Heuristic Search Practices. Course 2020-2021. N-queens with A* and Genetic Algorithms.

Hugo Fonseca Díaz
uo258318@uniovi.es

School of Computing Engineering. University of Oviedo.

Abstract. This is the abstract. TBD.

Keywords: A* · Genetic algorithms · N-queens problem · Heuristic Search.

1 Introduction

In this paper a research will be conducted in order to solve the N-queens search problem using two different approaches. This strategies consist of the use of the A* algorithm and the genetic algorithms.

First, the N-queens problem will be defined. After that, the A* algorithm and its application to this concrete problem will be explained. The same structure will be later applied to the genetic algorithms. Finally, an experimental study will be carried out in order to check the efectiveness of both approaches and the conclusions of the paper will be layed out.

2 N-queens Problem

Introduced in 1848 by chess composer Max Bezzel as the *eight queens puzzle*, it was later solved and extended by Franz Nauck to the *n queens problem*. It is a very well known constraint satisfaction problem, and consists in setting N chess queens in a gameboard of NxN dimensions such as no pair of queens are attacking themselves. In chess, queens can attack other pieces if they are in the same row, column or diagonal as them.

3 A* algorithm

The A* algorithm is an informed search algorithm. Its also an specialization of the BF (Best First) algorithm with a different evaluation function. Before expressing this evaluation function, some concepts must be introduced:

$$f^*(n) = g^*(n) + h^*(n)$$

- $g^*(n)$ is the lowest path cost between the initial node and n.
- $h^*(n)$ is the lowest path cost between n to the nearest target node.
- $f^*(n)$ is the lowest path cost between the initial node to the target node passing through node n.
- $C^* = f^*(initial) = h^*(initial)$ is the cost of the optimal solution.

Once those concepts have been introduced, the A* evaluation function can be defined by:

$$f(n) = g(n) + h(n)$$

- $g(n)$ is the best lowest path cost between the initial node and n obtained during the search until that moment.
- $h(n)$ is a positive estimation of $h^*(n)$, such as $h(n) = 0$ if n is a target node.
- $f(n)$ is an estimation of $f^*(n)$

A* has the following formal properties:

- **Admissibility:** a search algorithm is said to be admissible if it always finds the optimal solution. This means that the heuristic function h must be a positive estimation of the value of $h^*(n)$ for all nodes, that is, $h(n) \leq h^*(n) \forall n$.
- **Dominance:** if h_2 is better informed than h_1, then every node expanded by $A^*(h_2)$ is expanded by $A^*(h_1)$. This means that $A^*(h_2)$ dominates $A^*(h_1)$. For an heuristic function h_2 to be better informed than another one h_1 the following must be true: $h_1(n) < h_2(n) \leq h^*(n)$ for every n non final.
- **Monotony:** an heuristic function $h1$ is monotonous if for every pair of nodes n_1 and n_2 it is true that $h(n_1) \leq h(n_2) + c(n_1, n_2)$ being c the cost of going from n_1 to n_2.

4 Genetic algorithms

Genetic algorithms are metaheuristic search and optimization algorithms and a class of evolutionary algorithms based on the model of natural biological evolution depicted by Charles Darwin in his book *On the Origin of Species*. They perform the evolution of a population, selecting the fittest individuals to become the parents of the next generation. The next generation will be reproduced using crossover and mutation operators.

A genetic algorithm has the following components:

- A **codification scheme**. The genetic algorithm needs some kind of codification in order to represent potential solutions. It can be a chain of binary digits, permutations, vectors, etc.
- A **fitness function**. This function is used to evaluate each individual and assign to them a value used for their selection as potential parents for the next generation. The fitness value is the non-negative value representing the performance of a given individual.
- A way of **generating the initial population**. This method of creating initial populations can be done by the use of an heuristic function or, in the simplest case, making a random population disregarding their level of quality.
- A set of **genetic operators**.
  - **Selection:** operator that chooses a set of individuals from the current generation to be the parents for the next generation.

    - **Crossover:** operator that combines two or more parents to obtain offsprings that most likely will inherit characteristics from their parents.
    - **Mutation:** operator that makes minor random changes in the genetic structure of a chromosome in order to obtain a new range of genetic material.
- Some **parameters**. Like the population size, number of generations, crossover probability, mutation probability, etc.

5 Application of A* algorithm to the N-queens problem

Two ways of modeling the search space have been defined for this problem:

- **Incremental:** in this version, the initial state is the empty gameboard and in each iteration a new queen is placed in a position such as it is not attacking or being attacked by any other queen in the board. This keeps going until the N queens are placed or until there are no locations in the board where the queen can be placed without breaking the constraints.
- **Complete:** in this version, the initial state is a board where the N queens are already placed. However, the queens can be in locations where they are attacking one another. In each iteration a queen is moved to a different position until a state where no pair of queens are attacking themselves is reached. There are two variants:
    - **Queens in first row:** in this variant the n queens are placed in the first row of the board.
    - **Queens in every column:** in this variant the n queens are placed in random positions of the board, so different executions can lead to different results.

Three heuristic functions were implemented for solving the problem:

- **Null heuristic:** heuristic function that always return zero. It is an admissible heuristic.
- **Number of attacking pairs:** heuristic function that estimates de number of necessary movements in order to achieve a solution by means of analysing the number of attacking pairs in a board where there are already N queens placed. It is an admissible and a monotonous heuristic.
- **Probabilistic estimation of solution:** heuristic function that estimates the probability of finding a solution from any given state. It is an admissible heuristic.

6 Applications of genetic algorithms to the N-queens problem

The genetic algorithm implemented for solving the N-queens problem has the following characteristics:

- **Codification scheme:** a chain of permutations that contains the rows in which the queens are located.
- **Fitness function:** implements fitness scaling, that is, to further differenciate the good chromosomes from the bad ones. The way it is done is by substracting from the fitness value of all individuals the fitness value of the worst in that generation.

- **Generating the initial population:** a chromosome is a random permutation of the numbers ranged from one to the size of the board such as no number is repeated.
- A set of **genetic operators**.
  - **Selection:** introduces elitism, that is, the best individual of that generation is carried over the next generation.
  - **Crossover:** implements the OX (Order crossover) operator. This method of crossover implies that the offspring will inherit the order and position of some of the genes of a parent and the relative order of the remaining genes of the other parent.
  - **Mutation:** swaps the content of two random positions of the permutation.
- **Parameters:** $Populationsize = 50$, $mutationprobability = 0.15$, $numberofgenerations = 100$.

7 Experimental Research

The behaviours of the A* and genetic algorithm previously defined are going to be measured next. For this, a modified version of the *aima-java* project available at [1] is used. As the name implies, these algorithms are implemented in the *Java* programming language.

For the enviroment used to test the algorithms the following information is listed:

- **Operating system:** Manjaro Linux *x86-64*
- **Kernel:** 5.7.19-2-MANJARO
- **Host:** 81DE Lenovo ideapad 330-15IKB
- **CPU:** Intel i7-8550U (8) @ 4.000GHz
- **RAM:** 8GB

This experimental research consists of a description of the dataset, a list of the results obtained for both algorithms and a comparison between their behaviours for solving the N-queens problem.

7.1 Dataset

The problems to be resolved are the following:

- **A*:**
  - **Complete version:**
    - **In the first row:** executed one time for each heuristic function and n in range 4, 6, 8, 16 and 32. The following heuristic functions will be tested:
      - *Null heuristic*
      - *Number of attacking pairs*

        - *Probabilistic estimation of solution*

    - **In every column:** executed ten times for each heuristic function and n in range 4, 6, 8, 16 and 32. The average and best execution will be listed. The following heuristic functions will be tested:

        - *Null heuristic*
        - *Number of attacking pairs*
        - *Probabilistic estimation of solution*

- **Incremental version:** executed one time for each heuristic function and n in range 4, 6, 8, 16 and 32. The following heuristic functions will be tested:

    - *Null heuristic*
    - *Number of attacking pairs*
    - *Probabilistic estimation of solution*

– **Genetic algorithm:** executed ten times for each n in range 4, 6, 8, 16 and 32. The average and best execution will be listed.

7.2 A* Results

The results obtained in the executions of A* are the following:

Null heuristic For the *null* heuristic the measurements obtained are listed below in the form of two tables. **Table 1** shows the expanded nodes for every version as well as the best, average and standard deviation for the completed version with queens in every column. **Table 2** shows the same information but for the path cost of the solution.

Expanded

N	Incremental	CompleteFirstRow	ComplEveryCol Best	ComplEveryCol Average	ComplEveryCol Std. Deviation
4	15	70	6	68	58,256
6	149	15259	1667	8873,2	5215,761
8	1965	Not enough resources	1738	7276	7831,915

Table 1. Expanded nodes for the *null* heuristic algorithm

Number of attacking pairs For the *number of attacking pairs* heuristic the results obtained are listed below in the form of two tables. **Table 3** shows the expanded nodes for every version as well as the best, average and standard deviation for the completed version with queens in every column. **Table 4** shows the same information but for the path cost of the solution.

Path cost

N	Incremental	CompetelFirstRow	ComplEveryCol Best	ComplEveryCol Average	ComplEveryCol Std. Deviation
4	4	3	1	2,2	0,919
6	6	5	3	4,2	0,632
8	8	Not enough resources	3	3	0,000

Table 2. Path cost for the *null* heuristic algorithm

Expanded

N	Incremental	CompleteFirstRow	ComplEveryCol Best	ComplEveryCol Average	ComplEveryCol Std. Deviation
4	15	4	2	6,7	4,855
6	149	76	3	30,1	17,527
8	1965	14	11	37,2	24,521
16	Not enough resources	77	9	194	172,059

Table 3. Expanded nodes for the *number of attacking pairs* heuristic algorithm

Probabilistic estimation of solution For the *probabilist estimation of solution* heuristic the results obtained are listed below in the form of two tables. **Table 5** shows the expanded nodes for every version as well as the best, average and standard deviation for the completed version with queens in every column. **Table 6** shows the same information but for the path cost of the solution.

7.3 GA Results

Now the data obtained from the executions of the genetic algorithm will be shown in the form of three tables. **Table 7** shows the parameter values for the genetic algorithm. **Table 8** shows the average and best fitness values, the standard deviation and the number of solutions for each ten executions of the problem.

7.4 Comparison of A* and GA results

Comparison of A* and GA results.

8 Conclusions

Conclusions.

Path cost

N	Incremental	CompleteFirstRow	ComplEveryCol Best	ComplEveryCol Average	ComplEveryCol Std. Deviation
4	4	3	2	2,8	0,632
6	6	6	3	4,1	0,738
8	8	8	4	4,9	0,876
16	Not enough resources	17	7	9	1,333

Table 4. Path cost for the *number of attacking pairs* heuristic algorithm

pop size	50
mutation prob	0.15
n generations	100

Table 5. Parameters of the genetic algorithm

N	Best Fitness	Average Fitness	Std. Deviation	Valid Solutions
4	6	6	0	10
6	15	15	0	10
8	28	28	0	10
16	120	119,5	0,527	5
32	493	492,3	0,949	0

Table 6. Data obtained for the genetic algorithm

References

[1] Github's repository page for the *aima-java* project, `https://github.com/aimacode/aima-java`. Last accesed 1 November 2020.

[2] Author, F.: Article title. Journal **2**(5), 99–110 (2016)

[3] Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016). https://doi.org/10.10007/1234567890

[4] Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999)

[5] Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010)

[6] LNCS Homepage, `http://www.springer.com/lncs`. Last accessed 4 Oct 2017