The n-Queen Problem

黃偉祥 X1136010

Problem formulation

- IDS use bitwise mask to check next legal move and search the next nodes
- Hill climbing and Genetic algorithm use each array[i] to represent queen at column i → array[0] is queen at column 0, if array[0] == 4 → queen i at column 0 row 4
- Hill climbing neighbour → any swap is a neighbour
- Genetic algorithm declare as below different parameters

IDS

- Use bit operation to get next legal placement

Hill Climbing

- max iteration = 999, swap to get neighbour

Genetic Algorithm

- 1-point crossover
- Random to choose parent to generate child
- Random to select if put child or put both parent into next generation
- Mutation may swap more than once for a child.

OX-cross GA strategy(parameter)

- Use **order crossover**
- **Mutation** only appear **at most once** in a child
- Population model, **GA2** = **SSGA**, **GA** = **GGA**
- **Parent selection** = **Tournament Selection**
- **GA2**: **Survivor selection(fitness-based)**, **GA**: **random** choose **parent or child** to be survivor

8-Queen Result (basic algorithm)

Results	IDS	Hill Climbing	Genetic Algorithm
Average #attacks	0	0.533333	0.566667
Average running time	0ms	0.0333333	4.63333ms

Results	IDS	Hill Climbing	Genetic Algorithm
Success rate	100%	46.6667%	43.3333%

- Genetic algorithm parameters
  - **populations size** = **100**, **generations** = **100**
  - If use changed strategy
    - **OX-cross GA**: ave run time = 3ms, ave #ATK = 0.566667, SR = 46.6667%
    - **OX-cross GA2**: run time=1.83333ms, #ATK= 0.333333, SR=66.6667%
      - with p=100, g=10, k=30

Hill Climbing

Results	stuck and break	random restart, max=999
Average #attacks	0.533333	0.933333
Average running time	0.0333333	0.0666667
Success rate	46.6667%	96.6667%

- Using **random restart** when stuck with maximum iterations = **9999** can increase success rate but take more time.

Genetic Algorithm

Results	p = 500, g = 100	p = 100, g = 500	p = 500, g = 500	p = 1000, g = 100	p = 100, g = 1000
Average #attacks	0.0666667	0.333333	0.133333	0	0.2
Average running time	8.86667ms	24.6667ms	37.4333ms	3.83333ms	39.5333ms
Success rate	93.3333%	66.6667%	86.6667%	100%	80%

- p → populations size, g → generations
- Increase both **populations size** and **generations** can increase **success rate** in general, but sometimes **increase both may not** better than increase populations size only.
- Increase **populations size** can significantly **increase success rate** and decrease average #attacks
- Increasing **populations size** may help to get optimal solution faster, cause have more chance to escape local minimal
- Increasing **generations** may help get to optimal solution (due to mutation) but not as much as populations size, and it takes more time.

50-Queen Result (basic algorithm)

Results	IDS(1 iteration)	Hill Climbing	Genetic Algorithm
Average #attacks	0	1.03333	15
Average running time	1001 minutes	19.9333ms	39.65s
Success rate	0%	26.6667%	0%

- **IDS** taking too much time, I stop it manually (ctrl + C), depth 7(51) ≥ 218257274202 , depth 8(1000) ≥ 4845396852714
  - depth 7(51)≥ 218257274202 → at **depth 7** used **51 minutes** and **at least expanded 218257274202 nodes**
- Genetic algorithm parameters
  - populations size = 5000, generations = 5000
- p=5000, g=500, k=10
  - **OX-cross GA**: 8351.7ms, #ATK = 1.03333, SR=23.3333%
  - **OX-cross GA2**: 6626.2ms, #ATK=0.1, SR=90%

Hill Climbing

Results	random restart, max=999	random restart, max=9999	random restart, max=99999	random restart, max=150000
Average #attacks	0.966667	0.866667	0.9	0.933333
Average running time	1.315s	13.3227s	120.771s	193.100s
Success rate	26.6667%	26.6667%	30%	26.6667%

OX-cross Genetic Algorithm(GA2)

Results	p = 5000, g = 500, k=50	p = 5000, g=1000, k=100	p=10000, g=500, k=50	p=5000, g=500, k=10
Average #attacks	0.166667	0.166667	0.0333333	0.1
Average running time	6881.17 ms	15727.5 ms	6690.1 ms	6626.2 ms
Success rate	83.3333%	83.3333%	96.6667%	90%

20-Queen Result

Results	IDS(1 iteration)	Hill Climbing	Genetic Algorithm(GA2)
Average #attacks	0	0.666667	0.233333
Average running time	1000minutes	662.767 ms	1002.13ms
Success rate	0%	36.6667%	76.6667%

- **IDS** expanded too unnecessary nodes

    - At **depth 11**(19) ≥ **36103179716** , **depth 12**(76) ≥ **123079546646**, **depth 13**(200) ≥ **340072748633**, **depth 14**(484) ≥ **771655937979**, **depth 15**(1000) ≥ **1228753776575**, stop cause exceed 1000 minutes
    - depth 11(19)≥ 36103179716 → at **depth 11** used **19 minutes** and **at least expanded 36103179716 nodes**
    - If skip unnecessary trials (max_depth < N, impossible to get solution), only use **4ms** to get solution.
- **HC**: max_depth=9999, random restart
- p=2500, g=200, k=50
    - **GA**: 3078.97 ms, #ATK = 0.666667, SR = 33.3333%

Conclusion

IDS

- **Completeness** → Yes, **Optimality** → Yes, **Time complexity** → depend on problem size, **Space complexity** → depend on problem size
- When **problem size** is **small**, it is **fast** and get **optimal** if exists, **but** when **problem size** became **large** then it will consume very much **time** and **space**(expanded nodes)

Hill climbing

- **Completeness** → No, **Optimality** → No, **Time complexity** → depends on parameter, **Space complexity** → O(1)
- If problem size is small → few local minimal → more chance to get optimal
- If problem size is large → much local minimal → depends on luck to get optimal
- Only request O(1) space to store current node and check neighbours
- Time consumption depends on parameters and luck
    - If **lucky** → get optimal faster → end
    - If **not lucky** → run until parameters (max depth, random restart)

Genetic Algorithm

- **Completeness** → No, **Optimality** → No, **Time complexity** → depends on parameters and lucky, **Space complexity** → depends on parameters
- **Request** a **well design** on all functions(selection, tournaments, crossover, mutation,...)
    - Bad design will lead to lower chance to get an optimal, while good design can use less parameters(populations, generations,...) to get an optimal with a high chance
- **Populations** influence most for getting an optimal, because more populations can have more chance to get to an global minimal point → end faster

- **Generations** may help to get an optimal, but depends on lucky mutation
- **Tournament selection** with **small k** may help to get optimal, because have less chance always selected the same parent

References

[1] ChatGPT free version. (2025, 28 March of queries). "請給我爬山演算法的雛形","請用一個線程每30秒讓我知道程式還沒掛掉","怎麼快速做到shuffle" Generated using OpenAI. https://chatgpt.com

[2] DeepSeek (2025, 28 Match of queries). "怎麼快速計算n皇后的attacks數","怎麼從populations中快速取得最小的attack數的chromosome""我要可以隨機到的東西是均勻的" Generated using DeepSeek. https://chat.deepseek.com

[3] 丁川康教授（ http://mx.nthu.edu.tw/~ckting/) 人工智慧概論Introduction to Artificial Intelligence 上課講義