*Un exemple d'utilisation du programme n-Queens Completion pour compléter des compositions*

*Grigoryan E.*

*1. Génération de la composition*

Dans le programme *Generation\_k\_Queens\_Composition*, définissez *n = 100* pour afficher un damier de *100 x 100*. Après le démarrage du programme, une composition arbitraire sera formée. Soit le résultat une composition de *47* reines: *Q(1:100) =*

0 44 16 33 77 0 0 52 30 0

0 0 91 0 66 31 0 0 0 0

0 56 53 59 28 22 39 0 0 0

90 0 0 0 37 0 97 99 24 87

38 49 0 94 0 74 0 0 0 0

65 0 47 0 92 0 0 0 0 7

69 80 0 0 0 0 0 35 88 76

0 29 0 1 18 50 19 0 0 0

68 0 0 10 43 0 0 0 100 61

8 0 0 60 41 0 0 0 0 0

Cent nombres, qui sont énumérés ci-dessus, caractérisent systématiquement *100* lignes de la matrice de décision. Une valeur de zéro indique que la reine n'est pas placée dans la ligne correspondante (la ligne est libre), tout autre nombre indique la position de la reine dans la ligne en question. Par exemple, dans la composition résultante, le premier élément du tableau Q (1: 100) est égal à 0, ce qui signifie que la première ligne de l'échiquier est libre; le deuxième élément du tableau est 44, ce qui signifie que dans la deuxième rangée du damier la reine est située en position 44; etc.

Sauvegardons les données reçues dans le fichier *kQueens\_Test\_Composition.mat*. Ce nom apparaît à la fin du programme *Generation\_k\_Queens\_Composition* à titre d'exemple. (Évidemment, vous pouvez spécifier n'importe quel autre nom).

1. *Compléter la composition*

Pour terminer la composition résultante, exécutez le programme *Solution\_n\_Queens\_Completion\_Problem* pour l'exécution, où vous devez spécifier le nom du fichier d'entrée avec la composition. (Dans cet exemple, nous utilisons le nom du fichier de données d'entrée: *kQueens\_Test\_Composition.mat*) En conséquence, la composition sera terminée, et les résultats seront enregistrés dans le fichier: *nQueens\_Test\_Completion\_Solution.mat*. (Ce nom de fichier est choisi comme exemple). La solution résultante a la forme:

>> Solution \_nQueens\_Completion\_Problem

Input file name: kQueens\_Test\_Composition.mat

The size of a chessboard= 100

Composition Size = 47

Number of free Positions = 53

The first 50 positions of queens:

0 44 16 33 77 0 0 52 30 0

0 0 91 0 66 31 0 0 0 0

0 56 53 59 28 22 39 0 0 0

90 0 0 0 37 0 97 99 24 87

38 49 0 94 0 74 0 0 0 0

Elapsed time is 0.051897 seconds.

Number of complete re-counting cycles = 0

Total number of usage the Back Tracking procedure = 0

Solution is Ok!

The first 50 positions of solution:

58 44 16 33 77 26 63 52 30 34

20 57 91 54 66 31 93 71 55 21

84 56 53 59 28 22 39 95 51 25

90 81 3 75 37 79 97 99 24 87

38 49 11 94 6 74 40 4 12 15

65 36 47 2 92 73 67 27 85 7

69 80 14 89 96 5 83 35 88 76

48 29 13 1 18 50 19 82 32 17

68 70 72 10 43 46 86 78 100 61

8 64 98 60 41 45 9 42 23 62

Solution saved in file: nQueens\_Test\_Completion\_Solution.mat

1. *Vérification de l'exactitude de la solution*

Pour vérifier l'exactitude de la solution obtenue, nous utiliserons le programme *Validation\_n\_Queens\_Problem\_Solution*. Ici, le nom du fichier d'entrée est *nQueens\_Test\_Completion\_Solution.mat*. (Si un autre fichier sera pris en compte, vous devez spécifier le nom approprié.) Après le démarrage, le programme affichera le message suivant:

>> Validation\_n\_Queens\_Problem\_Solution

Input file name: nQueens\_Test\_Completion\_Solution.mat

The size of a chessboard = 100

Elapsed time is 0.000021 seconds.

Solutions is ok!

1. *Vérification de l'exactitude de la composition*

Comme nous l'avons déjà dit, le programme *Validation\_n\_Queens\_Problem\_Solution* permet non seulement de vérifier l'exactitude de la solution du *problème n-Queens*, mais aussi l'exactitude d'une composition arbitraire.

Pour vérifier l'exactitude de la composition résultante, exécutez le programme *Validation\_n\_Queens\_Problem\_Solution* pour l'exécution, où vous devez spécifier le nom du fichier d'entrée avec la composition. (Ici, en tant que fichier de données d'entrée, le nom *kQueens\_Test\_Composition.mat* est spécifié, que nous avons utilisé pour enregistrer la composition).

À la suite de la vérification de la composition considérée, le programme affiche le message suivant, confirmant l'exactitude de la composition:

>> Validation\_n\_Queens\_Problem\_Solution

Input file name: kQueens\_Test\_Composition.mat

The size of a chessboard = 100

Elapsed time is 0.000014 seconds.

Composition size = 47

Composition is ok!

1. *Achèvement d'une composition "complexe"*

Regardons un autre exemple. Soit, à la suite de la génération, une composition qui a la forme suivante: *Q(1:100) =*

0 91 10 0 29 39 59 77 26 2

95 20 94 58 64 22 83 76 34 4

37 0 51 25 0 38 0 50 68 0

0 87 19 21 92 8 96 75 48 0

60 93 46 100 24 66 23 9 53 80

65 0 85 45 28 61 31 89 18 54

0 3 0 0 27 32 0 13 15 0

82 0 70 1 98 47 73 0 62 49

79 36 41 74 0 17 55 90 0 78

0 44 11 16 14 56 0 7 12 99

La composition ici se compose de *80* reines. Le contrôle montre que la composition est correcte. Lorsque le programme *Solution\_n\_Queens\_Completion\_Problem* démarre, les calculs prennent un peu plus de temps que d'habitude, mais en conséquence le programme affiche le message suivant:

>> Solution \_nQueens\_Completion\_Problem

Input file name: kQueens\_Test\_Composition.mat

The size of a chessboard = 100

Composition Size = 80

Number of free Positions = 20

Elapsed time is 2.187818 seconds.

falseNegSimCount = 5

Total number of usage the Back Tracking procedure = 5474

Columns 1 through 10

30 91 10 72 29 39 59 77 26 2

95 20 94 58 64 22 83 76 34 4

37 69 51 25 5 38 81 50 68 43

84 87 19 21 92 8 96 75 48 40

60 93 46 100 24 66 23 9 53 80

65 97 85 45 28 61 31 89 18 54

71 3 33 57 27 32 88 13 15 52

82 67 70 1 98 47 73 42 62 49

79 36 41 74 63 17 55 90 6 78

35 44 11 16 14 56 86 7 12 99

Solution saved in file: nQueens\_Test\_Completion\_Solution.mat

Comme nous pouvons le voir, l'algorithme a essayé cinq fois depuis le tout début de terminer cette composition (*falseNegSimCount = 5*), et ce n'est qu'à la sixième tentative qu'il a réussi. Pendant ce temps, la procédure de *Back Tracking* a été exécutée plus de *5000* fois.

La complexité de cette composition réside dans le fait que *80* reines y sont situées de telle sorte que non seulement 80 colonnes sont fermées dans les lignes libres restantes, mais également la plupart des positions libres restantes. C'est le résultat du travail des contraintes diagonales, qui sont formées par les reines précédemment fixées. Lors de l'achèvement, aux dernières étapes, une situation survient lorsqu'une position libre reste sur deux ou trois rangées libres et une tentative de placer la reine dans l'un d'entre eux fermera immédiatement la position libre dans une autre rangée (en raison de restrictions diagonales). C'est un exemple plutôt rare et intéressant.

Nous appelons ces compositions qui ne peuvent pas être complétées comme négatives. Voici un exemple de composition négative: *Q (1:100) =*

21 31 89 20 50 46 28 39 78 66

57 45 37 52 41 100 92 8 29 19

58 75 6 25 96 34 71 95 15 23

79 73 68 88 67 72 35 98 86 1

83 12 24 44 32 61 14 70 99 80

60 47 81 77 7 59 2 38 0 13

17 51 90 55 49 16 22 26 11 0

76 18 91 87 63 93 9 0 64 94

4 0 56 0 10 5 30 85 65 97

42 40 0 69 54 0 0 62 27 3

Après le démarrage du programme, le message suivant s'affiche:

>> Solution \_nQueens\_Completion\_Problem

Input file name: kQueens\_Test\_Composition.mat

The size of a chessboard = 100

Composition Size = 92

Number of free Positions = 8

Elapsed time is 0.148292 seconds.

falseNegSimCount = 10

Total number of usage the Back Tracking procedure = 10084

This composition cannot be completied!

The error of such conclusion is less than 0.00001

Comme on peut le voir, l'algorithme a essayé dix fois depuis le tout début de compléter cette composition. Dans le même temps, dans l'ensemble, la procédure de *Back Tracking* a été utilisée plus de *10 000* fois pour former une solution. Mais toutes ces tentatives ont échoué. Par conséquent, en tenant compte du fait que la limite autorisée d'options pour trouver une solution a été épuisée), il est décidé qu'avec une probabilité de *0,00001* cette composition ne peut pas être achevée.

Si nous imprimons le nombre de positions libres dans les lignes restantes, le résultat sera à chaque fois quelque chose comme ceci:

*étape-93:* 1 1 1 1 1 2 3

*étape-94:* 0 1 1 1 1 1

ou

*étape-94:* 0 0 1 1 2 3

Ou des options proches de cela. Quelle que soit la ligne libre que nous choisissons pour positionner la reine, elle fermera la dernière position libre dans l'une des lignes libres restantes.

1. *Un exemple de composition de grande taille*

Let's generate a composition for a chessboard, the size of which is *100 \* 106*. Let the size of the composition be equal to *7 888 826*. Here are the first *50* values of the array of values of queens' positions:

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 9002222 0

0 0 0 0 0 0 0 13573604 0 0

84739423 0 0 0 0 0 39546789 0 0 0

21295044 99655382 0 0 0 0 0 0 0 0

- Le temps nécessaire pour générer une telle composition était de *17.59* secondes (tous les calculs ont été effectués sur *DeskTop-13*, la configuration de cet ordinateur est décrite dans la publication au *arhiv.org:* <https://arxiv.org/abs/1912.05935> ).

- Le temps nécessaire pour achever la composition résultante était de *384.51* secondes. Voici les 50 premières valeurs consécutives du tableau de solution à 1 dimension:

89582163 60368706 59234218 20261866 86935797 30224854 77724126 36762224 45267928 73444501

30441569 40452963 29233166 74466403 31027892 16431542 83945356 25649731 9002222 26312393

99907709 4403090 19464886 43088993 92254324 37208851 91076429 13573604 48390687 50900879

84739423 22553311 18099114 61465930 76001929 56522925 39546789 31942742 33849160 75874627

21295044 99655382 79211814 74172179 90416939 47905163 37217953 24667656 80381835 35026191

- Le temps pour vérifier l'exactitude de la solution obtenue était de *15.71* secondes.

>> Validation\_n\_Queens\_Problem\_Solution

Input file name: kQueens\_Test\_Composition.mat

The size of a chessboard = 100000000

Elapsed time is 15.712291 seconds.

Solutions is ok!

1. *Sur la linéarité de l’algorithme d’achèvement des compositions*

Tous les programmes gardent une trace de la durée d'exécution de l'algorithme et, une fois terminé, le résultat correspondant s'affiche. Supposons qu'un grand nombre d'expériences ont été effectuées avec différentes compositions (positives et négatives) pour différentes valeurs de *n*. Déterminons le temps moyen pour terminer chaque composition pour chaque valeur de *n*. En conséquence, nous obtenons un échantillon des temps d'exécution moyens pour chaque valeur de *n*. Si nous divisons la valeur moyenne du temps d'achèvement du prélèvement par la valeur correspondante de *n*, nous obtenons le "*temps de traitement d'une ligne*". Il s'agit du temps moyen nécessaire à l'algorithme pour placer la reine unique sur une ligne. Si l'algorithme considéré est linéaire dans le temps, alors avec une valeur croissante de *n*, le temps de traitement d'une ligne ne devrait pas changer. Dans une petite erreur, *le temps donné doit rester constant*.

Pour une évaluation rapide de l'algorithme, il suffit de générer *20-30* compositions pour une liste de valeurs *n = (100, 1000, 10 000, 100 000, 1 000 000, 10 000 000).* Vous pouvez obtenir des résultats plus précis si vous créez et testez de très grands échantillons de compositions pour différentes valeurs de *n*. Au cours de la recherche, des échantillons de *100 000* compositions ont généralement été formés pour différentes valeurs de *n*. Lorsque le temps total de comptage augmentait de manière significative, la taille de l'échantillon diminuait en conséquence. Pour *n = 1000*, pour l'étude, un échantillon a été formé, constitué *d'un million* de compositions.

L'étude a examiné la gamme des valeurs de taille d'échiquier *de 7 à cent millions*. Pour les valeurs *n = (7, ..., 99)*, l'algorithme désactive certaines procédures. Cela est dû au fait que pour une plage donnée de *n* valeurs, il est nécessaire de sélectionner plus soigneusement les indices de lignes libres et les indices de positions libres dans les lignes sélectionnées. Au sens figuré, nous pouvons dire que cette zone est assez turbulente et que l'algorithme de calcul de cette section (qui fait *0,000001* partie de l'intervalle entier) diffère en efficacité de l'algorithme principal.