronds de la solution
- Somme des valeurs
- Répartition des valeurs de la grille

Le fichier Dimacs :

- Affiche le fichier dimacs.
- Permet de prendre ou non en compte les ronds remplis par l'utilisateur.

Le walksat :

- Option 1 : Gérer les clauses valides.
- Option 2 : Eliminer les clauses en doubles.
- Option 3 : Repérer les contradictions de clauses unitaires.
- Option 4 : Sauvegarde des assignations déjà testées.
- Nombre d'itération maximum.

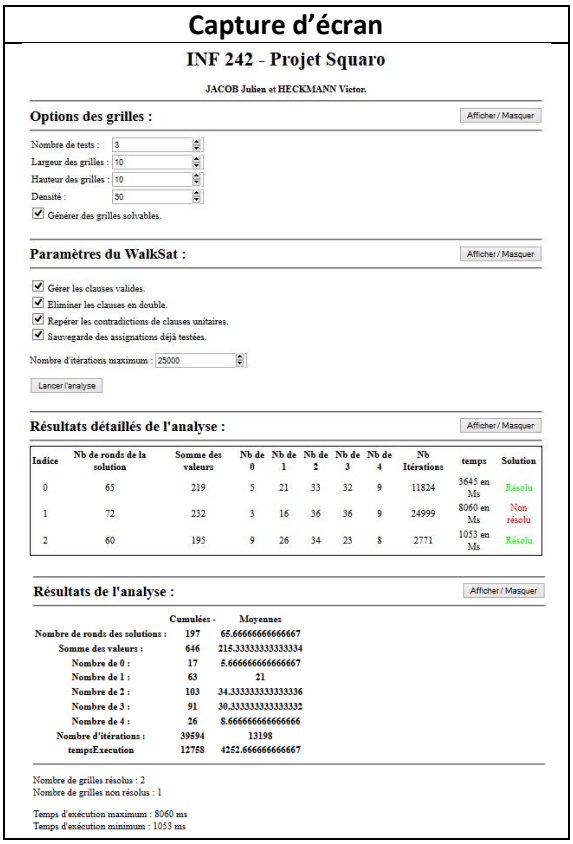
Résultat du WalkSat :

- Affiche le temps d'exécution.
- Affiche le nombre d'itération.
- Affiche un message d'erreur si celui-ci est renseigné.
- Si la résolution par le WalkSat échoue, l'utilisateur est averti par un message. Sinon, affiche une grille rempli selon le retour du WalkSat .

Analyse d'une série de grille de squaro

Cette partie du programme a pour but de permettre l'analyse d'un grand nombre de grilles. Après avoir défini les options pour la génération de grille et celle du WalkSat, le programme affiche les informations sur les grilles et sur leurs résolutions par le WalkSat.

Capture d'écran



Options des grilles

- Nombre de tests
- Largeur : Nombre de cases.
- Hauteur : Nombre de cases.
- Densité : Compris entre 0 et 100, plus la valeur est grande, plus la somme des valeurs de la grille sera grande, et inversement.
- Solvable : si oui, la grille aura forcément une solution et le programme pourra l'afficher. Sinon, la grille est totalement aléatoire.

Paramètres du WalkSat :

- Option 1 : Gérer les clauses valides.
- Option 2 : Eliminer les clauses en doubles.
- Option 3 : Repérer les contradictions de clauses unitaires.
- Option 4 : Sauvegarde des assignations déjà testées.
- Nombre d'itération maximum.

Résultats détaillés :

- Affiche un tableau contenant pour chaque grille :
    - Nombre de cases
    - Nombre de ronds
    - Nombres de ronds de la solution
    - Somme des valeurs
    - Répartition des valeurs de la grille
    - Nombre d'itération du WalkSat.
    - Temps d'exécution du WalkSat.
    - Résolu ou Non-Résolu selon le retour du WalkSat pour la grille concernée.

Résultats globaux :

- Nombre de cases cumulées et valeurs moyennes.
- Nombre de ronds cumulés et valeurs moyennes.
- Nombres de ronds de la solution cumulés et valeurs moyennes.
- Somme des valeurs cumulées et valeurs moyennes.
- Répartition des valeurs cumulées et valeurs moyennes.
    - Nombre d'itération du WalkSat cumulées et valeurs moyennes.
    - Temps d'exécution du WalkSat cumulés et valeurs moyennes.
    - Nombre de grilles résolues et non résolus.
    - Temps d'exécution maximum et minimum.

Annexes

Table de vérité vérifiant les FND permettant d'établir les modèles et contre-modèles.

X1	X2	X3	X4	Y=0	Y=1	Y=2	Y=3	Y=4
0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	0
0	0	1	0	0	1	0	0	0
0	0	1	1	0	0	1	0	0
0	1	0	0	0	1	0	0	0
0	1	0	1	0	0	1	0	0
0	1	1	0	0	0	1	0	0
0	1	1	1	0	0	0	1	0
1	0	0	0	0	1	0	0	0
1	0	0	1	0	0	1	0	0
1	0	1	0	0	0	1	0	0
1	0	1	1	0	0	0	1	0
1	1	0	0	0	0	1	0	0
1	1	0	1	0	0	0	1	0
1	1	1	0	0	0	0	1	0
1	1	1	1	0	0	0		1

Détailles de la transformation des contre-modèles en forme normale conjonctive.

```
################################################
# Liste des possibilités :
################################################
        ( X1   . X2    . X3    . X4 )
      + ( X1   . X2    . X3    . -X4 )
      + ( X1   . X2    . -X3   . X4 )
      + ( X1   . X2    . -X3   . -X4 )
      + ( X1   . -X2   . X3    . X4 )
      + ( X1   . -X2   . X3    . -X4 )
      + ( X1   . -X2   . -X3   . X4 )
      + ( X1   . -X2   . -X3   . -X4 )
      + ( -X1  . X2    . X3    . X4 )
      + ( -X1  . X2    . X3    . -X4 )
      + ( -X1  . X2    . -X3   . X4 )
      + ( -X1  . X2    . -X3   . -X4 )
      + ( -X1  . -X2   . X3    . X4 )
      + ( -X1  . -X2   . X3    . -X4 )
      + ( -X1  . -X2   . -X3   . X4 )
```

```
        + ( -X1  . -X2    . -X3    . -X4 )
##############################################
# Modele pour Y=0 :
##############################################
                -X1      . -X2    . -X3    . -X4
##############################################
# Modele pour Y=1 :
##############################################
                -((  X1   . X2     . X3     . X4 )


                + (  X1   . X2     . -X3    . -X4 )

                + (  X1   . -X2    .  X3    . -X4 )
                + (  X1   . -X2    . -X3    .  X4 )
                + (  X1   . -X2    . -X3    . -X4 )

                + ( -X1   .  X2    .  X3    . -X4 )
                + ( -X1   .  X2    . -X3    .  X4 )
                + ( -X1   .  X2    . -X3    . -X4 )
                + ( -X1   . -X2    .  X3    .  X4 )
                + ( -X1   . -X2    .  X3    . -X4 )
                + ( -X1   . -X2    . -X3    .  X4 )
                + ( -X1   . -X2    . -X3    . -X4 )   )
##########################################
                 -(  X1   . X2     . X3     . X4 )
                . -(  X1   . X2    . -X3    . -X4 )
                . -(  X1   . -X2   .  X3    . -X4 )
                . -(  X1   . -X2   . -X3    .  X4 )
                . -(  X1   . -X2   . -X3    . -X4 )
                . -( -X1   .  X2   .  X3    . -X4 )
                . -( -X1   .  X2   . -X3    .  X4 )
                . -( -X1   .  X2   . -X3    . -X4 )
                . -( -X1   . -X2   .  X3    .  X4 )
                . -( -X1   . -X2   .  X3    . -X4 )
                . -( -X1   . -X2   . -X3    .  X4 )
                . -( -X1   . -X2   . -X3    . -X4 )
##########################################
                 ( - X1   + - X2   + - X3   + - X4 )
                . ( - X1   + - X2   + --X3   + --X4 )
                . ( - X1   + --X2   + - X3   + --X4 )
                . ( - X1   + --X2   + --X3   + - X4 )
                . ( - X1   + --X2   + --X3   + --X4 )
                . ( --X1   + - X2   + - X3   + --X4 )
                . ( --X1   + - X2   + --X3   + - X4 )
```

```
                . ( --X1  + - X2   + --X3   + --X4 )
                . ( --X1  + --X2   + - X3   + - X4 )
                . ( --X1  + --X2   + - X3   + --X4 )
                . ( --X1  + --X2   + --X3   + - X4 )
                . ( --X1  + --X2   + --X3   + --X4 )
##########################################
                  ( -X1   + -X2    + -X3    + -X4 )
                . ( -X1   + -X2    +  X3    +  X4 )
                . ( -X1   +  X2    + -X3    +  X4 )
                . ( -X1   +  X2    +  X3    + -X4 )
                . ( -X1   +  X2    +  X3    +  X4 )
                . (  X1   + -X2    + -X3    +  X4 )
                . (  X1   + -X2    +  X3    + -X4 )
                . (  X1   + -X2    +  X3    +  X4 )
                . (  X1   +  X2    + -X3    + -X4 )
                . (  X1   +  X2    + -X3    +  X4 )
                . (  X1   +  X2    +  X3    + -X4 )
                . (  X1   +  X2    +  X3    +  X4 )
##############################################
# Modele pour Y=2 :
##############################################
                -((  X1   .  X2     .  X3     .  X4 )
                + (  X1   .  X2     .  X3     . -X4 )
                + (  X1   .  X2     . -X3     .  X4 )

                + (  X1   . -X2     .  X3     .  X4 )


                + (  X1   . -X2     . -X3     . -X4 )
                + ( -X1   .  X2     .  X3     .  X4 )


                + ( -X1   .  X2     . -X3     . -X4 )

                + ( -X1   . -X2     .  X3     . -X4 )
                + ( -X1   . -X2     . -X3     .  X4 )
                + ( -X1   . -X2     . -X3     . -X4 )   )
##########################################
                  -(  X1   .  X2     .  X3     .  X4 )
                . -(  X1   .  X2     .  X3     . -X4 )
                . -(  X1   .  X2     . -X3     .  X4 )
                . -(  X1   . -X2     .  X3     .  X4 )
                . -(  X1   . -X2     . -X3     . -X4 )
                . -( -X1   .  X2     .  X3     .  X4 )
                . -( -X1   .  X2     . -X3     . -X4 )
```

```
                . -( -X1  . -X2     .  X3     . -X4 )
                . -( -X1  . -X2     . -X3     .  X4 )
                . -( -X1  . -X2     . -X3     . -X4 )
########################################
                 ( - X1   + - X2    + - X3    + - X4 )
                . ( - X1   + - X2    + - X3    + --X4 )
                . ( - X1   + - X2    + --X3    + - X4 )
                . ( - X1   + --X2    + - X3    + - X4 )
                . ( - X1   + --X2    + --X3    + --X4 )
                . ( --X1   + - X2    + - X3    + - X4 )
                . ( --X1   + - X2    + --X3    + --X4 )
                . ( --X1   + --X2    + - X3    + --X4 )
                . ( --X1   + --X2    + --X3    + - X4 )
                . ( --X1   + --X2    + --X3    + --X4 )
########################################
                 ( -X1   + -X2    + -X3    + -X4 )
                . ( -X1   + -X2    + -X3    +  X4 )
                . ( -X1   + -X2    +  X3    + -X4 )
                . ( -X1   +  X2    + -X3    + -X4 )
                . ( -X1   +  X2    +  X3    +  X4 )
                . (  X1   + -X2    + -X3    + -X4 )
                . (  X1   + -X2    +  X3    +  X4 )
                . (  X1   +  X2    + -X3    +  X4 )
                . (  X1   +  X2    +  X3    + -X4 )
                . (  X1   +  X2    +  X3    +  X4 )
##############################################
##############################################
# Modele pour Y=3 :
##############################################
                -((  X1   .  X2     .  X3     .  X4 )
                + (  X1   .  X2     .  X3     . -X4 )
                + (  X1   .  X2     . -X3     .  X4 )
                + (  X1   .  X2     . -X3     . -X4 )
                + (  X1   . -X2     .  X3     .  X4 )
                + (  X1   . -X2     .  X3     . -X4 )
                + (  X1   . -X2     . -X3     .  X4 )

                + ( -X1   .  X2     .  X3     .  X4 )
                + ( -X1   .  X2     .  X3     . -X4 )
                + ( -X1   .  X2     . -X3     .  X4 )

                + ( -X1   . -X2     .  X3     .  X4 )


                + ( -X1   . -X2     . -X3     . -X4 )    )
```

```
########################################
                -( X1  . X2    . X3    . X4 )
               . -( X1  . X2    . X3    . -X4 )
               . -( X1  . X2    . -X3   . X4 )
               . -( X1  . X2    . -X3   . -X4 )
               . -( X1  . -X2   . X3    . X4 )
               . -( X1  . -X2   . X3    . -X4 )
               . -( X1  . -X2   . -X3   . X4 )
               . -( -X1 . X2    . X3    . X4 )
               . -( -X1 . X2    . X3    . -X4 )
               . -( -X1 . X2    . -X3   . X4 )
               . -( -X1 . -X2   . X3    . X4 )
               . -( -X1 . -X2   . -X3   . -X4 )
########################################
                ( - X1  + - X2  + - X3  + - X4 )
               . ( - X1  + - X2  + - X3  + --X4 )
               . ( - X1  + - X2  + --X3  + - X4 )
               . ( - X1  + - X2  + --X3  + --X4 )
               . ( - X1  + --X2  + - X3  + - X4 )
               . ( - X1  + --X2  + - X3  + --X4 )
               . ( - X1  + --X2  + --X3  + - X4 )
               . ( --X1  + - X2  + - X3  + - X4 )
               . ( --X1  + - X2  + - X3  + --X4 )
               . ( --X1  + - X2  + --X3  + - X4 )
               . ( --X1  + --X2  + - X3  + - X4 )
               . ( --X1  + --X2  + --X3  + --X4 )
########################################
                ( -X1  + -X2  + -X3  + -X4 )
               . ( -X1  + -X2  + -X3  + X4 )
               . ( -X1  + -X2  + X3   + -X4 )
               . ( -X1  + -X2  + X3   + X4 )
               . ( -X1  + X2   + -X3  + -X4 )
               . ( -X1  + X2   + -X3  + X4 )
               . ( -X1  + X2   + X3   + -X4 )
               . ( X1   + -X2  + -X3  + -X4 )
               . ( X1   + -X2  + -X3  + X4 )
               . ( X1   + -X2  + X3   + -X4 )
               . ( X1   + X2   + -X3  + -X4 )
               . ( X1   + X2   + X3   + X4 )
############################################
# Modele pour Y=4 :
############################################
               X1      . X2    . X3    . X4
```