Searching Algorithms Implementation and Comparison of Romania Problem

Ismail. A. Humied

Assistant Professor of Computer Science, Faculty of Police, Policy Academic, Ministry of interior, Sana'a, Yemen

Abstract

Nowadays, permutation problems with large state spaces and the path to solution is irrelevant such as N-Queens problem has the same general property for many important applications such as integrated-circuit design, factory-floor layout, job-shop scheduling, automatic programming, telecommunications network optimization, vehicle routing, and portfolio management. Therefore, methods which are able to find a solution are very important. Genetic algorithm (GA) is one the most well-known methods for solving N-Queens problem and applicable to a wide range of permutation problems. In the absence of specialized solution for a particular problem, genetic algorithm would be efficient. But holism and random choices cause problem for genetic algorithm in searching large state spaces. So, the efficiency of this algorithm would be demoted when the size of state space of the problem grows exponentially. In this paper, the new method presented based on genetic algorithm to cover this weakness. This new method is trying to provide partial view for genetic algorithm by locally searching the state space. This may cause genetic algorithm to take shorter steps toward the solution. To find the first solution and other solutions in N-Queens problem using proposed method: dividing N-Queens problem into subproblems, which configuring initial population of genetic algorithm. The proposed method is evaluated and compares it with two similar methods that indicate the amount of performance improvement.

Keywords:

genetic algorithm; N-Queens problem; population; crossover; mutation.

1. INTRODUCTION

The permutation problem is a constraint satisfaction problem with a specified numbers of variables. Each variable is assigned a specific value. Every solution can be presented as a permutation of numbers, in which all conflicts are eliminated. For each permutation problem, one or more reasonable solutions are possible [1].

N-Queens problem is one of the best examples of permutation problems. N-Queens problem involves locating n queens on an n x n chessboard such that no queen attacks any other [2]. This problem is one of AI's complex and classic problems which classified in NP problems class [3]. On the chessboard, queens can be located in $\binom{n^2}{n}$ different permutation [4]. Only some of these permutations can be the solutions. For instance, with 8 queens it has 4426165368 different permutations, but only 92 of them are the solutions of the problem [5].

One of the first attempts to use genetic algorithm for solving n- queens problem has been made by Turner and his colleagues in [6], which solving limitation of memory for tackling large number of n, ≥ 200. In [7], Boiikovic and his colleagues applied genetic algorithm to N-Queens in parallel form to increase GA speed. Also in [8], Turky and his colleague used genetic algorithm with the repair function. In [9], Amooshahi and his colleagues presented a new cooperative Particle swarm optimization (PSO) method to solve permutation problems such as N-Queens problem. Also in [10], Sharma and his colleague formulated a new meta-heuristic Cuckoo search in combination with L'evy flights, based on the breeding strategy of some cuckoo species to search on N-Queens problem. In [11], heris and his colleague with the idea of hybridizing genetic algorithm and local search algorithms, try to increase the efficiency of genetic algorithm.

For purposes of finding the solutions, N-Queens problem is classified in 3 classes: 1) finding the first solution, 2) finding some solutions and 3) finding all solutions [9]. This paper aim to present a new method to finding the first solution and finding some solutions for N-Queens problem based on genetic algorithm. Due to primary studies on N-Queens problem, the proposed method has exceeded standard genetic algorithm. The new method begins with pair of individuals as initial population obtained from two subproplems, which includes potential solutions for that problem. Every individual of population is called a chromosome, each chromosome mating to other to obtain first solution, applied several operators for genetic algorithm on this first solution to obtain more solutions.

After introduction, in second part of this paper, the standard genetic algorithm was introduced and in the third part, the modified genetic algorithm was described. In the fourth part, the proposed method was presented and in the next part, the proposed method was evaluate and compare it with standard genetic algorithm and modified genetic algorithm. Finally, part 6, contains concluding remarks.

Manuscript received September 5, 2024
Manuscript revised September 20, 2024
https://doi.org/10.22937/IJCSNS.2024.24.9.12

2. Standard Genetic Algorithm

It is known that the maximum number of queens that can be placed on an n x n chessboard, so that no two attack one another, is n. This problem contains three constraints: 1st, no two queens can share a same column. 2nd, no two queens can share a same row. 3rd, no two queens can share a same diagonal.

Decision variable of this problem is a one-dimensional array of length n. Every permutation of possible values of the decision variables, presents a state of search-space of the problem. Definition of decision variable satisfies the first constraint:

$$A = \{(Q_1, Q_2, \ldots, Q_i) \mid Q_i \in (1,2,\ldots, n)) \quad (1)$$

Where A is decision variable and Q_i is the *i*th cell of decision variable, corresponding *i*th column of chessboard, containing the row of queen's location, the complexity of this problem becomes O(n!). Second constraint is expressed as follows:

$$\forall\ i, j \in \{1, \ldots, n\}, Q_i \neq Q_j \quad (2)$$

Third constraint is also expressed as follows:

$$\forall\ i, j \in \{1, \ldots, n\}, \left|Q_i - Q_j\right| \neq |i - j| \quad (3)$$

A fitness function should return higher values for better states, so, for the N-Queens problem we use the number of *nonattacking* pairs of queens [((n-1) x n)/2], which has a value of 45 for a solution 10-queens problem. This approach leads to O(n) complexity of the fitness function. Genetic algorithm is member of computational method's family which is inspired by evolution. Performance of genetic algorithm is flexible enough to make it applicable to a wide range of problems, such as the problem of placing n queens on n by n chessboard in order that no two queens can attack each other. The space solution is represented as the population, which consists of individuals that are evaluated using the fitness function representing the problem being optimized. The basic structure of a genetic algorithm is shown in Figure1.

In each iteration (generation) of algorithm, a certain number of best-ranking individuals (chromosomes) is selected in the manner to create new better individuals (children). Among the algorithms that are used for selection operation, 'roulette wheel' and 'reminder stochastic sampling' are more significant [12]. In this paper 'roulette wheel' technique is used for selection operation, where each individual is represented by a space that proportionally corresponds to its fitness.

The children are created by some type of recombination (crossover) and they replace the worst-ranked part of the population. After the children are obtained, a mutation operator is allowed to occur and the next generation of the population is created. The process is iterated until the evolution condition terminates.

Genetic algorithm like many of heuristic algorithms, does not guarantee of finding solution because choosing starting point of search and taking steps toward solution have been carried out randomly. In problems like N-Queens that its state space grows exponentially, starting point of search is directly related to the probability of finding solution [3][10] [13].

Genetic Algorithm

generate random initial population
evaluate the fitness of each individual in the population

repeat
  select best-ranking individuals to reproduce
  create new generation through crossover and mutation
  evaluate the individual fitnesses
until (terminating condition)

return best chromosome

Fig. 1: Structure of genetic algorithm [3].

3. Modified Genetic Algorithm

Modified genetic algorithm [11], is the result of collaboration between genetic algorithm and minimal conflicts algorithm. Minimal conflicts algorithm is looking at adjacent space of each candidate to problem's solution and trying to replace current candidate by one of its neighbors which has a better fitness-value. In figure 2, gray areas represent modified sections of standard genetic algorithm. Minimal conflicts operator applied to each candidates after crossover and mutation.

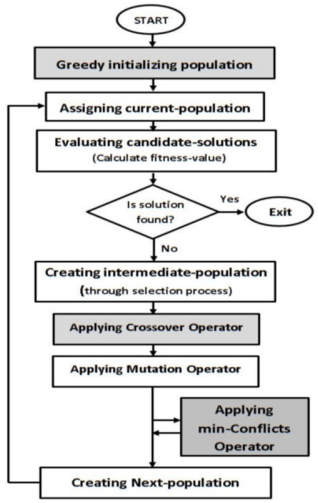

Fig. 2: Flowchart of modified genetic algorithm.

4. The Proposed Algorithm

As it is mentioned before, in N-Queens problem each permutation of possible values of the decision variable can be a candidate to problem's solution. These candidates are also called "chromosomes". A collection of candidates are called "population". Genetic algorithm is consisted of several operators to modify population in several iterations and during these iterations new chromosomes maybe solutions are created. In this part, the new method based on genetic algorithm is introduced to trying increase algorithm's speed of reaching to first solution with simplest single iteration; also it can obtain more solutions applying several operators for genetic algorithm on the first solution as shown in the following states:

4.1. First solution of N-Queens problem

The new method obtains the first solution of N-Queens problem using the following steps:

4.1.1 Generating subproblems of N-Queens

The subproblems in N-Queens obtain by choices two subsets of decision variable "A" in equation (1). First subset start from Q_1 to Q_{Cbest}, where Cbest = floor (n/2) called "local best crossover" which has been found to yield satisfactory results in a number of experiments and computational expenses significantly, the other subset start from $[Q_{Cbest+1}]$ to Q_n, as shown in figure 3(a).

Figure 3(b) shows two subproblems of the 10-queens. The Cbest = 5 so the first subproblem involves getting position of five queens Q_1, Q_2, Q_3, Q_4 and Q_5 into their correct positions. The other subproblem involves getting position of five queens Q_6, Q_7, Q_8, Q_9 and Q_{10} into their correct positions. (Notice that the locations of the other queens in two subpropblems which symbolic * are irrelevant for the purposes of solving problem and moves of those queens don't count towards the cost.). Clearly, the cost of the solution of each subproblem is a lower bound on the cost of the complete problem. The solution can find for every possible subproblem instance—in the example, every possible configuration of the five queens, as describe in the following step.

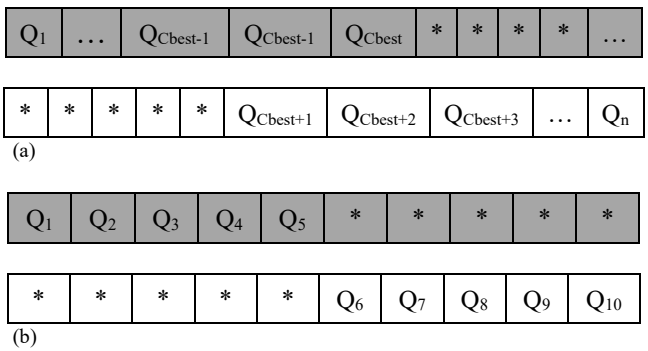

Figure 3: The two subsets of decision variable of the (a) n-queens and (b) 10-queens.

4.1.2 Configuration n queens onto subproblems

After generated two subproblems of N-Queens, the solution can find for every possible subproblem instance. The configuration n queens onto two subproblems can create a pair of chromosomes that can mate to obtain the first solution as describe in the following two cases:

Case 1: The pair of the chromosomes that configured in the manner as figure 4 where number of queens (n) = k x 6 + L; for k = 0,1 , 2, 3, … ; Figure 4(a) shows the two chromosomes where L = 8, in the first chromosome: Q_1 = Cbest, Q_2 = Cbest+2, Q_3 = Cbest+4, Q_4 = Cbest+6, …; and in the other chromosome: Q_n = Cbest+1, Q_{n-1} = Cbest-1, Q_{n-2} = Cbest-3, Q_{n-3} = Cbest-5, … ; figure 4(b) shows the two chromosomes where L = 9, in the first chromosome: Q_1 = Cbest+1, Q_2 = Cbest+3, Q_3 = Cbest+5, Q_4 = Cbest+7, …; and in the other chromosome Q_n = 1, Q_{n-1} = Cbest+2, Q_{n-2} = Cbest, Q_{n-3} = Cbest-2, Q_{n-4} = Cbest-4, … .

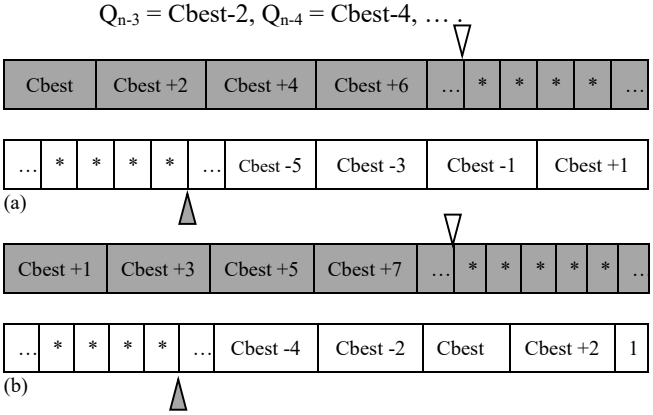

Fig. 4: 1-point crossover (Cbest) cuts pair of the chromosomes from 'break point', when n = k x 6 + L; for k = 0,1 , 2, 3, …; a) L = 8. b) L = 9.

Case 2: The pair of the chromosomes that configured in the manner as figure 5 where number of queens (n) ≠ k x 6 + L; for k = 0,1 , 2, 3, … . Figure 5 shows the two chromosomes, in the first chromosome: Q_1 = n-1, Q_2 = n-3, Q_3 = n-5, Q_4 = n-7, …; and in the other chromosome $Q_{Cbest+1}$ = n, $Q_{Cbest+2}$ = n-2, $Q_{Cbest+3}$ = n-4, $Q_{Cbest+4}$ = n-6, … .

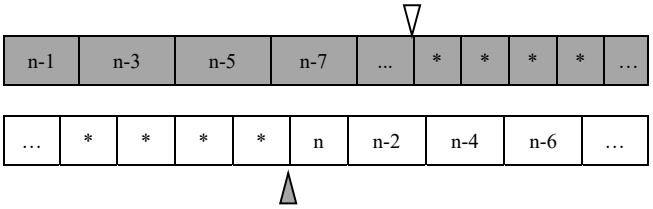

Fig. 5: 1-point crossover (Cbest) cuts pair of the chromosomes from 'break point', when n ≠ k x 6 + L; for k = 0,1 , 2, 3, … .

In the new method crossover operator has 1-point crossover (Cbest) in the pair of the chromosomes that obtained it in case1 or case 2. Then it recombines them to form *first solution*. Figure 6(a) shows the two chromosomes in the 10-Queens obtained using case 2 and configuration at figure 5 because the number of queens $n \neq k \times 6 + L$. Crossover operator in two chromosomes Cbest = 5. Then it recombines them to form the first solution, as shown in Fig. 6(b).

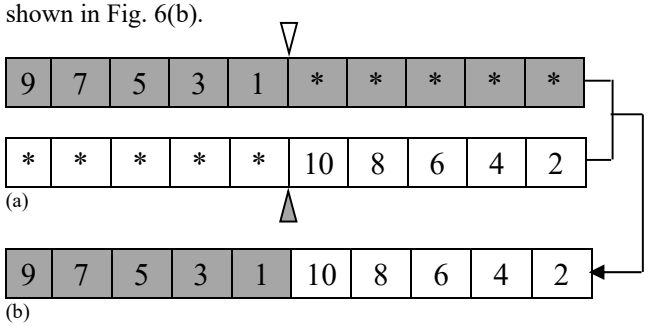

Fig. 6: a) 1-point crossover (Cbest=5) cuts pair of the chromosomes from 'break point'. b) Replaces primary pieces.

4.2. The other solutions of N-Queens problem

To find other solutions of N-Queens problem, the mutation operation can apply repeatedly onto the first solution that obtained in previous subsection. The mutation operation use swapping between two column values (that is queen positions) to create a certain number [n x (n-1)] of individuals (children) which contain other solutions, Figure 7 shows two other solutions using mutation on Fig.6 (b).

4	7	5	3	1	10	8	6	9	2
9	2	5	3	1	10	8	6	4	7

Fig. 7: Two other solutions using mutation operation on Fig.6 (b).

4.3. More solutions of N-Queens problem

Additional the previous solutions the new method can obtain more solutions using [n x (n-1)] individuals (children) that created by mutation operation in previous subsection as initial population instead of random initialization in standard genetic algorithm. Figure 8 shows fourteen different solutions using ninety individuals that created by mutation operation on figure 6 (b) (Notice the number of solutions based on number of runs, iterations, probability of crossover (P_C) and probability of mutation (P_m) as will mention in the next section).

To remember, initializing population is especially important in genetic algorithm and has a significant impact on its efficiency. The initial population can be generating from the subproblems of N-Queens. Every permutation of possible values of the subproblems, presents chromosomes in initial population of the subproblem. Before the first iteration begins, the initial population is assigning so that it is investigating equations 1, 2 and 3.

2	4	6	8	10	1	3	5	7	9
10	5	7	2	4	8	1	9	6	3
4	7	10	3	9	2	5	8	6	1
6	4	2	7	9	3	1	8	10	5
6	8	2	4	9	7	10	1	3	5
6	10	2	5	8	4	1	3	9	7
8	5	3	1	7	10	6	9	2	4
4	2	9	6	10	7	1	3	5	8
7	2	10	6	1	9	5	3	8	4
8	4	9	3	5	10	1	6	2	7
7	3	6	9	1	10	4	2	8	5
1	8	2	7	10	3	5	9	4	6
6	4	9	1	3	10	7	2	8	5
6	8	3	5	9	2	10	7	4	1

Fig. 8: The fourteen different solutions using the individuals that created by ninety mutation operation on figure 6 (b).

5. Experimental Results

The "new method" tested to ensure that performance of it is efficient as expected. The amount of improved efficiency can assess by comparing the results of "new method" with the results of "standard genetic algorithm" and "modified genetic algorithm". According to [11], the upper-bound for iterations in "new method" (4.3) and other methods is considered equal to 50 x n, if the number of iterations in a run of algorithm is exceeded the limit, then the result is a failure and its number of iteration is considered equal to the upper-bound, also population size for "standard genetic algorithm" is equal to 25 x n and for "new method" in 4.3 is equal to n x (n-1). The probability of crossover is equal to 0.7 ($P_C = 0.7$) and the probability of mutation ($P_m = 0.01$).

Table 1 shows in variant number of queens first solutions for "new method" according to 4.1. Table 2 according to 4.1.2 case 1 and table 3 according to 4.1.2 case 2 show the variant number of queens and the "average number of solutions" in "standard genetic algorithm" at first column, in "new method" 4.3 at second column and "number of solutions" in "new method" 4.2 at third column, in 20 runs. All these tables show that the "new method" successfully completed of result in all runs and various numbers of queens. In the otherwise when number of queens is large the results of "standard genetic algorithm" at first column in tables 2 and 3 are not successfully completed so the efficiency of this method would be

demoted when the size of state space of the problem grows exponentially and contains failure.

Also regardless of size of problem, the "new method" in 4.1 reaches to first solution using once mating two chromosomes without any iteration and the "new method" in 4.2 reaches to solutions using simple iterations (mutation operation), finally in 4.3 reaches to solutions using "*average number of iterations*" the same as "the genetic algorithm" because it used the same operator but the "new method" in 4.3 use state space less than genetic algorithm, so the "new method" in 4.3 has less space complexity.

In [11], heris and his colleague results with comparison between "modified genetic algorithm" and "standard genetic algorithm" based on their "*average number of iterations*", shows that the "modified genetic algorithm" is successfully completed. Also shows that the "modified genetic algorithm" reaches to solution in approximately 3 iterations. But average number of iterations for "standard genetic algorithm" increases non-linearly according to size of the problem so these methods are more computational complexity compared with the "new method".

Table 1: First solution according to 4.1 in the "new method".

n	first solution of N-Queens problem
n = 4	3 1 4 2
n = 5	4 2 5 3 1
n = 6	5 3 1 6 4 2
n = 7	6 4 2 7 5 3 1
n = 8	4 6 8 2 7 1 3 5
n = 9	5 7 9 3 8 2 4 6 1
n = 10	9 7 5 3 1 10 8 6 4 2
n = 11	10 8 6 4 2 11 9 7 5 3 1
n = 12	11 9 7 5 3 1 12 10 8 6 4 2
n = 13	12 10 8 6 4 2 13 11 9 7 5 3 1
n = 14	7 9 11 13 1 3 5 10 12 14 2 4 6 8
n = 15	8 10 12 14 2 4 6 11 13 15 3 5 7 9 1
n = 16	15 13 11 9 7 5 3 1 16 14 12 10 8 6 4 2

Beside "*average number of iterations*" which can assess computational complexity of algorithms, the "*average number of solutions*" can be used another criterion which can assess superiorities of algorithms. First column in tables 2 and 3 show that the "standard genetic algorithm" is efficient when uses small size of state space and in the second and third columns show that the efficient of "new method" in 4.3 better than efficient of "new method" in 4.2. On the other hand when size of state space is large then the "standard genetic algorithm" is failure and a good efficient of "new method" in both of 4.2 and 4.3. From another side the "modified genetic algorithm" generate only one solution based on it algorithm at figure 2 and has additional computational complexity due to minimal conflicts operator.

Table 2: Comparing the "average number of solutions" in 20 runs according to 4.1.2(case 1) of the "standard genetic algorithm" and "new method" 4.2, 4.3.

n	Average No. of Solutions in (GA)	Average No. of Solutions in New method (4.3)	No. of Solutions in New method (4.2)
n = 8	57.7	13.1	1
n = 9	67.3	13.6	1
n = 14	0.5	3.7	2
n = 15	0.3	5	3
n = 20	-	6	5
n = 21	-	6	4
n = 26	-	6	4
n = 27	-	4	3
n = 32	-	11	11
n = 33	-	8	10
n = 38	-	12	12
n = 39	-	10	12
n = 44	-	10	11
n = 45	-	13	13

Table 3: Comparing the "average number of solutions" in 20 runs according to 4.1.2(case 2) of the "standard genetic algorithm" and "new method" 4.2, 4.3.

n	Average No. of Solutions in (GA)	Average No. of Solutions in New method (4.3)	No. of Solutions in New method (4.2)
n = 4	2	2	1
n = 5	9.4	2.9	1
n = 6	4	2.1	1
n = 7	38	5.3	1
n = 10	20.1	11.3	3
n = 11	6.5	8.8	3
n = 12	1.3	6.5	3
n = 13	1.5	7	4
n = 16	-	4.8	3

n = 17	-	5.6	4
n = 18	-	6	4
n = 34	-	17	16
n = 36	-	19.3	16
n = 40	-	21	17
n = 42	-	24	21
n = 46	-	31	23
n = 48	-	32	29
n = 50	-	13	13

6. Conclusion

Considering that standard genetic algorithm and modified genetic algorithm are not efficient enough in solving large state spaces of N-Queens problem as the new method. This paper, presented a new method to attempt resolve weakness of these algorithms using subproblems of N-Queens, which obtain the initial population of genetic algorithm. This can accelerate genetic algorithm in order to find the problem's solution, quicker. Also the new method can help genetic algorithm to avoid complexity of iterations and reducing it. According to the results which are declared in section 5, the new method for all states improved efficiency of standard genetic algorithm and modified genetic algorithm in solving N-Queens problem.

REFERENCES

[1] X. Hu, R. C. Eberhart, and Y. Shi, "Swarm intelligence for permutation optimization: a case study of n-queens problem", Proceedings of the IEEE Swarm Intelligence Symposium (SIS '03), pp. 243–246, Indianapolis, Ind, USA, April 2003.

[2] Stuart j. Russell and Peter Norvig, "Artificial Intelligence *A Modern Approach*", *(*3rd *Edition),* Prentice Hall, 2010.

[3] I. Martinjak and M. Golub, "Comparison of Heuristic Algorithms for the N-Queen Problem", Proceedings of the 29th International Conference on Information Technology Interfaces , ITI 2007, pp. 759-764, Cavtat, Croatia, June 25-28,2007.

[4] K. D. Crawford, "Solving the N-Queens Problem Using Genetic Algorithms", In Proceedings ACM/SIGAPP Symposium on Applied Computing,Kansas City, pp. 1039-1047, 1992.

[5] S. Khan, M. Bilal, M. Sharif, M. Sajid and R. Baig, "Solution of n-Queen Problem Using ACO" Multitopic Conference, 2009. INMIC 2009. IEEE 13th International , Islamabad, pp. 1 - 5, 14-15 Dec. 2009.

[6] J. Turner, A. Homaifar, S. Ali, "The n-queens problem and genetic algorithms", in IEEE, pp. 262–267, 1992.

[7] M. Boiikovic, M. Golub, L. Budin, "Solving n-Queen problem using global parallel genetic algorithm", in IEEE, pp. 104 – 107, vol.2, 2003.

[8] A. M. Turky, M. S. Ahmad "Using Genetic Algorithm for Solving N –Queens Problem", in IEEE, pp. 745–747, 2010.

[9] A. Amooshahi, M. Joudaki, M. Imani and N. Mazhari, "Presenting a new method based on cooperative PSO to solve permutation problems: A case study of n-queen problem", in IEEE, pp. 218–222, 2011.

[10] R. G. Sharma, B. Keswani, "implementation of n-queens puzzle using meta-heuristic algorithm (cuckoo search)", in International Journal of Latest Trends in Engineering and Technology, pp. 343- 347, vol. 2, 2013.

[11] J. E. A. heris and M. A. Oskoei, "Modified Genetic Algorithm for Solving n-Queens Problem", in IEEE, 2014.

[12] "D. Whitley, "a genetic algorithm tutorial", statistics and computing, pp. 65-85, 1994".

[13] J. Bell, B. Stevens, "A survey of known results and research areas for n-queens", Discrete Mathematics 309, pp. 1-31, 2009.

AUTHOR'S PROFILE

Ismail Abdullah Humied

He received the B. Sc. degree in Electrical Engineering (Computers & Control), from the Faculty of Engineering, University of Sana'a, Sana'a, Yemen, in 2003, the M.Sc. degree in Electrical Engineering (Computer & Systems), from the Faculty of Engineering, University of Ain-Shams, Cairo, Egypt, in 2009, and the Ph.D. degree in Computer Science from the Faculty of Computers & Information, University of Mansoura, Mansoura, Egypt, in 2012. He is having 3 years of experience in teaching. He is working as an Assistant Professor in Policy Academic, Ministry of interior, Sana'a, Yemen. His research interests and activities cover the Digital Images Processing, Computer Organization and Architecture, Intelligent Control Systems, Artificial Intelligence, Modeling and Simulation, Operating Systems.