Modélisation & Résolution

BOUDET Alexandre BOURIAUD Thomas LE QUEC Vincent PERICHET Thomas RAFIKI Younes

Plan

I/ Les n-reines

1) M1 & R1
2) M2 & R2
3) M3 pseudo booléen
4) M3 booléen

II/ Garam

III/ Rikudo

I/ Les n-reines

1) M1

- Variables : $Xi, i \in [1..4]$
  - Reines sur l'échiquier
- Domaines (valeurs possible) : {1,...,n}
  - Index des colonnes de chaque reines placées
- Contraintes (requirements):
  - Une et une seule reine par colonne
    - Déjà dans la formulation des variables et du domaine
  - Une et une seule reine par ligne
    - $\forall i, j \in [1..4], i =! j$
  - Pas deux reines sur une diagonale
    - Voir R1

R1

R1	R2	R3	R4
		Q	
Q			
			Q
	Q		

Chaque reine à sa colonne.

Nous voulons savoir quelle ligne elle occupe.

Nous avons appliqué des contraintes finies pour 4 reines :

```
% ligne
constraint R1 != R2;
constraint R1 != R3;
constraint R1 != R4;
constraint R2 != R3;
constraint R2 != R4;
constraint R3 != R4;
```

```
% Diagonales montantes
constraint R1 != R2 + 1;
constraint R1 != R3 + 2;
constraint R1 != R4 + 3;
constraint R2 != R3 + 1;
constraint R2 != R4 + 2;
constraint R3 != R4 + 1;
```

```
% Diagonales descendantes
constraint R1 != R2 - 1;
constraint R1 != R3 - 2;
constraint R1 != R4 - 3;
constraint R2 != R3 - 1;
constraint R2 != R4 - 2;
constraint R3 != R4 - 1;
```

2) M2

- Variables : q = array[1..n] of var 1..n
  - Reines sur l'échiquier
- Domaines (valeurs possible) : {1,...,n}
  - Index des colonnes de chaque reines placées
- Contraintes (requirements):
  - Une et une seule reine par colonne
    - Déjà dans la formulation des variables et du domaine
  - Une et une seule reine par ligne
    - alldifferent(q)
  - Pas deux reines sur une diagonale
    - Voir R2

R2

```
include "alldifferent.mzn";

int: n;
array[1..n] of var 1..n:q;

% Une et une seule reine par colonne
constraint alldifferent(q);
% Diagonale
constraint alldifferent([q[i] + i | i in 1..n]);
constraint alldifferent([q[i] - i | i in 1..n]);

constraint q[1] <= q[n];

solve satisfy;

output [ if fix(q[j]) == i then "|Q" else "| " endif ++
         if j == n then "|\n" else "" endif | i, j in 1..n];
```

3) M3 pseudo booléen

Première modélisation

```
% lines
constraint forall(i in 1..n)(
    sum(k in 1..n)(bool2int(q[k,i])) = 1
);
```

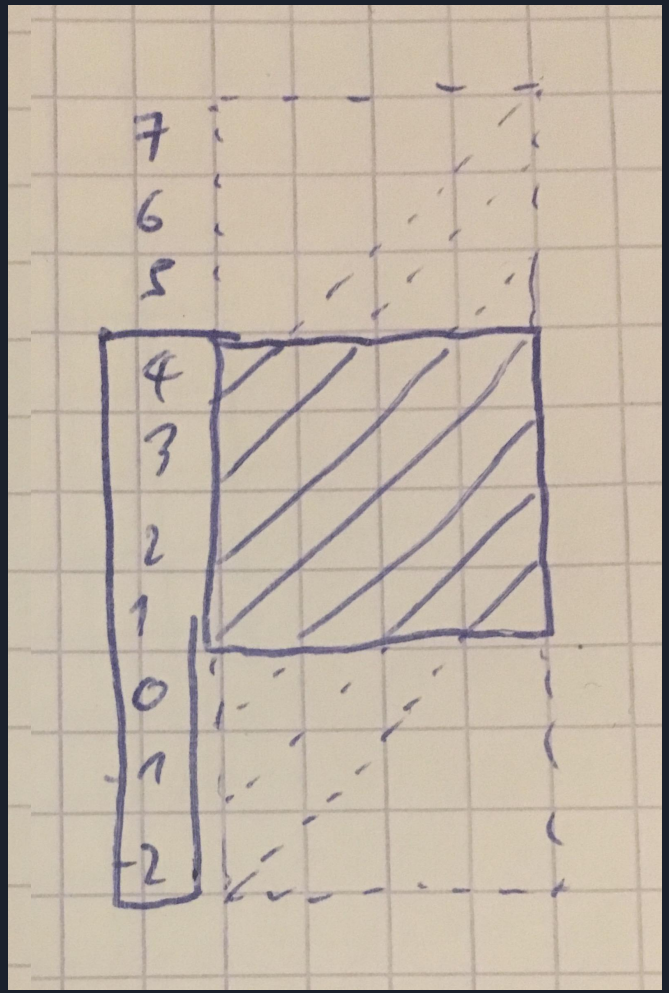
Variables booléennes

Contrainte sur les colonnes

```
% columns
constraint forall(i in 1..n)(
    sum(k in 1..n)(bool2int(q[i,k])) = 1
);
```

3) M3 pseudo booléen

```
% increasing diagonales
% decreasing diagonales
constraint forall(i in 1..(2 * n - 1))(
    sum(k in 1..n)(
        if((i - k + 1 > n) \/ (i - k + 1 < 1)) then 0
        else bool2int(q[i - k + 1, k])
        endif
    ) < 2
);
```



4) M3

P

=

```
% lines
constraint forall(i in 1..n) (
    forall(j in 1..n) (
        orall([q[i, k] | k in 1..n where k != j]) != q[i, j]
    )
);
% columns
constraint forall(i in 1..n) (
    forall(j in 1..n) (
        orall([q[k, j] | k in 1..n where k != i]) != q[i, j]
    )
);
```

lse”

```
function var bool: orall(array[int] of var bool: tab) =
    if length(tab) = 1
    then tab[1]
    else tab[1] \/ orall([tab[i] | i in 1..length(tab) where i > 1])
    endif;
```

4) M3

```
constraint
for i in lines do
        for j in columns do
                if      not out of bounds && queen [ i, j ] == true
                then    orall ( AllOtherCases ) == false
                else    true
                endif
        endfor
endfor
```

Néanmoins, pour ce qui est des diagonales, le travail est plus fastidieux.

Pour chaque case :

```
% decreasing diagonales
constraint forall(i in 1..n) (
    forall(j in 1..n) (
        if q[i, j] = true /\ length([q[i + k, j + k] | k in (-n)..n where i + k > 0 /\ i + k < n /\ j + k > 0 /\ j + k < n /\ k != 0]) != 0
        then orall([q[i + k, j + k] | k in (-n)..n where i + k > 0 /\ i + k < n /\ j + k > 0 /\ j + k < n /\ k != 0]) != true
        else true
        endif
    )
);
```

```
% increasing diagonales
constraint forall(i in 1..n) (
    forall(j in 1..n) (
        if q[i, j] = true /\ length([q[i - k, j + k] | k in (-n)..n where i - k > 0 /\ i - k < n /\ j + k > 0 /\ j + k < n /\ k != 0]) != 0
        then orall([q[i - k, j + k] | k in (-n)..n where i - k > 0 /\ i - k < n /\ j + k > 0 /\ j + k < n /\ k != 0]) != true
        else true
        endif
    )
);
```

II/ Garam

II/ Garam

On travaille sur des indices au début de bas gauche

```
% 1st line
o10 = "-";
o90 = "+";
x100 = 1;

% 2nd line
x01 = 2;
x81 = 2;
x121 = 4;
```

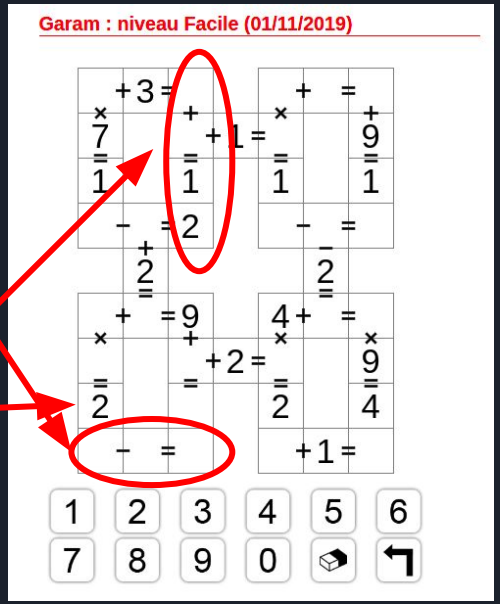
```
numbers = [
    _,_,_,_,1,_,
    2,_,2,4,
    _,_,2,_,9,
    _,_,9,4,_,_,
    2,2,
    _,_,2,_,_,_,
    1,1,1,1,
    7,_,1,_,9,
    _,3,_,_,_,_
];

operators = [
    "-", "+",
    "+",
    "*", "+", "*", "*",
    "+", "+",
    "+", "-",
    "-", "-",
    "+",
    "*", "+", "*", "+",
    "+", "+"
];
```

```
predicate operation(string: operator, var 0..9: x1, var 0..9: x2, var 0..90: x3) =
    (operator = "+" /\ x1 + x2 = x3) \/
    (operator = "-" /\ x1 - x2 = x3) \/
    (operator = "*" /\ x1 * x2 = x3) \/
    (operator = "/" /\ x1 / x2 = x3);
```

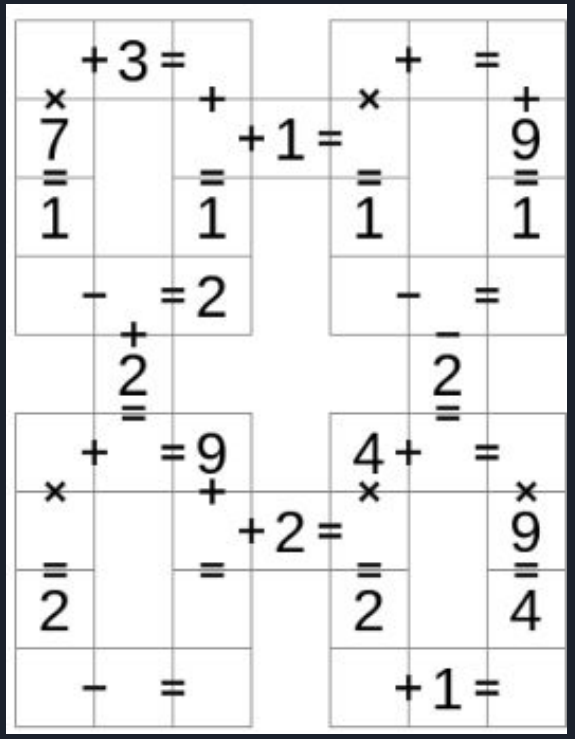
```
constraint operation(operators[1], numbers[1], numbers[2], numbers[3]);
```

```
constraint numbers[7] > 0;
```

```
constraint operation(operators[15], numbers[39], numbers[34], numbers[30] * 10 + numbers[24]);
```

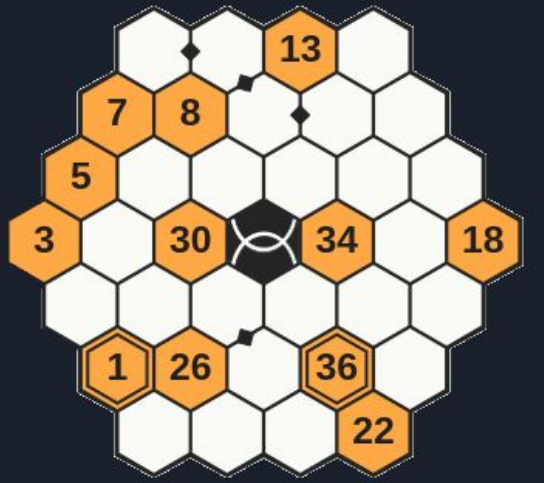


```
2+3=5    2+2=4
*    +   *    +
7    7+1=8    9
=    =   =    =
1    1   1    1
4-2=2    6-3=3
  +          -
  2          2
  =          =
5+4=9    4+1=5
*    +   *    *
5    4+2=6    9
=    =   =    =
2    1   2    4
5-2=3    4+1=5
```



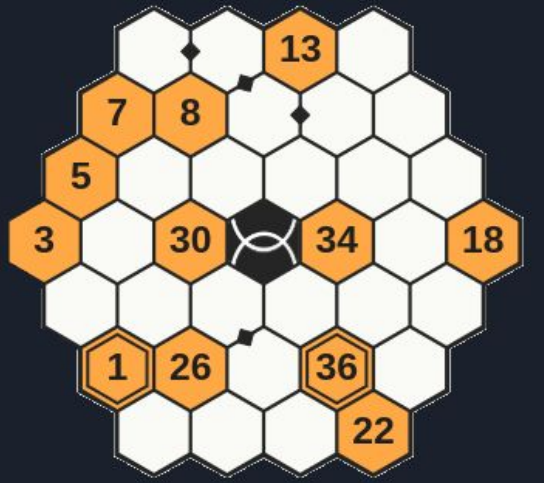
III/ Rikudo

III/ Rikudo



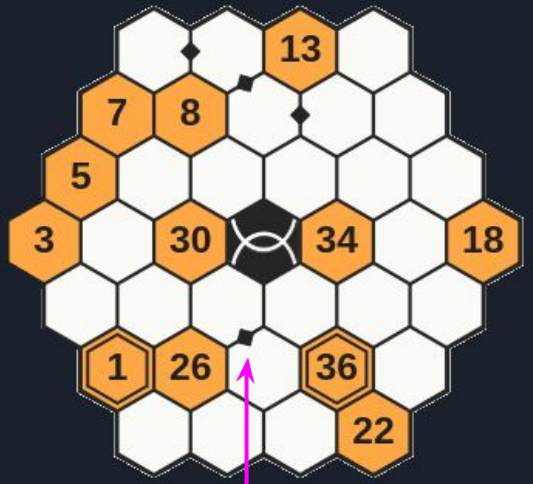
-1	-1	-1	0	-1	0	-1	13	-1	0	-1	-1	-1
-1	-1	7	-1	8	-1	0	-1	0	-1	0	-1	-1
-1	5	-1	0	-1	0	-1	0	-1	0	-1	0	-1
3	-1	0	-1	30	-1	-1	-1	34	-1	0	-1	18
-1	0	-1	0	-1	0	-1	0	-1	0	-1	0	-1
-1	-1	1	-1	26	-1	0	-1	36	-1	0	-1	-1
-1	-1	-1	0	-1	0	-1	0	-1	22	-1	-1	-1

III/ Rikudo



			33		34		35		36			
		28		29		30		31		32		
	22		23		24		25		26		27	
16		17		18				19		20		21
	10		11		12		13		14		15	
		5		6		7		8		9		
			1		2		3		4			

III/ Rikudo

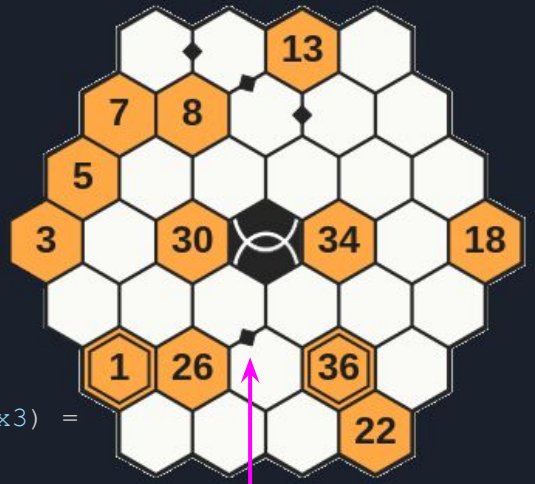


Sens de lecture

```
rikudo = [
    _,_,_,22,
    1,26,_,36,_,
    _,_,_,_,_,_,
    3,_,30,34,_,18,
    5,_,_,_,_,_,
    7,8,_,_,_,
    _,_,13,_
];
```

```
links_size = 4;
links = array2d(1..links_size,1..2,
    [ 7,12,
    30,31,
    33,34,
    34,30
    ]
);
```

III/ Rikudo




```
% for boxes with 3 links
predicate links(var 1..36 : x, var 1..36 : x1, var 1..36 : x2, var 1..36 : x3) =
    x != 36 -> x1 = x + 1 \/ x2 = x + 1 \/ x3 = x + 1;
                                              3-6 links ...

% first line
constraint links(rikudo[1], rikudo[2], rikudo[5], rikudo[6]);
constraint links(rikudo[2], rikudo[1], rikudo[3], rikudo[6], rikudo[7]);
constraint links(rikudo[3], rikudo[2], rikudo[4], rikudo[7], rikudo[8]);
constraint links(rikudo[4], rikudo[3], rikudo[8], rikudo[9]);
                                              1-7 line …

% for links
                                  rikudo[links[1]] = rikudo[links[2]] + 1
                                                  OU
                                  rikudo[links[2]] = rikudo[links[1]] + 1
```

III/ Rikudo


```
$> minizinc --all --solver Gecode resolution.mzn data3.dzn
      |09|10|13|14|
    |07|08|11|12|15|
  |05|06|31|32|33|16|
|03|04|30|XX|34|17|18|
  |02|29|28|35|20|19|
    |01|26|27|36|21|
      |25|24|23|22|
----------
==========
```

Merci
Avez-vous des questions ?