Le problème des n reines

Position du problème

On se place sur un échiquier de taille $n \times n$, sur lequel n reines sont disposées. On dit qu'une reine est *menacée* si elle se trouve sur la même ligne, colonne ou diagonale qu'une autre reine. On cherche à placer les n reines de sorte qu'aucune reine ne soit menacée.
On va chercher, en partant d'une situation initiale quelconque, à déplacer les reines de sorte à atteindre une position dans laquelle aucune d'elles n'est menacée. Pendant ces déplacements, on va permettre à chaque case de contenir un nombre quelconque de reines simultanément, et aussi d'être marquée par des signaux provenant d'un nombre quelconque de directions, qui traduisent la présence d'une reine dans la direction d'où ils proviennent.
On cherche à se donner une procédure de mise à jour locale telle qu'à chaque pas de temps, les états de certaines cases sont modifiés à partir d'informations sur les cases voisines : création et transmission de signaux, déplacements de reines... L'objectif est que les signaux traduisent au mieux la présence ou l'absence de reines dans la direction d'où ils viennent, et que les reines menacées se déplacent au cours du temps pour chercher des cases sûres. On cherche également à ce que les reines ne bougent plus une fois qu'une solution est atteinte.
On s'astreint à ne déplacer les reines que d'au plus une case par pas de temps, de sorte que la règle de mise à jour puisse rester purement locale. On voudrait ne déplacer les reines que sur des cases sûres, mais c'est trop restrictif : il existe des positions qui ne sont pas solution du problème mais où toutes les reines sont entourées de cases menacées (cf. Figure 1). En revanche, permettre aux reines de se déplacer sur des cases dangereuses sans utiliser les informations locales dont elles disposent donne des mouvements trop aléatoires pour qu'on puisse espérer atteindre une solution en un temps raisonnable. On fixe donc une probabilité $\varepsilon \in [0, 1]$, qui correspond à la probabilité pour une reine qui envisage un déplacement vers une case dangereuse de l'effectuer, alors qu'un déplacement vers une case sûre est

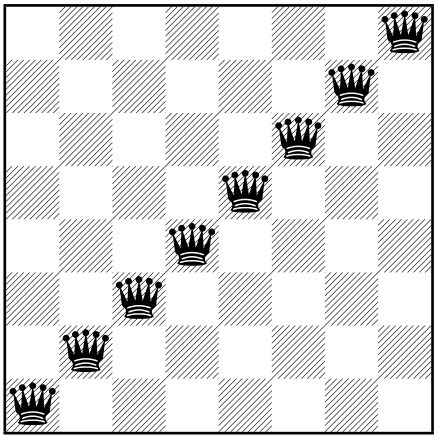

FIGURE 1 – Les reines sont toutes menacées, mais aucun déplacement d'une case ne conduit à une case sûre.

systématique. En fixant ε assez petit (de l'ordre de 0.01 par exemple), on privilégie les déplacements vers des cases sûres tout en se donnant quand même la possibilité de sortir de situations bloquées.

Formalisation

Représentation du système

On représente l'échiquier par l'ensemble $\mathcal{L} = [\![0, n-1]\!]^2$. On pose $\mathcal{P} = \{0, 1\}$ et $\mathcal{D} = \{NW, N, NE, W, E, SW, S, SE\}$ l'ensemble des huit directions. L'état d'une cellule étant déterminé par le nombre de reines présentes et la présence ou l'absence d'un signal en provenance de chaque direction, on note $\mathcal{Q} = \mathcal{P}^8 \times \mathbb{N}$ l'ensemble des états. Une *configuration* est un élément de $\mathcal{E} = \mathcal{Q}^{\mathcal{L}}$. Pour toute configuration x, et toute cellule $c \in \mathcal{L}$, on note $x_c = (s_{x,c}, n_{x,c})$ l'état de la cellule c dans la configuration x, où $n_{x,c}$ est le nombre de reines sur la cellule c dans la configuration x, et $s_{x,c} = (s_{x,c,N}, s_{x,c,NE}, s_{x,c,E}, \ldots, s_{x,c,NW})$ code la présence ou l'absence de signaux venant de chaque direction : $s_{x,c,d}$ vaut 1 si dans la configuration x, la cellule c contient un signal provenant de la direction d, et 0 sinon.
Par souci de lisibilité, on omettra les indices x de configuration partout où on pourra le faire sans ambiguïté et on parlera de l'état (s_c, n_c) d'une cellule c.
On pose $\mathcal{N} \subset \mathcal{L}^2$ l'ensemble des paires de cellules adjacentes :

$$(c, c') \in \mathcal{N} \iff \left\|(c - c')\right\|_\infty = 1$$

À chaque pas de temps, on choisit au hasard une paire de cases adjacentes et on les met à jour l'une par rapport à l'autre. On se donne donc $(u_t)_{t\in\mathbb{N}} \in \mathcal{N}^{\mathbb{N}}$ la suite des paires de cellules mises à jour : pour tout $t \in \mathbb{N}$, la paire de cellules mises à jour à l'étape t est u_t. Ce sont les valeurs d'une suite de variables aléatoires indépendantes à distribution uniforme sur $\mathcal{N}$. On définit de même $(x_t)_{t\in\mathbb{N}}$ la suite des configurations prises par le système au cours du temps. Initialement, aucun signal n'est présent et les n reines sont réparties aléatoirement sur l'échiquier, selon une loi uniforme (de façon indépendante : notamment plusieurs reines peuvent se trouver initialement sur la même case).
On cherche à définir une fonction de mise à jour globale $\mathcal{F} : \mathcal{E} \times \mathcal{N} \to \mathcal{E}$ qui à une configuration associe une configuration qui lui succède selon la procédure de la partie précédente : $\mathcal{F}(x_t, u_t) = x_{t+1}$.

Choix des cases à mettre à jour

On souhaite choisir une paire de cases adjacentes à mettre à jour, avec la même probabilité de choisir chaque paire de cases. On observe qu'il y a $n(n-1)$ paires de cases voisines verticalement, $n(n-1)$ horizontalement et $(n-1)^2$ selon chaque diagonale, soit $(4n-2)(n-1)$ paires de cases adjacentes au total. On applique donc la procédure de sélection suivante :

- On choisit selon une loi uniforme $N \in [\![0, (4n-2)(n-1)-1]\!]$, on en déduit $A \in [\![0, 4n-3]\!]$ et $B \in [\![0, n-2]\!]$ le quotient et le reste de la division euclidienne de N par $(n-1)$.
- Si $A < n$, on prend les cases (B, A) et $(B+1, A)$.
- Si $A = n + i, 0 \leq i < n$, on prend les cases (i, B) et $(i, B+1)$.
- Si $A = 2n + i, 0 \leq i < n-1$, on prend (i, B) et $(i+1, B+1)$.
- Sinon, $A = 3n - 1 + i$ et $0 \leq i < n-1$, et on prend $(i, B+1)$ et $(i+1, B)$.

La fonction qui à N associe une paire de cases adjacentes étant bijective, on a bien la même probabilité de choisir chaque paire de cases.

Mise à jour : les déplacements

On se donne deux cases adjacentes c et c' à mettre à jour. On commence par envisager des déplacements de reine entre c et c'. Le principe général est qu'une reine menacée essaie de quitter sa case lorsqu'elle est mise à jour. Ce déplacement réussit ou non en fonction de l'état de l'autre case mise à jour, avec une part d'aléatoire. La procédure est la suivante :

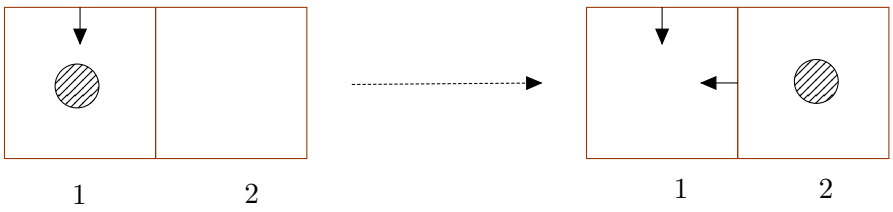

FIGURE 2 – La reine présente en 1, menacée depuis le nord, se déplace en 2 qui est une case sûre, puis les signaux sont mis à jour : il y a une reine en 2, donc un signal venant de l'est apparaît en 1.

- Si une reine est présente en case c, alors que la case c contient un signal, elle essaie de se déplacer vers la case c'. De même, si plusieurs reines sont présentes en c, elles sont menacées et l'une d'entre elles tente de se déplacer vers la case c'.
- S'il n'y a pas de signal en c' sauf éventuellement un venant de c (qui est peut-être dû à la présence de la reine qui cherche à bouger), et qu'il n'y a pas d'autre reine en c', la case est considérée comme sûre et le déplacement a lieu (cf. Figure 2). Sinon, la case est dangereuse, et le déplacement a lieu seulement avec une probabilité ε.
- Symétriquement, une reine de c' peut se déplacer vers c selon la même procédure. Les deux déplacements se font de manière simultanée afin que l'ordre des deux cases choisies pour la mise à jour n'ait pas d'importance.

Formalisons cette procédure à l'aide de quelques fonctions auxiliaires :

- On pose SAFESTAY : $\mathcal{L} \to \{0,1\}$ qui indique si une reine peut rester sur une case donnée :

$$\text{SAFESTAY}(c) = \begin{cases} 1 & \text{si } n_c \leq 1 \text{ et } \forall d \in \mathcal{D}, s_c(d) = 0 \\ 0 & \text{sinon} \end{cases}$$

- On pose SAFEGO : $\mathcal{L} \times \mathcal{D} \to \{0,1\}$ qui indique si une reine peut se déplacer de façon sûre vers une certaine case en venant d'une certaine direction :

$$\text{SAFEGO}(c, d_{\text{from}}) = \begin{cases} 1 & \text{si } n_c = 0 \text{ et } \forall d \in \mathcal{D} \setminus \{d_{\text{from}}\}, s_c(d) = 0 \\ 0 & \text{sinon} \end{cases}$$

  En toute rigueur, SAFESTAY et SAFEGO devraient prendre aussi la configuration en argument : on l'omet pour plus de lisibilité.

- On pose $\delta : \mathcal{N} \to \mathcal{D}$ telle que $\delta(c_{\text{from}}, c_{\text{to}})$ soit la direction de c_{from} vue de c_{to}. Plus formellement :

$$\delta(c_{\text{from}}, c_{\text{to}}) = \begin{cases} N & \text{si } c_{\text{to}} - c_{\text{from}} = (0,1) \\ NE & \text{si } c_{\text{to}} - c_{\text{from}} = (1,1) \\ \cdots & \\ W & \text{si } c_{\text{to}} - c_{\text{from}} = (-1,0) \\ NW & \text{si } c_{\text{to}} - c_{\text{from}} = (-1,1) \end{cases}$$

- On peut alors définir $\textsc{move} : \mathcal{N} \to \{0,1\}$ qui traduit la procédure décrite ci-dessus : $\textsc{move}(c_{\text{from}}, c_{\text{to}})$ vaut 1 si une reine se déplace de c_{from} à c_{to} et 0 sinon. Formellement :

$$\textsc{move}(c_{\text{from}}, c_{\text{to}}) = \begin{cases} 0 & \text{si } \textsc{safestay}(c_{\text{from}}) = 1 \\ 1 & \text{si : } \begin{cases} \textsc{safestay}(c_{\text{from}}) = 0 \\ \text{et } n_{c_{\text{from}}} \geq 1 \\ \text{et } \textsc{safego}(c_{\text{to}}, \delta(c_{\text{from}}, c_{\text{to}})) = 1 \end{cases} \\ \mathcal{B}(\varepsilon) & \text{sinon} \end{cases}$$

  où $\mathcal{B}(\varepsilon)$ est une variable aléatoire valant 1 avec une probabilité ε et 0 avec une probabilité $1-\varepsilon$. Comme précédemment, on omet de passer la configuration en argument à MOVE tant qu'il n'y a pas d'ambiguïté.

On peut maintenant écrire $\mathcal{F}_{\text{move}} : \mathcal{E} \times \mathcal{N} \to \mathcal{E}$ qui modifie la configuration par d'éventuels déplacements de reine entre les cases mises à jour :

$$\mathcal{F}_{\text{move}}(x, (c_1, c_2)) = \begin{cases} x_{c_1 \to c_2} & \text{si } \textsc{move}(c_1, c_2) = 1 \text{ et } \textsc{move}(c_2, c_1) = 0 \\ x_{c_2 \to c_1} & \text{si } \textsc{move}(c_1, c_2) = 0 \text{ et } \textsc{move}(c_2, c_1) = 1 \\ x & \text{sinon} \end{cases}$$

où $x_{a \to b}$ désigne la configuration obtenue à partir de x en retranchant 1 à n_a et en ajoutant 1 à n_b.

Mise à jour : les signaux

Après avoir effectué d'éventuels déplacements de reine, on met à jour les signaux de c et c'.

- Un signal en c' provenant de la direction de c apparaît ou reste présent s'il y a une reine en c, (cf. Figure 2) ou s'il y a un signal en c dans

la même direction (intuitivement, il y a une reine plus loin dans la direction de c) (cf. Figure 3). Sinon, rien n'indique plus la présence d'une reine dans cette direction : le signal disparaît s'il était présent (cf. Figure 4).

- Symétriquement, le signal en c provenant de c' est mis à jour de la même façon.
- Les autres signaux ne sont pas modifiés.

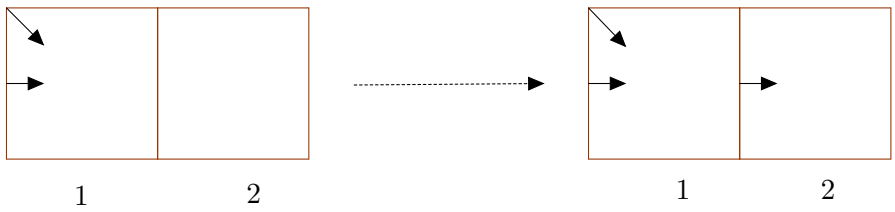

FIGURE 3 – Le signal de la cellule 1 venant de l'ouest est transmis à la cellule 2.

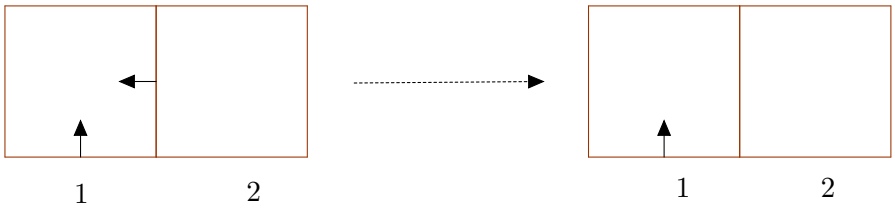

FIGURE 4 – Le signal en 1 venant de l'est disparaît.

Formellement, on pose, pour tout $d \in \mathcal{D}$, $s^{c'}_{x,c,d}$ le signal de la case c se rapportant à la direction d, mis à jour par rapport à c' selon la règle :

$$s^{c'}_{x,c,d} = \begin{cases} s_{x,c,d} & \text{si } d \neq \delta(c',c) \\ 1 & \text{si } d = \delta(c',c) \text{ et } n_{x,c'} \geq 1 \\ s_{x,c',d} & \text{sinon} \end{cases}$$

On va s'en servir pour définir une fonction $\mathcal{F}_{\text{sign}} : \mathcal{E} \times \mathcal{N} \to \mathcal{E}$ de mise à jour des signaux : $\mathcal{F}_{\text{sign}}(x, (c_1, c_2)) = x'$ avec :

$$\begin{cases} x'_{c_1} = (s^{c_2}_{x,c_1}, n_{x,c_1}) \\ x'_{c_2} = (s^{c_1}_{x,c_2}, n_{x,c_2}) \\ \qquad x'_c = x_c \qquad\qquad \text{pour tout } c \notin \{c_1, c_2\} \end{cases}$$

On peut alors définir la fonction de mise à jour :

$$\mathcal{F}(x_t, u_t) = \mathcal{F}_{\text{sign}}(\mathcal{F}_{\text{move}}(x_t, u_t), u_t)$$

On remarque que lors de la mise à jour d'une case, la nouvelle valeur du signal qu'on modifie ne dépend pas de l'ancienne valeur, mais seulement de la case voisine : les signaux sont transmis localement de manière instantanée mais les cases n'ont pas de mémoire. Ainsi, lorsqu'une reine quitte une rangée de cases, les signaux dans cette direction n'arrivent plus et l'information de l'absence de cette reine est propagée.
L'asynchronisme de la mise à jour est utile : si on mettait les cases à jour de façon synchrone, deux reines sur la même ligne, colonne ou diagonale recevraient en même temps l'information, et pourraient se déplacer toutes les deux en même temps alors que le comportement souhaité est qu'une seule des deux reines se déplace.

Expérimentations

Lancé à partir d'une situation de départ aléatoire (la case de départ de chacune des n reines est choisie selon une loi uniforme de façon indépendante), on observe que le système finit par atteindre un point fixe où chaque reine est sur une case sans signal. La position des reines est alors solution du problème des n reines.

Comportement qualitatif

On observe en général un régime transitoire sur les premiers milliers de pas de temps, pendant lesquels les signaux ne se sont pas encore propagés sur tout l'échiquier, donc pendant lesquels les mouvements des reines sont trop libres par rapport à la situation globale. Les mouvements des reines sont ensuite très restreints : une fois les signaux convenablement propagés, il y a très peu de cases sûres sur l'échiquier, voire aucune. De temps en temps, une reine se déplace vers une case dangereuse, ce qui libère certaines cases ; des déplacements vers des cases devenues sûres ont lieu puis on retrouve une situation de blocage, et ainsi de suite jusqu'à ce qu'une solution soit trouvée.

Effets de ε

Le paramètre ε a des effets multiples. Plus ε est grand, plus le système sort rapidement de situations bloquées où aucun déplacement vers une case sûre n'est possible. En revanche, plus il est petit, plus le système a tendance à effectuer tous les déplacements possibles vers des cases sûres avant de déplacer une reine vers une case dangereuse, ce qui peut empêcher de passer à côté d'une solution. En effet, si une solution est accessible sans avoir besoin de déplacer de reine vers une case dangereuse, plus ε est grand, plus le risque de déplacer une reine vers une case dangereuse et d'avoir peu de chances de trouver rapidement cette solution est important. Enfin, comme il y a typiquement très peu de cases sûres, ε contrôle également la fréquence des déplacements : plus il est petit, plus les signaux ont de temps pour se propager sur l'échiquier avant le prochain déplacement de reine. Ceci implique que plus ε est petit, plus les reines se déplacent en se basant sur des informations à jour. Indirectement, ε contrôle donc également le degré d'asynchronisme à retards du système.
Le choix de ε pour minimiser le temps mis par le système pour trouver une solution relève donc d'un compromis entre ces différents effets : on le voudrait petit pour privilégier les déplacements sûrs et déplacer les reines à

partir d'informations valides, mais grand pour que les déplacements soient suffisamment fréquents.
La figure 5 représente la variation du temps de convergence (le nombre de mises à jour avant d'atteindre une solution) et du nombre de déplacements effectifs de reine en fonction de la probabilité ε, dans le cas $n = 8$. Les valeurs sont des moyennes sur 100 échantillons, sauf les deux points extrémaux où l'on s'est restreint à 30 échantillons pour des raisons de temps de calcul. On observe un optimum de ε pour ce qui est du temps de convergence, autour de 0.01, tandis que le nombre de déplacements croît avec ε sur les valeurs considérées.
La monotonie apparente du nombre de déplacements en fonction de ε n'est pas étonnante : diminuer ε revient à augmenter la qualité des déplacements (plus souvent vers des cases sûres, avec des informations plus à jour) quitte à ce qu'ils aient lieu moins fréquemment.
Ce modèle ne fait pas de distinction entre les cases dangereuses : on aurait pu envisager, par exemple, de toujours autoriser les déplacements vers des cases portant moins de signaux que la case de départ. Ceci autorise beaucoup de déplacements de reine et le comportement du système est analogue à celui de notre modèle avec ε très élevé : la convergence, s'il y en a une, est en fait moins rapide.

Temps de convergence en fonction de n

La figure 6 représente l'évolution du temps de convergence en fonction de n, dans le cas $\varepsilon = 0.01$. Le temps de convergence semble exponentiel en n. On a tracé sur la même figure la variation, en fonction de n, de $1/d$ où d est la densité des solutions au problème des n reines (quotient du nombre de solutions par le nombre total de positions). Le nombre de solutions est une valeur tabulée par recherche exhaustive [1]. On constate que d décroît également exponentiellement, ce qui tend à expliquer la croissance rapide du temps de convergence. L'irrégularité au point $n = 6$ s'explique de même par le fait que la variation du nombre de solutions au problème présente la même irrégularité. En effet, il existe 10 solutions au problème pour $n = 5$, mais seulement 4 pour $n = 6$ alors qu'asymptotiquement, le nombre de solutions croît exponentiellement, même si le nombre total de positions croît encore plus rapidement.

1. http ://jsomers.com/nqueen_demo/nqueens.html

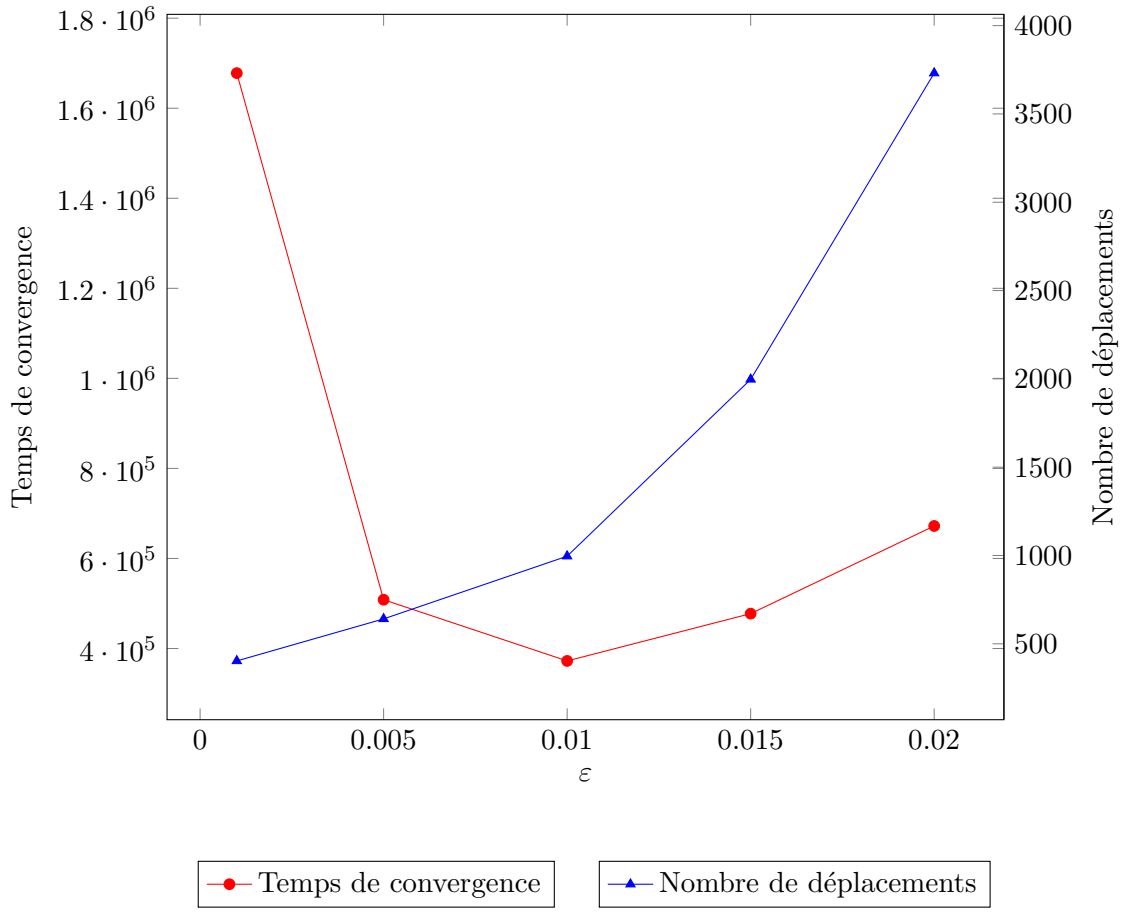

FIGURE 5 – Temps de convergence moyen et nombre moyen de déplacements en fonction de ε pour $n = 8$.

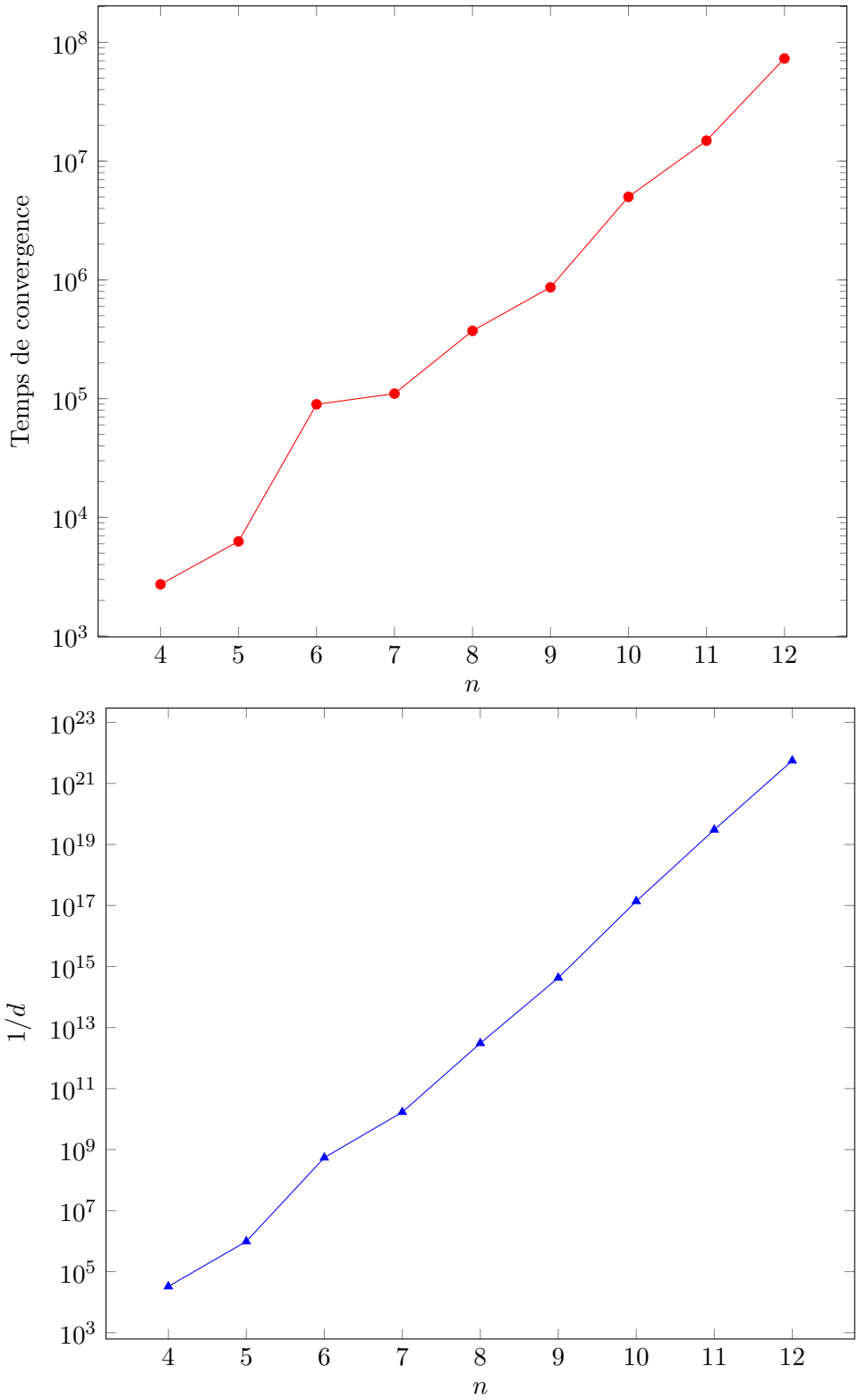

FIGURE 6 – Temps de convergence pour $\varepsilon = 0.01$ et inverse de la densité de solutions en fonction de n (échelle logarithmique).

Un autre algorithme

Le modèle précédent ne passe pas très bien à l'échelle : obtenir une solution pour des tailles supérieures à 12 peut s'avérer très long. On va donc envisager un nouveau modèle dans lequel plus d'informations sont prises en compte à chaque étape de calcul. L'objectif est que l'évolution du système soit plus dirigée que précédemment, tout en se restreignant encore à des règles de mise à jour locales. Plutôt que de choisir une paire de cases à chaque pas de temps et ne considérer que les informations contenues dans ces deux cases, on va mettre à jour une case par pas de temps, en utilisant les informations de toutes ses voisines.
On utilise le même système de signaux que dans l'algorithme précédent et on garde le même espace d'états : $\mathcal{Q} = \mathcal{P}^8 \times \mathbb{N}$. On pose, pour tout $c \in \mathcal{L}$, $\mathcal{V}_c \subset \mathcal{L}$ l'ensemble des cellules voisines de c : $\mathcal{V}_c = \{c' \in \mathcal{L}, \|(c - c')\|_\infty = 1\}$. À chaque pas de temps, on choisit une case au hasard, on met à jour ses signaux par rapport à ceux des cellules voisines, et une reine quitte éventuellement la case.

Contrairement à l'algorithme précédent, comme les signaux d'une seule case sont modifiés à chaque pas de temps, on les met à jour avant d'envisager un déplacement de reine. On met à jour les signaux d'une cellule c par rapport à chacune de ses voisines, de la même façon que dans le modèle précédent : pour toute cellule $c' \in \mathcal{V}_c$, le signal en c provenant de c' apparaît ou reste présent s'il y en a un en c' provenant de la même direction, ou s'il y a une reine en c'. Dans le cas contraire, il disparaît ou n'apparaît pas.

L'idée de la procédure pour déplacer une reine est la suivante : si on met à jour une case c contenant des reines menacées, l'une d'entre elles choisit une case de $\mathcal{V}_c$, et tente de s'y déplacer selon la même procédure que dans l'algorithme précédent.
On définit $\omega : \mathcal{L} \to \mathbb{N} \cup \{\infty\}$ qui sert à quantifier à quel point une cellule est dangereuse, de la façon suivante :

$$\omega(c) = \begin{cases} \infty \text{ si } n_c > 0 \\ \sum_{d \in \mathcal{D}} s_{c,d} \text{ sinon} \end{cases}$$

On aimerait toujours envoyer une reine menacée de c vers une cellule de V_c qui minimise ω, mais il existe des positions pour lesquelles un tel choix déterministe ne permet pas de résoudre le problème. Ainsi, la position de la figure 7, avec les signaux convenablement propagés (ie. une case porte un

signal venant d'une certaine direction si et seulement si il y a effectivement une reine dans cette direction), est une position insoluble. En effet, en partant de cette position, on aura à chaque instant les reines des colonnes **c** à **f** sur leur case initiale, et les deux autres dans le triangle de cases colorées en rouge, ce qui ne peut constituer une solution. Lorsqu'elles bougent, les deux reines du triangle rouge restent dedans : à chaque instant, la case rouge inoccupée vérifie $\omega = 2$, contre 3 ou 4 pour les cases adjacentes au triangle rouge. Quelle que soit la position des deux reines dans le triangle rouge, les autres reines ne sont pas menacées : à chaque instant, le prochain mouvement est donc celui d'une des deux reines du triangle rouge, et elle reste dans ce triangle.

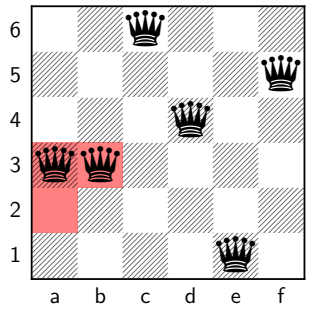

FIGURE 7 – Exemple de position insoluble avec $\eta = 0$.

On veut donc que le choix de la destination ne soit pas toujours la case adjacente qui minimise ω. On se donne donc une probabilité $\eta \in [0, 1]$, et on choisit avec une probabilité η une case aléatoire de V_c comme destination, et avec une probabilité $1 - \eta$ une case aléatoire parmi les éléments de V_c qui minimisent ω. Ainsi, la position de la figure 7 ne pose pas de problème pour $\eta > 0$: une reine va finir par sortir du triangle rouge et une des autres reines se déplacera par la suite.
Une fois la destination choisie, la procédure qui décide si le déplacement se fait ou non est exactement la fonction MOVE de l'algorithme précédent.

La figure 8 montre que ce nouvel algorithme passe un peu mieux à l'échelle que précédemment : la pente de la courbe qui représente le temps de convergence moyen en fonction de n est plus faible pour le nouvel algorithme que pour l'ancien. De fait, il permet de trouver en un temps raisonnable (quelques minutes sur une machine individuelle) des solutions jusqu'à $n = 16$, contre 12 pour le modèle précédent, alors que les temps de calcul sont très

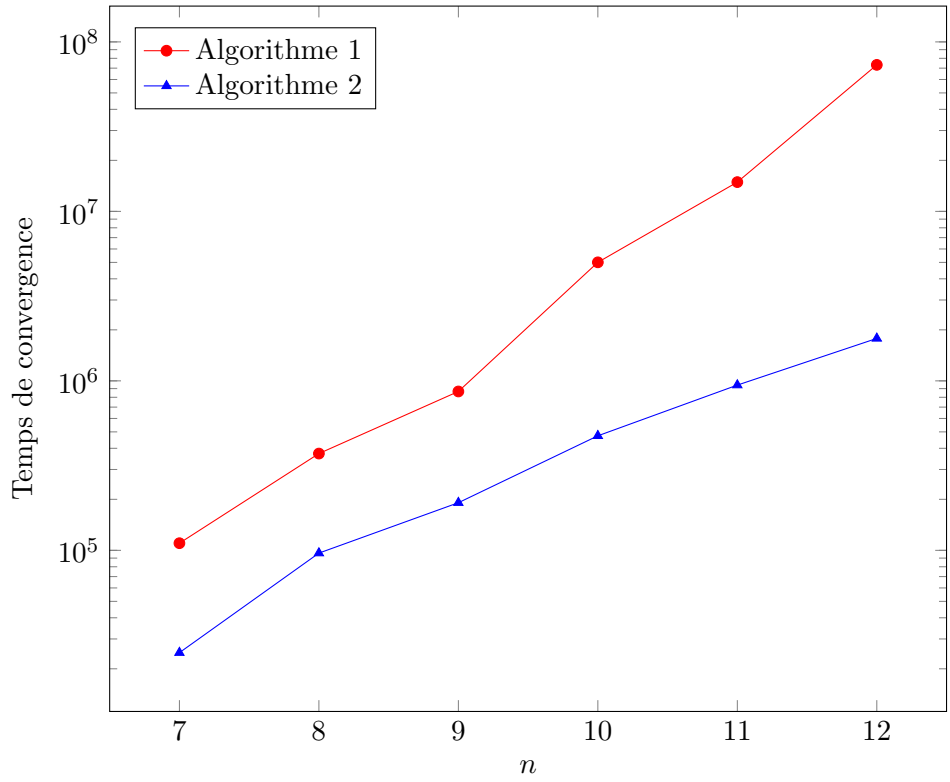

FIGURE 8 – Temps de convergence moyen en fonction de n.

similaires pour $n = 8$. Ceci peut s'expliquer par le fait que les mouvements sont plus dirigés que dans le premier algorithme, car on utilise plus d'informations à chaque coup et on s'efforce de choisir des directions intéressantes pour déplacer les reines.

					1	0	1	1	$\times$
					1	0	1	1	1
$+$				0	0	0	0	.	0
$+$			1	0	1	1	.	.	1
$+$		1	0	1	1	.	.	.	1
	1	0	0	0	1	1	1	1	

FIGURE 9 – Multiplication binaire de 11 (1011 en base 2) par 13 (1101, écrit verticalement de bas en haut). Les lignes non-nulles sont bien égales à translation près.

Lien avec la factorisation

Dans le cadre du problème de la factorisation, les problèmes liés à la propagation de contraintes peuvent se traduire de manière assez analogue aux signaux transmis par les cellules de ce modèle. En effet, les termes de l'addition binaire doivent respecter une contrainte globale : les lignes non-nulles sont toutes égales (à la translation due à la retenue près) : si on pose la multiplication binaire de a par b, on somme des copies de a, décalées de i crans vers la gauche pour tout $i \in I$, avec I tel que $b = \sum_{i \in I} 2^i$ (cf. Figure 9) . Ceci peut se traduire de la façon suivante : une cellule sur la même ligne qu'un 1 (la ligne est non nulle) et sur la même diagonale qu'un 1 doit contenir elle-même un 1. On peut donc envisager de transmettre des signaux sur les lignes et les diagonales pour détecter la présence de 1 avec des règles purement locales.

Références

[1] Jacques M. Bahi and Sylvain Contassot-Vivier. Basins of attraction in fully asynchronous discrete-time discrete-state dynamic networks. *IEEE Transactions on Neural Networks*, 17(2) :397–408, 2006.

[2] Vincent Chevrier and Nazim Fatès. Multi-agent Systems as Discrete Dynamical Systems : Influences and Reactions as a Modelling Principle. Rapport de recherche, 2008.

[3] Nazim Fatès. Stochastic Cellular Automata Solutions to the Density Classification Problem - When Randomness Helps Computing. *Theory Comput. Syst.*, 53(2) :223–242, 2013.