Blin Sébastien
Collin Pierre-Henri

École supérieure d'ingénieurs de Rennes

1ère Année
Parcours Informatique

Algorithmie et complexité

Compte-Rendu TP1

Sous l'encadrement de :

Ridoux Olivier
Maurel Pierre

1 Problème des N reines

1.1 Objectif

Dans la seconde partie de ce TP, nous devions utiliser les ROBDD pour pouvoir résoudre un problème. Le problème consiste à trouver une combinaison de cases où placer N dames sur un plateau de taille NxN pour ne pas qu'elles puissent se menacer mutuellement. Donc qu'il y en ait une par colonne, une par ligne, et pas 2 sur la même diagonale.
La solution consiste donc à modéliser une équation booléenne modélisant les conditions en prenant chaque case comme une variable.

1.2 Moyens mis en œuvre

L'expression finale est un ET logique entre 3 sous équations, puisque nous souhaitons avoir une dame par ligne ET une dame par colonne ET au maximum une dame par diagonale.
La première sous-équation décrit le fait qu'il y ait une dame par ligne.

```
for(int i = 0; i < n; ++i)
{
        Expression temp2 = new Constante(false);
        for(int k = 0; k < n; ++k)
        {
                Expression temp = new Constante(true);
                for(int j = 0; j < n; ++j)
                {
                        if(j != k)
                                temp = new Et(new Non(new Atome("x"+ i + j)), temp);
                        else
                                temp = new Et(new Atome("x"+ i + j), temp);
                }
                temp2 = new Ou(temp2, temp);
        }
        expDame = new Et(expDame, temp2);
}
```

Ce qui peut se traduire par $\vee_{(1 \geq i \geq N, 1 \geq j \geq N)}[x_{ij} \Rightarrow \wedge_{(1 \geq l \geq N, l \neq j)} \neg x_{il}]$ avec au moins un x_{ij} vrai.
Même si dans notre code, nous le modélisons à l'aide de Et et de Ou logique.
Pour les colonnes, c'est à peut prêt la même méthode, il suffit d'échanger 2 boucles, on peut considérer le problème comme celui de résoudre une dame par ligne avec un plateau à 90 degrés.

```
for(int j = 0; j < n; ++j)
{
        Expression temp2 = new Constante(false);
        for(int k = 0; k < n; ++k)
        {
                Expression temp = new Constante(true);
                for(int i = 0; i < n; ++i)
                {
                        if(i != k)
                                temp = new Et(new Non(new Atome("x"+ i + j)), temp);
```

```java
                else
                        temp = new Et(new Atome("x"+ i + j), temp);
            }
            temp2 = new Ou(temp2, temp);
        }
        expDame = new Et(expDame, temp2);
}
```

Le traitement des diagonales est un peu différent. Ici, il faut considérer des droites et prendre seulement les cases dans le plateau. L'équation nous donne : $x_{ij} \Rightarrow \wedge_{(1 \geq k \geq N, 1 \geq j+k-i \geq N, k \neq i)} \neg x_{k,j+k-i}$ dans un sens, $x_{ij} \Rightarrow \wedge_{(1 \geq k \geq N, 1 \geq j+i-k \geq N, k \neq i)} \neg x_{k,j+i-k}$ pour l'autre sens.

```java
for(int i = 0; i < n; ++i)
{
        Expression temp2 = new Constante(false);
        for(int j = 0; j < n; ++j)
        {
                Expression temp = new Constante(true);
                for(int k = 0; k < n; ++k)
                {
                        int v = j+k-i;
                        int w = j+i-k;
                        boolean find = false;
                        if(i != k && v < n && v >= 0)
                        {
                                temp = new Et(new Non(new Atome("x" + k + v)), temp);
                                find = true;
                        }
                        if(i!=k && w < n && w >= 0)
                        {
                                temp = new Et(new Non(new Atome("x" + k + w)), temp);
                                find = true;
                        }
                        if(!find)
                                temp = new Et(new Atome("x" + i + j), temp);
                }
                temp2 = new Ou(temp2, temp);
        }
        expDame = new Et(expDame, temp2);
}
```

Ce qui nous permet de modéliser une équation booléenne représentant le fait de n'avoir qu'une dame par ligne, une par colonne et au maximum une par diagonale.
Enfin, pour la partie de l'affichage, il suffit de trouver une solution (première partie de la fonction) en parcourant le ROBDD de l'expression précédement créée et de stocker les cases où les dames sont présentes à true (via un tableau représentant linéairement le plateau). Puis on dessine avec la seconde partie de la fonction.

```java
//Affiche le tableau solution pour le probleme des dames
public void reines_affiche_sat(int n)
{
```

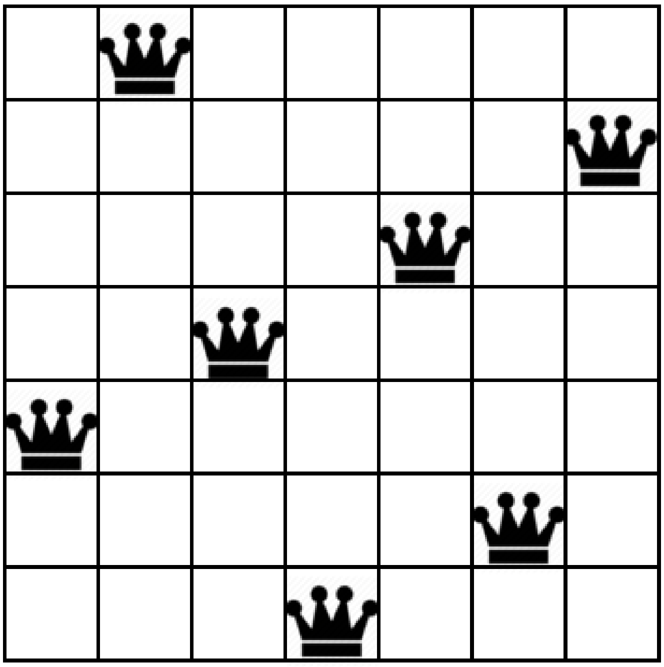
```
		//Le tableau contenant true pour les cases possedant une dame
		boolean[] solution = new boolean[n*n];
		Iterator<Noeud_ROBDD> it = R.iterator();
		int toSearch = 1;
		while(it.hasNext())
		{
			Noeud_ROBDD node = it.next();
			if(node.getIdFilsDroit() == toSearch)
			{
				toSearch = node.getId();
				try
				{
					//Pour savoir quelle variable correspond a quelle case
					Pattern p = Pattern.compile("x([0-9])([0-9])");
					Matcher m = p.matcher(node.getNom());
					m.matches();
					solution[Integer.parseInt(m.group(1))*n+
					Integer.parseInt(m.group(2))] = true;
				}
				catch (Exception e)
				{

				}
			}
			if(node.getIdFilsGauche() == toSearch)
				toSearch = node.getId();
		}

		/**Affichage**/
		for(int i = 0; i < n; ++i)
		{
			for(int j = 0; j < 3*n; ++j)
				System.out.print("_");
			System.out.print("\n");
			for(int j = 0; j < n; ++j)
				if(solution[i*n+j])
					System.out.print("|X|");
				else
					System.out.print("| |");
			System.out.print("\n");
		}
		for(int j = 0; j < 3*n; ++j)
			System.out.print("_");
		System.out.print("\n");
	}
```

1.3 Résultats

Pour un plateau de où N = :



Solution fournie par l'algorithme n=7 reines

- 1 : x00
- 2, 3 : NON SAT
- 4 : x13.x01.x20.x32.
- 5 : x02.x21.x40.x14.x33.
- 6 : x02.x21.x40.x15.x34.x53.
- 7 : x01.x40.x32.x24.x63.x16.x55.
- 8 : x03.x41.x34.x76.x20.x62.x15.x57

Pour N=8, la résolution commence à être longue, mais le ROBDD construit est de taille conséquente.

2 Conclusion

Pour conclure, nous pouvons remarquer que choisir une représentation d'un problème de type SAT avec un ROBDD est une méthode qui fonctionne bien et dont le parcourt afin de trouver une solution est relativement rapide. Par contre, la taille du ROBDD peut très vite exploser.