Un programme d'intelligence artificielle en LOGO
LE PROBLÈME DES HUIT REINES

Michel DUPONT

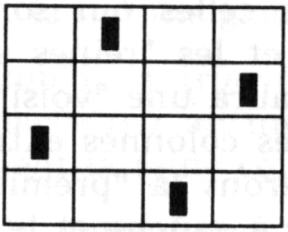
1. PRÉSENTATION

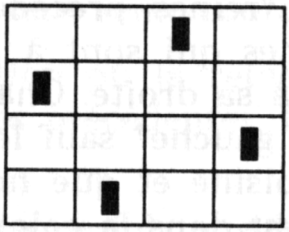
Certains avantages du LOGO sont bien connus: la grande facilité d'accès que procure sa TORTUE en fait un outil privilégié des pédagogues et son caractère procédural facilite la programmation structurée. Il permet de plus à l'utilisateur de faire évoluer son environnement de programmation en ajoutant ses propres procédures aux primitives. Il est même ainsi possible de créer des sur-langages adaptés à des domaines particuliers.

Mais LOGO présente d'autres avantages moins connus qui en font notamment un langage bien adapté pour traiter les problèmes relevant de ce qu'il est convenu d'appeler l'Intelligence Artificielle. Nous en donnons une illustration avec l'étude du problème des huit reines. On verra que LOGO présente d'indéniables facilités lors de l'écriture du programme mais, ce langage, pas plus qu'aucun autre, n'est doté d'une quelconque intelligence et il ne peut nous dispenser d'analyser le problème, de trouver un algorithme et d'en donner une description avant de passer à l'écriture du programme. Nous nous attacherons donc à détailler toute cette démarche.

Il s'agit de placer huit reines sur un échiquier 8x8, telles qu'aucune ne soit menacée par l'une des 7 autres. C'est avec cette dimension conventionnelle de l'échiquier que le problème a été posé en 1848 par un certain Max Bezzel. Rappelons pour la petite histoire que les 92 solutions ont été données dès 1850 par le Dr Nauck et que Gauss n'en a trouvé que 72.

Il convient de préciser que dans ce décompte les solutions qui se déduisent l'une de l'autre par rotation ou par symétrie sont considérées comme des solutions distinctes. C'est ainsi que, pour un échiquier 4x4, nous considérons que le problème admet les deux solutions ci-dessous :





Il est bien évidemment possible de traiter le problème quelle que soit la dimension de l'échiquier pourvu qu'il soit carré et que le nombre de reines soit égal au nombre de cases par côté. Notre programme devra d'ailleurs être capable de donner les solutions dans tous les cas. Seule la nécessité de ne pas atteindre des durées de traitement prohibitives nous amènera à limiter la dimension de l'échiquier.

2. DESCRIPTION DE L'ALGORITHME

Étant donné que nous allons découvrir l'algorithme en travaillant "à la main", nous vous conseillons de limiter la dimension de l'échiquier à 4 cases de côté. Il est inutile d'y faire apparaître des cases blanches et des cases noires car les couleurs des cases ne concernent pas les reines. Rappelons en effet que pour qu'une reine ne soit menacée par aucune autre reine trois conditions doivent être remplies:

1. il ne doit pas y avoir d'autres reines sur sa colonne;
2. il ne doit pas y avoir d'autres reines sur sa ligne;
3. il ne doit pas y avoir d'autres reines sur "ses diagonales" (nous appelons ainsi, par abus de langage, les deux parallèles aux diagonales du carré, qui passent par sa case).

Avec la stratégie que nous préconisons, la première condition est remplie d'emblée. Nous décidons en effet d'affecter une reine à chaque colonne. La position de toutes les reines peut ainsi s'exprimer simplement par une suite de nombres qui représentent les numéros des lignes des reines prises de la gauche vers la droite. Ainsi, les solutions sur un échiquier 4x4 peuvent s'exprimer : 2 4 1 3 et 3 1 4 2 .

Nous appellerons "reine courante" la reine que nous déplaçons. Les "reines précédentes" sont celles qui sont situées dans les colonnes qui sont à sa gauche et les "reines suivantes" celles qui sont à sa droite. Chaque reine aura une "voisine droite" et une "voisine gauche" sauf les reines des colonnes extrêmes qui n'ont qu'une voisine et que nous appellerons la "première reine" pour celle qui est dans la colonne la plus à gauche et le "dernière reine" pour celle qui est dans la colonne la plus à droite.

Ces points de vocabulaire étant précisés, nous pouvons commencer. On démarre en plaçant la 1ère reine sur la 1ère case. Puis, c'est au tour de la 2ème reine que l'on place sur la case 1 et que l'on monte de case en case jusqu'à ce qu'elle ne soit plus menacée par la première reine. Elle monte ainsi jusqu'à la troisième case. On place ensuite la troisième reine et ainsi de suite en appliquant les huit règles suivantes.

1. Une reine ne peut être déplacée que dans sa colonne.
2. Une reine ne se déplace qu'en montant de case. Si elle est sur la dernière case, on ne peut que la remettre sur la case de départ. Cela se fait par application de la règle 5.
3. On change de reine courante en prenant une voisine.
4. Si c'est la voisine gauche qui devient la reine courante, elle monte d'une case en partant de l'endroit où on l'avait laissée précédemment. Si elle est déjà en haut de l'échiquier, c'est la nouvelle voisine gauche qui devient la reine courante par application de la règle 7.
5. Si c'est la voisine droite qui devient la reine courante, elle part de la case de départ.
6. Lorsqu'on déplace la reine courante, on cherche une position où elle n'est plus menacée par les reines précédentes sans se soucier des reines suivantes. En d'autres termes, la solution est toujours recherchée de la gauche vers la droite. On fait donc monter la reine courante case par case jusqu'à ce qu'elle soit libre de toute menace des reines précédentes ou qu'elle arrive en haut de l'échiquier.
7. Lorsque la reine courante est en haut de l'échiquier et qu'on cherche à la faire monter pour appliquer les règles 4 ou 6, c'est la voisine gauche qui devient reine courante.

8. Lorsqu'on a trouvé une position où la reine courante n'est menacée par aucune des reines précédentes, la voisine droite devient la reine courante.

On obtient une solution chaque fois que la dernière reine (devenue reine courante) trouve une position où elle est libre de toute menace. Puisqu'on ne change de reine courante qu'en prenant une voisine, c'est alors l'avant dernière reine qui devient reine courante et on continue en s'en tenant toujours à nos huit règles.

L'algorithme est terminé lorsque la première reine est déjà en haut de l'échiquier et que la deuxième reine arrive à son tour en fin de colonne. La première reine devient en effet reine courante mais la règle 8 est inapplicable puisqu'elle ne peut plus monter et qu'elle n'a pas de voisine gauche.

3. REPRESENTATION EN ARBRE

Nous allons maintenant formaliser le problème avec une représentation en arbre. La terminologie propre à cette méthode caractéristique de l'Intelligence Artificielle est suffisamment explicite pour qu'il ne soit pas nécessaire de donner toutes les définitions. Nous parlerons en effet de la racine, des nœuds, des feuilles, du parcours de l'arbre... Signalons seulement que les feuilles sont des nœuds terminaux.

Le dessin de l'arbre s'obtient en recherchant par une méthode classique de l'analyse combinatoire toutes les positions qui peuvent être prises par les reines en s'en tenant à notre première règle à savoir que chaque reine ne peut se déplacer que sur sa colonne.

La première rangée de nœuds à partir de la racine représente les positions de la première reine (positions indiquées sur notre dessin reproduit en annexe par le numéro des lignes), la deuxième rangée celles des deux premières reines (un couple de chiffres indique les numéros des lignes) et ainsi de suite jusqu'aux feuilles qui représentent l'ensemble des positions de toutes les reines.

On voit rapidement que le nombre de feuilles est N^N pour un échiquier de N cases de côté; soit 27 feuilles pour un échiquier 3x3, 256 feuilles pour un échiquier 4x4, 3 125 pour un échiquier 5x5, 46 656 pour un échiquier 6x6...

Compte tenu de cette inflation galopante, vous comprendrez que nous ne cherchions pas à dessiner un arbre complet pour un échiquier de plus de 3 cases de côté. Bien que dans cette dimension le problème n'admette aucune solution, il est possible de représenter notre algorithme par un parcours des décisions sur l'arbre correspondant. Chaque déplacement d'un nœud au suivant représente l'une des trois décisions possible.

1. Passage de la reine courante sur la case supérieure. Décision que nous symboliserons ultérieurement par la lettre H (comme haut). Le déplacement correspondant sera représenté sur l'arbre par le signe : 
2. Changement de reine courante en prenant la voisine gauche. Cette décision sera symbolisée par la lettre G et le déplacement correspondant sera représenté par le signe : 
3. Changement de reine courante en prenant la voisine droite. Symbole D. Signe 

Le parcours des décisions peut être représenté par une suite des trois signes que nous venons de définir. La taille des signes varie en fonction du niveau sur lequel ils se trouvent sur l'arbre.

Pour utiliser cette représentation, nous sommes amenés à planter notre arbre avec la racine à gauche et les feuilles à droite alors qu'habituellement ce type d'arbre se plante avec la racine en haut ! Fi des habitudes et des lois de la botanique ! Examinons le parcours obtenu (voir l'annexe 3). Il appelle deux remarques.

1. On cherche au départ à s'enfoncer le plus profondément possible dans l'arbre et on termine par la racine. Ce type de parcours est connu sous le nom de préordre ou parcours antérieur ou encore profondeur d'abord. C'est le parcours le plus couramment utilisé mais il en existe d'autres : le postordre ou parcours postérieur, le parcours largeur d'abord et le parcours intérieur.

2. Certaines parties de l'arbre ne sont pas explorées. C'est le cas dès le début du parcours lorsqu'on s'aperçoit que la décision prise au départ de placer les deux premières reines sur la première ligne était mauvaise. On décide alors de déplacer la 2ème reine sur la 2ème ligne puis sur la 3ème et c'est seulement à ce stade qu'on décide d'essayer de placer la 3ème reine. Sur notre exemple 21 feuilles sur 27 sont ainsi délaissées. Sur des arbres plus importants, ce procédé permettrait

d'économiser l'exploration non seulement de feuilles mais aussi celle de branches à plusieurs ramifications. Cette technique qui consiste à revenir sur des choix faits antérieurement et qui s'avèrent mauvais s'appelle le *backtrack*. Elle permet d'économiser distance, temps et énergie. Seuls les petits chefs bornés hésitent à l'utiliser. Hélas! L'Intelligence Artificielle n'a rien à céder à ceux que la nature n'a pas suffisamment dotés en matière grise.

4. LE PROGRAMME

Mais, ne nous égarons pas ! A ce stade de notre étude, une autre remarque s'impose : nous n'avons pas encore parlé de programmation ce qui prouve qu'il n'est pas obligatoire de faire de l'informatique pour s'intéresser à l'algorithmique et à l'Intelligence Artificielle. Cependant, on perçoit maintenant tout l'intérêt que présente le traitement du problème par les moyens de l'informatique. Nous voyons en effet à quel point la mise en œuvre de notre algorithme "à la main" devient longue, délicate et fastidieuse lorsque le nombre de cases de l'échiquier augmente. Seul le recours à l'ordinateur permet alors de traiter le problème dans des conditions de facilité et de rapidité acceptables.

Dans notre version, le programme intègre, outre l'algorithme que nous avons décrit, deux techniques expliquées dans notre étude préliminaire. L'édition des solutions se présente en effet sous la forme déjà employée d'une suite de nombres représentant les numéros des lignes de chaque reine et le programme est doté d'une fonction de trace rudimentaire qui affiche les symboles D, G et H que nous avons définis lors de la formalisation du problème par une arborescence. Par ailleurs, le programme contient un compteur qui permet d'indiquer, en fin d'exécution, le nombre des solutions.

Nous vous proposons de découvrir maintenant ce programme en examinant à la fois le listing et l'organigramme reproduits en annexe. Nous avons indiqué en caractères gras sur le listing tout ce qui concerne le compteur et la fonction de trace. Nous n'en parlerons plus. Quant aux parties grisées de l'organigramme, elles correspondent à chacune des cinq procédures.

Les reines, représentées chacune par un nombre qui indique le numéro de ligne, sont constamment réparties entre la reine courante (`reine :C`), les reines précédentes (`liste :P`) et les reines suivantes (`liste :S`).

Le programme est lancé avec la procédure DEPART dont le paramètre :N indique le nombre de cases par côté de l'échiquier. Cette procédure initialise le programme ainsi : la liste :P est vide; la liste :S contient (:N - 1) reines représentées chacune par un 0; la reine courante :C est la première reine (elle est représentée par un 1 puisqu'elle est sur la première ligne).

Les deux procédures D (voisine droite) et G (voisine gauche) permettent le changement de reine courante. Dans la procédure D, conformément à la règle 5 de notre étude préliminaire, la reine courante est placée sur la première case avec DONNE "C 1.

Au centre du programme se trouve la procédure DECIDER qui permet notamment de comparer la position de la reine courante avec celles des reines précédentes pour voir si elle est menacée. Elle contient trois paramètres.

1. La reine courante :C.
2. Une liste :LL qui contient au départ toutes les reines précédentes. A chaque appel récursif, une reine est ensuite supprimée de cette liste avec SD :LL.
3. Un compteur :X qui démarre avec 1 et qui est incrémenté à chaque appel récursif. Il indique en fait le nombre de colonnes qui séparent la reine courante de la dernière reine de la liste :LL.

La structure de cette procédure est assez classique. Elle contient trois tests répartis sur deux niveaux. A chaque appel de la procédure un test principal enclenche en effet l'un des deux tests secondaires. Quatre possibilités sont ainsi offertes. Celle qui apparaît en dernier sur la procédure est un appel récursif qui fonctionne en démontant :LL avec SD :LL et en incrémentant :X. Il s'agit donc d'une procédure récursive dont la condition d'arrêt est recherchée par le test principal (SI VIDE? :LL). Lorsque cette condition d'arrêt est satisfaite le premier des tests secondaires ouvre deux possibilités. Si la reine courante est la dernière reine (SI VIDE? :S), une solution vient d'être trouvée. Elle est donc éditée. Sinon, c'est la voisine droite qui devient reine courante.

Quant au deuxième test secondaire, c'est lui qui compare les positions respectives de la reine courante et de la dernière reine de la liste :LL pour détecter si elles sont en prise. Afin d'en expliquer le fonctionnement nous allons, dans un premier temps, utiliser les notations conventionnelles en mathématiques. Nous prendrons trois variables C, L et X qui représentent respectivement :C, DER :LL et :X.

Étant donné qu'avec notre algorithme les reines sont placées d'emblée dans des colonnes distinctes, deux reines ne peuvent être en prise que si elles sont sur une même ligne ou sur une même diagonale. Si elles sont sur une même ligne, on aura C=L qu'on peut également écrire C–L=0. Pour qu'elles soient sur une même diagonale, il y a deux possibilités C–L=X ou C–L=–X. En d'autres termes, on peut dire que pour que deux reines soient en prise, il faut que la différence entre C et L soit égale à 0, X ou –X. En LOGO cela s'écrira :

```
SI MEMBRE? DIFF :C DER :LL PH PH 0 :X PROD :X -1
```

Dès que cette condition est satisfaite, la procédure `MONTER` est enclenchée. Cette dernière procédure permet de simuler le passage de la reine courante à la case supérieure par incrémentation de `:C` sauf si elle est déjà en haut de l'échiquier (`SI EGAL? :C :N`). Dans ce dernier cas, deux possibilités sont à nouveau envisageables : si la reine courante est la première reine (`SI VIDE? :P`), l'algorithme est terminé et le programme édite le nombre de solutions, sinon la voisine gauche devient la reine courante.

5. CONCLUSIONS

Le programme ainsi obtenu est étonnant de brièveté au regard de la complexité du problème et de la longueur des explications qu'il nécessite. Mais, plus que les performances du programme lui-même, ce sont précisément les explications qui permettent d'en saisir le fonctionnement qui nous ont paru digne d'intérêt puisqu'elles permettent d'aborder sur un exemple concret quelques notions de l'Intelligence Artificielle.

Nous avons voulu montrer en quoi l'ordinateur apporte des facilités dans la résolution de ce type de problème tout en illustrant que les moyens de l'informatique par eux-mêmes ne peuvent apporter aucune solution. Même si des programmes de ce type imitent l'intelligence humaine avec des procédés comme le backtrack, ils ne font que mettre en oeuvre les algorithmes que nous élaborons.

Enfin, nous avons voulu montrer que le LOGO qui est issu de LISP, langage spécialement conçu pour l'Intelligence Artificielle, est bien adapté pour traiter de tels problèmes. L'algorithme que nous avons utilisé est en effet connu de longue date et il a d'ailleurs été traité en BASIC dans divers ouvrages notamment dans "Mathématiques élémentaires d'un point de vue algorithmique (Ed. CEDIC)" et dans

"Programmes d'intelligence artificielle en basic (Ed. EYROLLES)". Mais, par rapport à ces versions, l'écriture en LOGO est plus claire. Nous avons vu en effet qu'elle est très proche de la description de l'algorithme que nous avons donnée.

Michel DUPONT,
Avranches 1988

ANNEXE 1

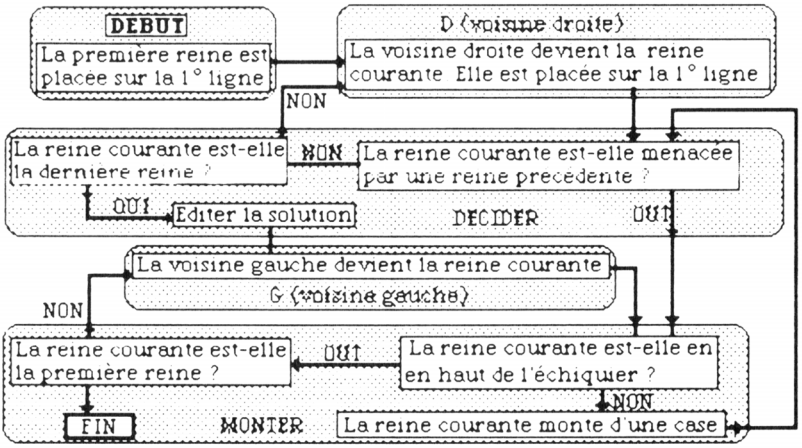
```
POUR DEPART :N
VT TAPE PH "reines.$
DONNE "CPT 0
DONNE "P []
DONNE "S []
REPETE ( :N - 1 ) [DONNE "S MD 0 :S]
D
FIN

POUR G                          POUR D
TAPE "G                         TAPE "D
DONNE "S MP :C : S              DONNE "P MD :C :P
DONNE "C DER :P                 DONNE "S SP :S
DONNE "P SD :P                  DONNE "C 1
MONTER                          DECIDER :C :P 1
FIN                             FIN

POUR DECIDER :C :LL :X
SI VIDE? :LL [SI VIDE? :S [EC [] TAPE PH PH :P :C "$
DONNE "CPT ( :CPT + 1 ) G] [D] ] [SI MEMBRE? DIFF :C
DER :LL PH PH 0 :X PROD  :X -1 [MONTER] [DECIDER :C
SD :LL ( :X + 1 )]]
FIN

POUR MONTER
SI EGAL? :C :N [SI VIDE? :P [EC [] TAPE :CPT EC "$
solutions.  STOP] [G]] [TAPE "H DONNE "C ( :C + 1 )
DECIDER :C :P 1]
FIN
```

ANNEXE 2



ANNEXE 3