یک روش برنامه‌ریزی عدد صحیح برای حل مسئله N-Queen با استفاده از بهینه‌سازی خطی با Pyomo

مقدمه

مسئله N-Queen یک مسئله کلاسیک ترکیبی است که هدف آن قرار دادن N وزیر روی یک صفحه شطرنج N در N است به طوری که هیچ دو وزیری یکدیگر را تهدید نکنند. این به این معنی است که هیچ دو وزیری نباید در یک سطر، ستون یا قطر مشترک باشند. این گزارش یک روش برای حل مسئله N-Queen با استفاده از تکنیک‌های بهینه‌سازی خطی که با استفاده از کتابخانه Pyomo در پایتون پیاده‌سازی شده است را توصیف می‌کند.

فرمولاسیون برنامه‌ریزی خطی

برای حل مسئله N-Queen ، LP به صورت زیر می‌باشد:

متغیرها:

$$x_{ij} : \begin{cases} 1 & \text{اگر وزیر در سطر} i \text{ و ستون } j \text{قرار دارد} \\ 0 & \text{در غیر این صورت} \end{cases}$$

محدودیت‌ها:

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \quad \text{(هر سطر دقیقا یک وزیر داشته باشد)}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j \quad \text{(هر ستون دقیقا یک وزیر داشته باشد)}$$

$$\sum_{i=1}^{n} x_{i,i+k} \leq 1 \quad \forall k \in \{-(n-1), ..., (n-1)\} \quad \text{(قطرها بالا\_چپ به پایین\_راست)}$$

$$\sum_{i=1}^{n} x_{i,k-i} \leq 1 \quad \forall k \in \{0, ..., 2(n-1)\} \quad \text{(قطرها بالا\_راست به پایین\_چپ)}$$

کد برنامه

در این بخش کد برنامه ارائه شده است و هر خط از کد توضیح داده شده است.

```
import numpy as np
import matplotlib.pyplot as plt
from pyomo.environ import ConcreteModel, Var, ConstraintList, SolverFactory, Binary, value

def solve_n_queens(n):
if n > 27:
raise ValueError("n must be less than or equal to 27.")

model = ConcreteModel()

# Define binary variables for each position on the board
model.x = Var(range(n), range(n), within=Binary)

# Define constraints
model.row_constraint = ConstraintList()
model.column_constraint = ConstraintList()
model.diagonal_constraint1 = ConstraintList()
model.diagonal_constraint2 = ConstraintList()

for i in range(n):
model.row_constraint.add(sum(model.x[i, j] for j in range(n)) == 1)

for j in range(n):
model.column_constraint.add(sum(model.x[i, j] for i in range(n)) == 1)

for k in range(-(n-1), n):
model.diagonal_constraint1.add(sum(model.x[i, i + k] for i in range(n) if 0 <= i + k < n) <=

for k in range(2 * n - 1):
model.diagonal_constraint2.add(sum(model.x[i, k - i] for i in range(n) if 0 <= k - i < n) <=

# Solve the problem using GLPK solver
solver = SolverFactory('glpk')
solver.solve(model)

# Extract the solution
queens = []
for i in range(n):
for j in range(n):
if value(model.x[i, j]) == 1:
queens.append((i, j))
```

```
# Create the board matrix
board = np.zeros((n, n))
for queen in queens:
row, col = queen
board[row][col] = 1

return board, queens

def plot_board(board, queens):
n = board.shape[0]
# Create a checkerboard pattern
checkerboard = np.zeros((n, n, 3))
for i in range(n):
for j in range(n):
if (i + j) % 2 == 0:
checkerboard[i, j] = [1, 1, 1]  # White square
else:
checkerboard[i, j] = [0, 0, 0]  # Black square

plt.figure(figsize=(6, 6))
plt.imshow(checkerboard, interpolation='nearest')

for queen in queens:
row, col = queen
plt.text(col, row, 'Q', ha='center', va='center', color='red', fontsize=20)

# Add grid lines
plt.grid(which='both', color='black', linestyle='-', linewidth=2)
plt.xticks(np.arange(0.5, n, 1), [])
plt.yticks(np.arange(0.5, n, 1), [])
plt.title(f"{n}-Queens Solution")
plt.show()

while True:
n = int(input('Enter the number of queens (0 to exit): '))
if n == 0:
break
elif n <= 27:
board, queens = solve_n_queens(n)

print("Solution in matrix form:")
print(board)

print("\nLocations of queens (i, j):")
for queen in queens:
print(queen)
```

```
plot_board(board, queens)
else:
print("Please provide a value of n less than or equal to 27.")
```

توضیحات خط به خط کد

- import numpy as np: کتابخانه NumPy برای ایجاد و دستکاری آرایه‌های چندبعدی استفاده می‌شود.
- import matplotlib.pyplot as plt: کتابخانه Matplotlib برای ترسیم نمودارها استفاده می‌شود.
- from pyomo.environ import ConcreteModel, Var, ConstraintList, SolverFactory, Binary, value: وارد کردن کتابخانه Pyomo برای مدل‌سازی بهینه‌سازی ریاضی.
- def solve_n_queens(n): تعریف تابعی برای حل مسئله n_queens با تعداد n.
- if n > 27: raise ValueError("n must be less than or equal to 27."): بررسی می‌کند که مقدار n بیشتر از ۲۷ نباشد و در صورت بیشتر بودن، خطایی ایجاد می‌کند.
- model = ConcreteModel(): ایجاد یک مدل Pyomo.
- model.x = Var(range(n), range(n), within=Binary): تعریف متغیرهای باینری برای هر موقعیت در صفحه شطرنج.
- model.row_constraint = ConstraintList(): تعریف لیست محدودیت‌ها برای سطرها.
- model.column_constraint = ConstraintList(): تعریف لیست محدودیت‌ها برای ستون‌ها.
- model.diagonal_constraint1 = ConstraintList(): تعریف لیست محدودیت‌ها برای قطرها (بالا–چپ به پایین–راست).
- model.diagonal_constraint2 = ConstraintList(): تعریف لیست محدودیت‌ها برای قطرها (بالا–راست به پایین–چپ).
- for i in range(n): model.row_constraint.add(sum(model.x[i, j] for j in range(n)) == 1): اضافه کردن محدودیت‌ها برای هر سطر به طوری که هر سطر تنها یک وزیر داشته باشد.
- for j in range(n): model.column_constraint.add(sum(model.x[i, j] for i in range(n)) == 1): اضافه کردن محدودیت‌ها برای هر ستون به طوری که هر ستون تنها یک وزیر داشته باشد.
- for k in range(-(n-1), n): model.diagonal_constraint1.add(sum(model.x[i, i + k] for i in range(n) if 0 <= i + k < n) <= 1): اضافه کردن محدودیت‌ها برای قطرها از بالا–چپ به پایین–راست.

- for k in range(2 * n - 1): model.diagonal_constraint2.add(sum(model.x[i, k - i] for i in range(n) if 0 <= k - i < n) <= 1): اضافه کردن محدودیت‌ها برای قطرها از بالا ـ راست به پایین ـ چپ.
- solver = SolverFactory('glpk'): تعریف حل‌کننده GLPK برای حل مدل.
- solver.solve(model): حل مدل.
- queens = []: ایجاد لیستی برای ذخیره محل وزیرها.
- for i in range(n): for j in range(n): if value(model.x[i, j]) == 1: queens.append((i, j)): بازیابی محل وزیرها از مدل حل شده.
- board = np.zeros((n, n)): ایجاد ماتریس صفحه شطرنج با اندازه $n \times n$.
- for queen in queens: row, col = queen; board[row][col] = 1: پر کردن ماتریس با محل قرارگیری وزیرها.
- return board, queens: بازگرداندن ماتریس صفحه شطرنج و محل وزیرها.
- def plot_board(board, queens): تعریف تابعی برای ترسیم صفحه شطرنج و نمایش محل وزیرها.
- n = board.shape[0]: دریافت اندازه صفحه شطرنج.
- checkerboard = np.zeros((n, n, 3)): ایجاد یک الگوی صفحه شطرنج با سه کانال رنگی.
- for i in range(n): for j in range(n): if (i + j) % 2 == 0: checkerboard[i, j] = [1, 1, 1] else: checkerboard[i, j] = [0, 0, 0]: پر کردن الگوی صفحه شطرنج با رنگ‌های سفید و سیاه.
- plt.figure(figsize=(6, 6)): ایجاد یک شکل جدید برای نمودار.
- plt.imshow(checkerboard, interpolation='nearest'): نمایش الگوی صفحه شطرنج با استفاده از Matplotlib.
- for queen in queens: row, col = queen; plt.text(col, row, 'Q', ha='center', va='center', color='red', fontsize=20): قرار دادن علامت 'Q' در محل هر وزیر با رنگ قرمز.
- plt.grid(which='both', color='black', linestyle='-', linewidth=2): اضافه کردن خطوط شبکه برای جداسازی خانه‌های صفحه شطرنج.
- plt.xticks(np.arange(0.5, n, 1), []): حذف برچسب‌های محور x.
- plt.yticks(np.arange(0.5, n, 1), []): حذف برچسب‌های محور y.
- plt.title(f"n-Queens Solution"): افزودن عنوان نمودار با ذکر تعداد وزیرها.
- plt.show(): نمایش نمودار.
- while True: n = int(input('Enter the number of queens (0 to exit): ')): حلقه اصلی برای دریافت ورودی کاربر و نمایش نتایج.

- if n == 0: break: خروج از حلقه اگر کاربر عدد ۰ وارد کند.
- elif n <= 27: board, queens = solve_n_queens(n); plot_board(board, queens): بررسی می‌کند که مقدار n کمتر یا برابر ۲۷ باشد و در این صورت مسئله را حل می‌کند و نتیجه را ترسیم می‌کند.
- else: print("Please provide a value of n less than or equal to 27."): چاپ پیام خطا اگر مقدار n بیشتر از ۲۷ باشد.

ورودی و خروجی نمونه

در این بخش، یک نمونه ورودی و خروجی برای برنامه ارائه شده است.

ورودی

فرض کنید کاربر عدد ۸ را به عنوان تعداد وزیرها وارد می‌کند:

```
Enter the number of queens (0 to exit): 8
```

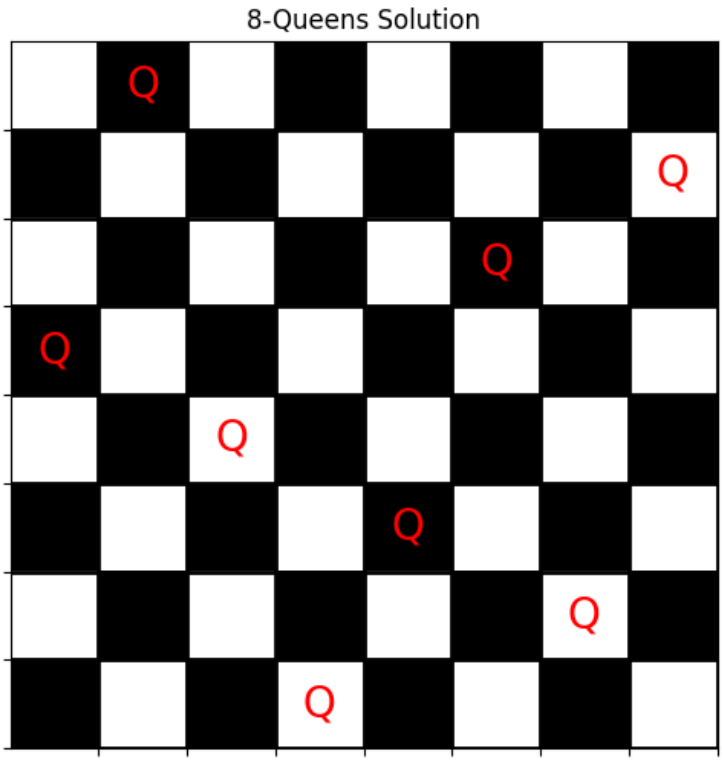
خروجی

برنامه پس از حل مسئله، محل قرارگیری وزیرها را به صورت یک ماتریس و یک نمودار شطرنجی نمایش می‌دهد. خروجی برنامه به صورت زیر خواهد بود:

```
Solution in matrix form:
[[1. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 1. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 1.]
[0. 0. 0. 0. 0. 1. 0. 0.]
[0. 0. 1. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 1. 0.]
[0. 1. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 1. 0. 0. 0. 0.]]

Locations of queens (i, j):
(0, 0)
(1, 4)
(2, 7)
(3, 5)
(4, 2)
(5, 6)
(6, 1)
(7, 3)
```

و یک نمودار شطرنجی که وزیرها را به صورت بصری نشان می‌دهد:



در نمودار بالا، هر وزیر به صورت یک خانه سیاه نشان داده شده است که موقعیت آنها به گونه‌ای است که هیچ دو وزیری یکدیگر را تهدید نمی‌کنند.

توضیحات خروجی

در خروجی بالا، لیستی از جفت‌های مرتب ارائه شده که هر جفت نشان‌دهنده موقعیت یک وزیر در صفحه شطرنج است. به عنوان مثال، $(0, 0)$ نشان می‌دهد که یک وزیر در سطر اول و ستون اول قرار دارد. همچنین نمودار شطرنجی بصری به کاربر کمک می‌کند تا محل وزیرها را به راحتی ببیند و درک کند.