Hindawi
Journal of Optimization
Volume 2017, Article ID 5650364, 8 pages
https://doi.org/10.1155/2017/5650364

Research Article

A Genetic Algorithm Based Approach for Solving the Minimum Dominating Set of Queens Problem

Saad Alharbi[1] and Ibrahim Venkat[2]

[1]*Computer Science Department, Taibah University, Medina, Saudi Arabia*
[2]*School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia*

Correspondence should be addressed to Saad Alharbi; salharbi20@gmail.com

Received 20 February 2017; Revised 24 April 2017; Accepted 3 May 2017; Published 4 June 2017

Academic Editor: Gexiang Zhang

In the field of computing, combinatorics, and related areas, researchers have formulated several techniques for the Minimum Dominating Set of Queens Problem (MDSQP) pertaining to the typical chessboard based puzzles. However, literature shows that limited research has been carried out to solve the MDSQP using bioinspired algorithms. To fill this gap, this paper proposes a simple and effective solution based on genetic algorithms to solve this classical problem. We report results which demonstrate that near optimal solutions have been determined by the GA for different board sizes ranging from 8×8 to 11×11.

1. Introduction

The MDSQP which is otherwise known as the chess covering problem is one of the well-known chessboard problems which has been receiving continued interests by researchers including the computational intelligence domain. In the field of graph theory, specifically the study of dominating sets had addressed this problem even in the early 1970s. In $n \times n$ chessboard, cells are organized as n rows and n columns. In chess layman terms, a queen (say Q) placed on a square will dominate all squares associated with the row, column, and two diagonals with reference to Q. The idea behind this problem is to find the minimum number of queens required to dominate the whole chessboard. Domination here refers to the coverage of all the possible squares being attacked by those queens including the square dominated by the respective queens. Previous researches have been trying to find the domination number $\gamma(Qn)$ for the n-queens problem using mathematical and combinatorial approaches [1–8]. Many formulations have been derived and presented in these approaches. It has been observed that limited efforts have been carried out to employ evolutionary algorithms to address the MDSQP. In fact, most of the studies in the literature were devoted to address the original n-queens problem which is called a nonattacking problem. The results of such studies have demonstrated that evolutionary algorithms are capable of outperforming other principled approaches such as backtracking to solve such problems. However, these studies have focused only on the nonattacking n-queens problem, whereas, to the best of our knowledge, the MDSQP has been least considered using bioinspired evolutionary algorithms.

Evolutionary algorithms have been proved to be successful for solving and optimizing a wide range of complex problems including combinatorial ones, such as the one studied here, within reasonable computing time [9]. Generally speaking, evolutionary algorithms can be classified into two main categories which are swarm intelligence based approaches and classical evolutionary approaches. Swarm intelligence (SI) approaches are inspired from the social natural behavior such as animals group behavior when searching for food or avoiding certain threat [10]. Several SI approaches with reference to the literature include Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO). ACO have been proved to efficiently solve several optimization problems [11]. PSO has also demonstrated good results in

Table 1: Domination number for some n values pertaining to the MDSQP.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\gamma(\mathbf{Qn})$	1	1	1	3	3	4	4	5	5	5	5	7	7	≤ 8	≤ 9	≤ 9	9	9

optimizing various complex problems such as the supply chain management (e.g., [12]) and shortest path problems [13, 14].

The classical evolutionary algorithms, on the other hand, including genetic algorithms, were successfully adopted in a wide range of optimization problems. Genetic algorithm is one of the most commonly used evolutionary algorithms in the literature which was first proposed in the 1970s [15]. GA was inspired from the natural process of searching and selection process which leads to the survival of the fittest individuals [15]. Researchers in the literature have demonstrated the efficiency of adopting GA in solving various optimization problems such as data mining [16] and network traffic control [17]. In fact, GA have been adopted in the literature solely to solve some problems such as [17, 18] and were hybridized with other approaches for more enhanced results. Commonly, GA is hybridized with local and heuristic search approaches such as Cuckoo search [19] and Tabu search [20]. Furthermore, various researchers hybridized it with other evolutionary approaches like dynamic programming as in [21].

Therefore, this paper intends to fill these gaps and present a bioinspired genetic algorithm (GA) to solve the MDSQP problem by considering board sizes gradually from the typical 8×8 chessboard up to a 11×11 board. The proposed approach gradually increments the number of queens placed on the board from just below the lower bounds suggested in the literature until an optimal solution has been determined (i.e., all the squares of the chessboard get dominated). The remaining paper is organized as follows: Section 2 discusses related work; Section 3 formulates the problem; Section 4 presents the evolutionary mechanism of the proposed technique; results are discussed in Section 5 and the paper is finally concluded with insights for future work in Section 6.

2. Related Work

Vast amount of efforts have been devoted to investigate chessboard problems. Problems related to chessboard can mainly be classified to the original n-queen problem and the dominating set (i.e., the covering problem). The original n-queen problem can be defined as placing n queens on $n \times n$ chessboard in an optimal way such as none of the queens should attack each other [22, 23]. It was first introduced in 1850 [24]. Numerous researches have been performed to solve this problem using different approaches; most of them adopted mathematical approaches and graph theory such as [25–27]. The study of such a problem may benefit in the understanding of various applications such as traffic control, deadlock prevention, and parallel memory storage [28]. The complexity of the n-queen problem is known as exponential where it gets more complex with large n values. Various approaches were adopted to solve this problem such as backtracking [29], neural networks [30–32], and evolutionary and optimization algorithms [33–42].

The domination set problem on the other hand has received less attention by computer science researchers. In fact, various researches have been devoted to investigate such a problem but were mainly adopting mathematical models. Burger and Mynhardt pointed out that the queens dominating problem is one of the most difficult chessboard problems [43]. It is commonly defined as finding the minimum numbers of queens required to cover all the squares of $n \times n$ chessboard. This number is termed as the domination number and referred by the notation $\gamma(Qn)$. Researchers have arrived at upper and lower bounds since early 1970s [3]. Many formulations and values were derived and found especially for small values of n [3, 43–45]. Table 1 shows some of these values. Figure 1 shows a brief taxonomy of typical chessboard problems. For instance, several studies were conducted to find the domination number of the normal domination [8]. In these studies, it is required to find the minimum queens to be optimally placed on $n \times n$ chessboard that will cover all squares in the board. However, many researchers considered the domination set problems under several variations. For instance, many studies were conducted to investigate the independent domination number which can be defined as the number of queens which do not attack each other and cover the entire $n \times n$ chessboard [3]. Various values and formulations were derived and proved in a mathematical context. Furthermore, the "diagonal queens domination problem" was frequently stated in the literature. In such a problem, it is required to find the number of minimum queens placed in the main diagonal and cover the entire $n \times n$ chessboard [3, 4]. Cockayne has also tried to find the domination number of the queens placed on a single column [3]. Furthermore, many researchers carried out studies to solve the problem of the domination set on a toroidal chessboard [1]. In such studies, it is required to find out the minimum number of queens that would cover the entire squares in the torus. Bozóki et al., on the other hand, shed the light on a different problem called "domination of the rectangular queens graph" where it is required to find the domination number of queens in a rectangular chessboard ($n \times m$) in which the number of squares in column and rows are different [46]. They derived lower bounds for small values of n and m particularly $4 \leq m \leq n \leq 18$.

As seen above, the queens domination set problem has been enormously studied especially using mathematical methods. However, literature still lacks the adoption of computational models particularly optimization approaches to solve this problem. Limited articles were found technically considering the problem. For instance, Fernau tried to analyze the complexity of this problem using backtracking, dynamic programming on subsets, and dynamic programming on path decomposition [47]. He indicated that the

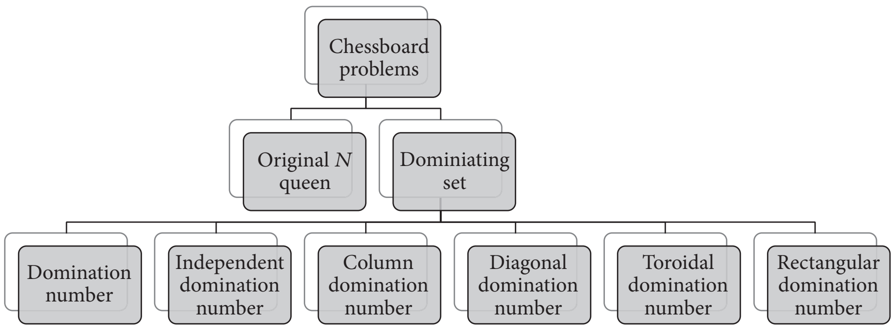
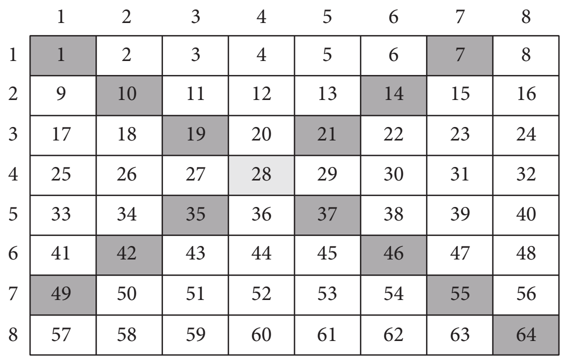

Chessboard problems
Original N queen
Dominiating set
Domination number
Independent domination number
Column domination number
Diagonal domination number
Toroidal domination number
Rectangular domination number

Figure 1: Basic taxonomy of chessboard problems.



	1	2	3	4	5	6	7	8
1	1 ▨	2	3	4	5	6	7 ▨	8
2	9	10 ▨	11	12	13	14 ▨	15	16
3	17	18	19 ▨	20	21 ▨	22	23	24
4	25	26	27	28 ▨	29	30	31	32
5	33	34	35 ▨	36	37 ▨	38	39	40
6	41	42 ▨	43	44	45	46 ▨	47	48
7	49 ▨	50	51	52	53	54	55 ▨	56
8	57	58	59	60	61	62	63	64 ▨

Figure 2: The proposed labelling scheme of a typical 8×8 chessboard.

complexity of the dominating queen problem is a challenging problem. On the other hand, Mohabbati-Kalejahi et al. (2012) presented a hybrid Imperialist Competitive Algorithm (ICA) with a local search algorithm in order to solve the nonattacking queens based domination problems [48]. The proposed hybrid algorithm was compared with the basic ICA in the two problems and the results indicated an improved performance in terms of average runtime and average fitness value. The solution was formulated for the nonattacking n queens problem. However, for the domination problem less details and information have been provided pertaining to the implementation details of the proposed algorithm. For example, crucial components of an evolutionary algorithm such as the formulation of a fitness function have not been presented in detail. Therefore, this work is intended to fill this gap in the literature by adopting a typical bioinspired genetic algorithm (GA) to solve the *Minimum Dominating Set of Queens Problem* (MDSQP). The proposed algorithm is described in the following sections.

3. Problem Formulation

In order to solve the problem and present it programmatically we firstly labelled all squares in the chessboard by sequencing numbers starting gradually from 1 to $n \times n$ starting from the top most left square to the bottom right most square. Figure 2 shows an example of the proposed labelling schema with reference to the commonly used 8×8 chessboard. In addition, all squares in the board have a location (analogous to the norms of the analytical geometry) in the (x, y) coordinate form, where x denotes the row number and y denotes the column number. For example, the location of the square 28 in Figure 2 is (4, 4). This location can be used to determine the dominated squares for a queen placed in a particular square. Let G be a set of chessboard squares $G = \{p_1, p_2, \ldots, p_{n\times n}\}$ and let S_q be the set of dominated squares by queen q for the case of i number of queens. Then it follows that

$$S_{qi} = \{p_1, p_2, \ldots, p_{n\times n}\}. \quad (1)$$

Defining a fitness function for this problem firstly requires defining a method to identify dominated squares for all queens on the board. For a queen placed on the location (x, y), a square n will be $n \in S_q$ if at least one of the following conditions holds:

$$x = xj \quad \forall j \ 1, 2, \ldots, n, \quad (2)$$

$$y = yj \quad \forall j \ 1, 2, \ldots, n, \quad (3)$$

$$\begin{aligned} xj &= x - j, \\ y &= y - j \\ &\qquad \forall j \ 1, 2, \ldots, x - 1, \end{aligned} \quad (4)$$

$$\begin{aligned} xj &= x + j, \\ yj &= y - j \\ &\qquad \forall j \ 1, 2, \ldots, y - 1, \end{aligned} \quad (5)$$

$$\begin{aligned} xj &= x - j, \\ yj &= y + j \\ &\qquad \forall j \ 1, 2, \ldots, x - 1, \end{aligned} \quad (6)$$

$$\begin{aligned} xj &= x + j, \\ yj &= y + j \\ &\qquad \forall j \ 1, 2, \ldots, n - 1. \end{aligned} \quad (7)$$

Equations (2) and (3) represent the dominated squares in terms of rows and columns, respectively. The rest of the equations from (4) to (7) represent the squares dominated

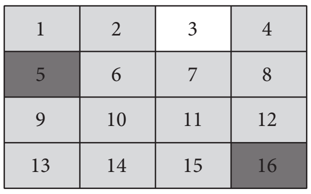


Figure 3: A typical example showing the proposed coding scheme.

by a queen placed in (x, y) for the diagonal directions. Specifically the dominated squares of the two diagonals pertaining to the four sides (upper left, lower left, upper right, and lower right). For example, squares 19, 10, and 1 in Figure 2 are instances of dominated squares by the queen placed on the square 28 and this case has been modeled using a logical algebraic expression using (4). The main objective is to maximize the number of dominated squares in a chessboard by k queens and finding optimal locations of these k queens is quite challenging owing to the complexity of permutations and combinations involved. Therefore, this is measured by the following fitness function:

$$f(x) = \frac{|S|}{|G|}, \tag{8}$$

where

$$S = s_{q_1} \cup s_{q_2} \cup \cdots \cup s_{q_k}. \tag{9}$$

The value of this function approaches 1 when all the squares of the chessboard get dominated at least once. In other words, a fitness value <1 could cohesively indicate that there are some squares or at least 1 square which has not been dominated by any ith queen (s_{q_i}).

4. Formulation of the Problem Using Genetic Algorithms

The proposed GA formulation has been modeled to solve the $n \times n$ chessboard. The overall structure of the proposed GA can be described in Figure 3.

4.1. Coding. Individuals have been coded in a matrix of order $k \times m$. Intuitively $k \times m$, the product of the columns and rows, represents the population size (number of possible candidate solutions). The elements of the matrix represent m bit binary encodings of the labels (each square has been labelled as discussed before) of the queens being placed on the chessboard. The following matrices show typical examples for the case of few possible candidate solutions

$$A = \begin{bmatrix} 1100110 & 0011010 \\ 0101100 & 1100010 \end{bmatrix}. \tag{10}$$

Furthermore, the dominated square set has also been coded in a matrix S as shown below; the total number of unique dominated squares is represented by the number of rows.

$$S = \begin{bmatrix} 1110010 \\ 0000001 \end{bmatrix}. \tag{11}$$

The above matrix shows an example instance of a dominated set containing two squares in the chessboard. Furthermore, let us assume that we would like to encode only one candidate solution in a 4×4 chessboard. Figure 3 shows that two queens are placed in the squares labelled 6 and 16. By converting the decimal values of these squares into 8-digit binary code and taking into consideration having two queens, therefore, we get a 1×2 matrix which is shown below:

$$A = \begin{bmatrix} 00000110 & 00010000 \end{bmatrix}. \tag{12}$$

Furthermore, to identify the dominated squares by the two queens placed in the chessboard, (2) to (7) must be applied. Accordingly, Figure 3 shows that all squares in the chessboard except label 3 are dominated by the two queens. Therefore, the dominated set S is as follows:

$$S = \{1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}. \tag{13}$$

By converting the elements of S into 8-digit binary code we will get the following dominated set matrix S:

$$S = \begin{bmatrix} 00000001 \\ 00000010 \\ 00000011 \\ \vdots \\ \vdots \\ 00010000 \end{bmatrix}. \tag{14}$$

Illustration of typical crossover operations is as follows:

$$\begin{matrix} 110|\,0100 & \Longrightarrow & 1101110 \\ 001|\,1110 & & 0010100 \end{matrix} \tag{15a}$$

$$\begin{matrix} 00|\,11111 & \Longrightarrow & 000\mathbf{1}000 \\ 11|\,00000 & & 111111\mathbf{0}. \end{matrix} \tag{15b}$$

4.2. Initial Population and Fitness Evaluation. A random population consisting of 100 individuals (possible candidate solutions) is generated at the initialization stage of the GA making sure that queens are placed in different squares in the board. The domination rate of the queens is computed using the fitness function as modeled in Section 2. The selection criteria of our proposed techniques are as follows. Individuals are sorted based on their fitness value and the top 50% of the population (candidates with the potential fitness capability) will be allowed to survive and will be considered into the next generation. Candidates whose fitness capability is below 50% will be discarded (they will not be allowed to survive). Subsequent to this selection process, the classical roulette-wheel selection method has been adopted for selecting individuals that will be mated for reproduction.

4.3. Crossover and Offspring Generation. The new generation is produced by implementing the one-point crossover between two parents. The steps involved in the proposed crossover operator can be described as follows:

(i) Two individuals are selected according to the selection operator.

(ii) Generate a random crossover point in the two parents.

(iii) Exchange genetic materials after the crossover point.

(iv) Validate new individuals so that the individuals are retained within the boundaries of the chessboard.

(v) If an offspring's location happens to be to zero owing to stochastics, generate a random adjustment point.

(vi) Change the value of the adjustment position to 1.

(vii) If an offspring's location is larger than $n \times n$, based on the chessboard size change the value of the digits of the chromosome causing this issue to 0.

Equation (15a) shows an example of a crossover operation between two parents to produce two children (offspring); (15b) shows an example of a crossover operation to illustrate the validation and adjustment process.

Furthermore, we have also implemented mutation with the probability of 0.05. In mutation, we simply select two random positions for the selected individuals and substitute the values of these positions to their complements (i.e., change zero to one and vice versa). In order to avoid the production of individuals outside the boundaries of the chessboard, the same crossover validation and adjustment mechanism have been found to be efficient.

5. Experiments and Results

The proposed GA has been implemented using MATLAB and experimented using a 2.6 Ghz PC enabled with Intel i5 processor installed with Microsoft Windows 8 operating system. It was independently tested using a different size of the chessboard starting from the common chessboard size 8×8 to 11×11. In each test instance the number of queens placed in the chessboard is gradually incremented starting from the lower bounds reported in the literature until the maximum fitness has been reached. Furthermore, the proposed GA was iterated 1000 times for each instance and the best fitness values were recorded. The results in this contribution are reported after running the GA for five times for each test instance. Table 2 shows a summary of the maximum fitness reached by the GA for the value of $n = 8$ to $n = 11$ as well as the queen numbers placed in the chessboard. The results obtained were promising and close to upper bounds reported in the literature for MDSQP.

It can be noticed that the GA could cover 88% and 95% of the 8×8 chessboard by placing three and four queens on the board, respectively. The GA was also able to find the optimal solution for the same size board when placing five queens meaning that the domination number of 8×8 chessboard is five (i.e., $\boldsymbol{\gamma}(\mathbf{Qn})$). These findings compare well with the results reported in the literature. As the proposed GA is considered to be a pioneer on the literature considering the MDSQP, the obtained results will be compared with the values in the literature which have been derived mathematically. Table 3 shows the optimal domination numbers obtained by the GA compared with these. Table 2 shows also that the GA was able to dominate 99% of the 9×9 chessboard when five queens have been placed in the board, although five is derived as the domination number for a 9×9 chessboard ($\boldsymbol{\gamma}(\mathbf{Qn}) = \mathbf{5}$). However, an optimal solution was reached by increasing number of queens placed on the board to six ($\boldsymbol{\gamma}(\mathbf{Qn}) = \mathbf{6}$) (see Table 3). Similarly, five was reported as the domination number for the case of 10×10 and 11×11 chessboard; however, the GA was not able to cover the whole board. In fact, only 95% of the board was covered when $n = 10$ and five queens were placed ($k = 5$) (see Table 2). The domination percentage has been improved slightly when adding one queen ($k = 6$) to 10×10 board. However, an optimal solution was found when placing seven queens to the board in the case of n =10, 11 ($\boldsymbol{\gamma}(\mathbf{Qn}) = \mathbf{7}$). All solutions produced were graphically simulated by the GA at the end of the execution. Figure 4 shows the simulations pertaining to the optimal solutions of the proposed GA for the case of $n = 8$ to 11. The red circles (dots) represent the queens and the blue circles represent the dominated squares in a typical $n \times n$ chessboard. It also shows the fitness value of the presented solution corresponding to the generation number of the solution. In addition, the performance of the proposed GA solution was measured after the execution of each test instance. The results indicated that the performance in terms of fitness value is increasing over iterations. Due to the space limitation we present the performance results of the 10×10 chessboard in Figure 5. It can clearly be observed that the fitness values in all the cases have been improved when the iterations tend to approach their upper limits.

It can be seen that the solutions yielded by the proposed technique could yield a good domination performance (95%

Table 2: Progression of fitness values: n is the board size and k is number of queens placed on the board.

n	k	Fitness value
8	3	0.88
	4	0.95
	5	1
9	5	0.99
	6	1
10	5	0.95
	6	0.97
	7	1
11	6	0.97
	7	1

Table 3: The obtained results by the GA and domination numbers $\boldsymbol{\gamma}(\mathbf{Qn})$ reported by the literature.

n	8	9	10	11
GA	5	6	7	7
Literature	5	5	5	5

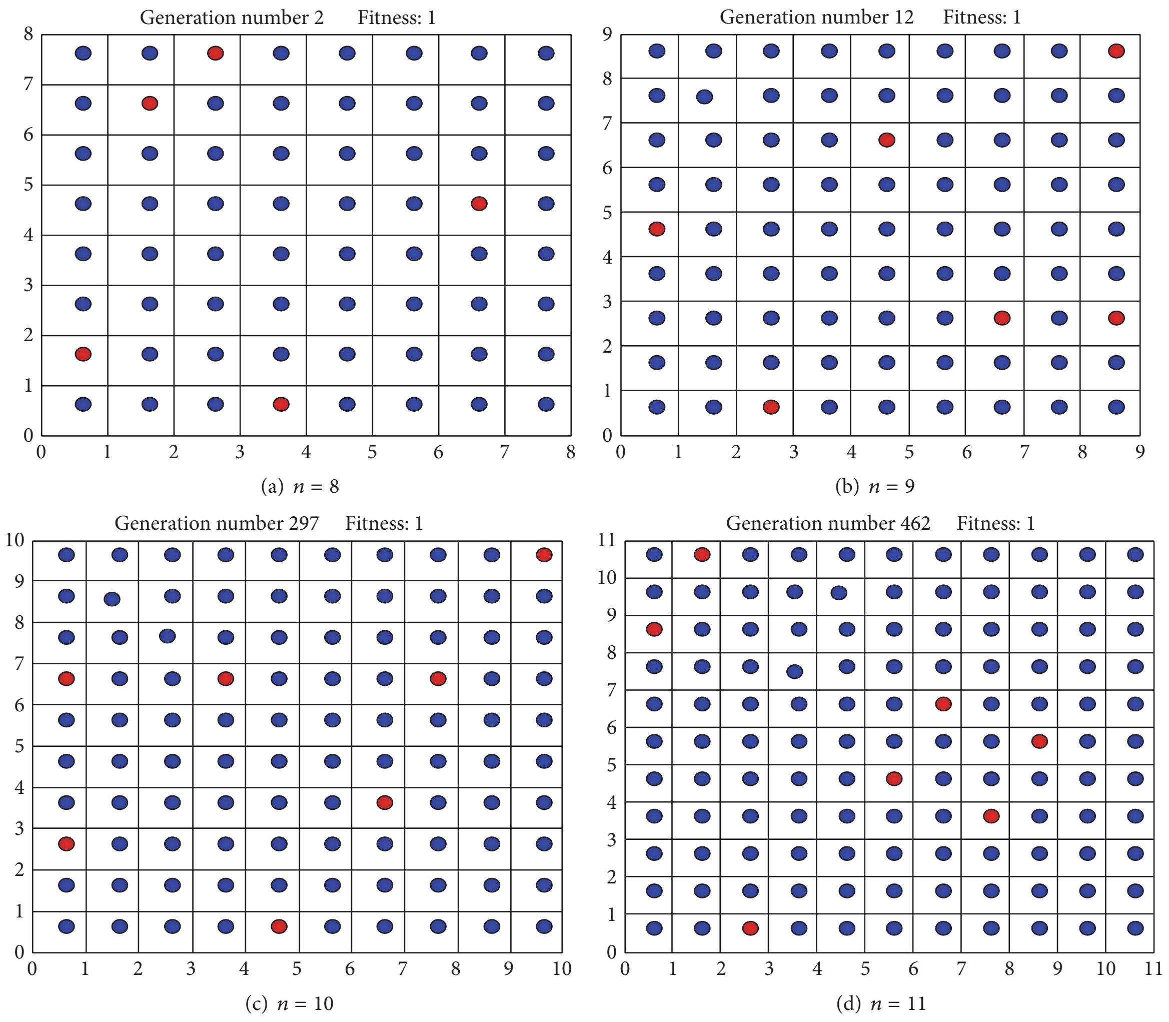

Generation number 2 Fitness: 1
(a) $n = 8$
Generation number 12 Fitness: 1
(b) $n = 9$
Generation number 297 Fitness: 1
(c) $n = 10$
Generation number 462 Fitness: 1
(d) $n = 11$

FIGURE 4: Optimal solutions yielded by the proposed GA based technique. Red dots represent queens and blue dots represent squares dominated by the queens.

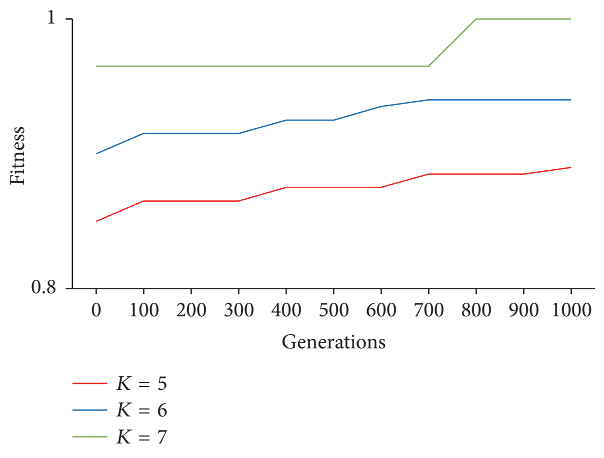


FIGURE 5: Performance of the GA for a typical case; $n = 10$ and $k = 5, 6$ and 7.

to 99%) even for larger board sizes, particularly for the cases of $n = 10$ and $n = 11$.

6. Conclusion

In this paper, we have proposed a *genetic algorithm* (GA) based technique to solve the classical *Minimum Dominating Set of Queens Problem* (MDSQP). Experimental results demonstrate that the proposed GA based solution was able to find optimal to near optimal solutions for even large sizes of boards ($n = 10$ and $n = 11$). For the case of a typical 8×8 chessboard, the proposed solution yielded a perfect optimal solution on a par with the solutions reported in the literature. Furthermore, experimental results also systematically demonstrate that the average fitness value approached towards the maximum over subsequent iterations. These results will be a basis for our future work where we intend to deploy other bioinspired computing techniques such as membrane computing.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] A. Burger, C. Mynhardt, and W. D. Weakley, "The domination number of the toroidal queens graph of size 3k x 3k," *Australasian Journal of Combinatorics*, vol. 28, pp. 137–148, 2003.

[2] A. P. Burger and C. M. Mynhardt, "An improved upper bound for queens domination numbers," *Discrete Mathematics*, vol. 266, no. 1–3, pp. 119–131, 2003.

[3] E. J. Cockayne, "Chessboard domination problems," *Annals of Discrete Mathematics*, vol. 48, pp. 13–20, 1991.

[4] E. J. Cockayne and S. T. Hedetniemi, "On the diagonal queens domination problem," *Journal of Combinatorial Theory, Series A*, vol. 42, no. 1, pp. 137–139, 1986.

[5] T. Kikuno, N. Yoshida, and Y. Kakuda, "A linear algorithm for the domination number of a series-parallel graph," *Discrete Applied Mathematics*, vol. 5, no. 3, pp. 299–311, 1983.

[6] A. Klobučar, "Domination numbers of cardinal products P_6 × P_n," *Mathematical Communications*, vol. 4, no. 2, pp. 241–250, 1999.

[7] N. Sari and D. IH Agustin, *On the Domination Number of Some Graph Operations*, 2016.

[8] S. S. Venkatesan and Venkatesan S., "Tight lower bounds for connected queen domination problems on the chessboard," https://arxiv.org/abs/1608.02531.

[9] B. Doerr, A. Eremeev, C. Horoba, F. Neumann, and M. Theile, "Evolutionary algorithms and dynamic programming," in *Proceedings of the 11th annual conference on genetic and evolutionary computation*, pp. 771–778, ACM, Québec, Canada, 2009.

[10] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: algorithms and applications," *Swarm and Evolutionary Computation*, vol. 33, pp. 1–17, 2017.

[11] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," *Artificial Intelligence Review*, vol. 42, pp. 21–57, 2014.

[12] R. S. Kadadevaramath, J. C. H. Chen, B. L. Shankar, and K. Rameshkumar, "Application of particle swarm intelligence algorithms in supply chain network architecture optimization," *Expert Systems with Applications*, vol. 39, no. 11, pp. 10160–10176, 2012.

[13] Y. Marinakis, A. Migdalas, and A. Sifaleras, "A hybrid particle swarm optimization—variable neighborhood search algorithm for constrained shortest path problems," *European Journal of Operational Research*, vol. 261, no. 3, pp. 819–834, 2017.

[14] M. Akhand, S. Hossain, and S. Akter, "A comparative study of prominent particle swarm optimization based methods to solve traveling salesman problem," *International Journal of Swarm Intelligence and Evolutionary Computation*, vol. 5, no. 139, p. 2, 2016.

[15] M. Srinivas and L. M. Patnaik, "Genetic algorithms: a survey," *Computer*, vol. 27, no. 6, pp. 17–26, 1994.

[16] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine Learning*, vol. 3, no. 2-3, pp. 95–99, 1998.

[17] A. Barolli, M. Takizawa, F. Xhafa, and L. Barolli, "Application of genetic algorithms for QoS routing in mobile ad-hoc networks: A survey," in *Proceedings of the 5th International Conference on Broadband Wireless Computing, Communication and Applications (BWCCA '10)*, pp. 250–259, Fukuoka, Japan, November 2010.

[18] B. M. Varghese and R. J. S. Raj, "A survey on variants of genetic algorithm for scheduling workflow of tasks," in *Proceedings of the 2nd International Conference on Science Technology Engineering and Management (ICONSTEM '16)*, 2016.

[19] A. Abu-Srhan and E. Al Daoud, "A hybrid algorithm using a genetic algorithm and cuckoo search algorithm to solve the traveling salesman problem and its application to multiple sequence alignment," *International Journal of Advanced Science and Technology*, vol. 61, pp. 29–38, 2013.

[20] A. M. Allakany, T. M. Mahmoud, K. Okamura, and M. R. Girgis, "Multiple constraints QoS multicast routing optimization algorithm based on Genetic Tabu Search Algorithm," *Advances in Computer Science*, vol. 4, no. 3, 2015.

[21] D. T. Pham, T. T. B. Huynh, and T. L. Bui, "A survey on hybridizing genetic algorithm with dynamic programming for solving the traveling salesman problem," in *Proceedings of the International Conference on Soft Computing and Pattern Recognition (SoCPaR '13)*, pp. 66–71, Hanoi, Vietnam, December 2013.

[22] I. Rivin, I. Vardi, and P. Zimmermann, "The *n*-queens problem," *The American Mathematical Monthly*, vol. 101, no. 7, pp. 629–639, 1994.

[23] A. Bruen and R. Dixon, "The *n*-queens problem," *Discrete Mathematics*, vol. 12, no. 4, pp. 393–395, 1975.

[24] C. Letavec and J. Ruggiero, "The *n*-queens problem," *INFORMS Transactions on Education*, vol. 2, no. 3, pp. 101–103, 2002.

[25] R. Sosic and J. Gu, "3,000,000 Queens in less than one minute," *ACM SIGART Bulletin*, vol. 2, no. 2, pp. 22–24, 1991.

[26] M. R. Engelhardt, "A group-based search for solutions of the *n*-queens problem," *Discrete Mathematics*, vol. 307, no. 21, pp. 2535–2551, 2007.

[27] Z. Szaniszlo, M. Tomova, and C. Wyels, "The *N*-queens problem on a symmetric Toeplitz matrix," *Discrete Mathematics*, vol. 309, no. 4, pp. 969–974, 2009.

[28] J. Bell and B. Stevens, "A survey of known results and research areas for *n*-queens," *Discrete Mathematics*, vol. 309, no. 1, pp. 1–31, 2009.

[29] R. Sosič and J. Gu, "Efficient local search with conflict minimization: a case study of the n-queens problem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 6, no. 5, pp. 661–668, 1994.

[30] T. Kwok and K. A. Smith, "Experimental analysis of chaotic neural network models for combinatorial optimization under a unifying framework," *Neural Networks*, vol. 13, no. 7, pp. 731–744, 2000.

[31] N. Funabiki, Y. Takenaka, and S. Nishikawa, "A maximum neural network approach for N-queens problems," *Biological Cybernetics*, vol. 76, no. 4, pp. 251–255, 1997.

[32] J. Mandziuk and B. Macuk, "A neural network designed to solve the N-queens problem," *Biological Cybernetics*, vol. 66, no. 4, pp. 375–379, 1992.

[33] Z. Wang, D. Huang, J. Tan, T. Liu, K. Zhao, and L. Li, "A parallel algorithm for solving the n-queens problem based on inspired computational model," *BioSystems*, vol. 131, pp. 22–29, 2015.

[34] A. Maroosi and R. C. Muniyandi, "Accelerated execution of P systems with active membranes to solve the *N*-queens problem," *Theoretical Computer Science*, vol. 551, pp. 39–54, 2014.

[35] R. Maazallahi, A. Niknafs, and P. Arabkhedri, "A polynomial-time dna computing solution for the *n*-queens problem," *Procedia—Social and Behavioral Sciences*, vol. 83, pp. 622–628, 2013.

[36] A. Draa, S. Meshoul, H. Talbi, and M. Batouche, "A quantum-inspired differential evolution algorithm for solving the N-queens problem," *Neural Networks*, vol. 1, p. 12, 2011.

[37] B. Keswani, *Implementation of n-Queens Puzzle Using Meta-Heuristic Algorithm (Cuckoo Search) [Dissertation]*, Suresh Gyan Vihar University, 2013.

[38] A. A. Shaikh, A. Shah, K. Ali, and A. H. S. Bukhari, "Particle swarm optimization for N-queens problem," *Journal of Advanced Computer Science & Technology*, vol. 1, no. 2, pp. 57–63, 2012.

[39] J. E. A. Heris and M. A. Oskoei, "Modified genetic algorithm for solving n-queens problem," in *Proceedings of the Iranian Conference on Intelligent Systems (ICIS '14)*, Bam, Iran, February 2014.

[40] S. Khan, M. Bilal, M. Sharif, M. Sajid, and R. Baig, "Solution of n-Queen problem using ACO," in *Proceedings of the IEEE 13th International Multitopic Conference (INMIC '09)*, pp. 1–5, IEEE, Islamabad, Pakistan, December 2009.

[41] N. Vaughan, "Swapping algorithm and meta-heuristic solutions for combinatorial optimization n-queens problem," in *Proceedings of the Science and Information Conference (SAI '15)*, pp. 102–104, London, UK, July 2015.

[42] E. Masehian, H. Akbaripour, and N. Mohabbati-Kalejahi, "Landscape analysis and efficient metaheuristics for solving the n-queens problem," *Computational Optimization and Applications*, vol. 56, no. 3, pp. 735–764, 2013.

[43] A. P. Burger and C. M. Mynhardt, "An upper bound for the minimum number of queens covering the $n \times n$ chessboard," *Discrete Applied Mathematics*, vol. 121, no. 1–3, pp. 51–60, 2002.

[44] W. D. Weakley, "Upper bounds for domination numbers of the queen's graph," *Discrete Mathematics*, vol. 242, no. 1–3, pp. 229–243, 2002.

[45] P. Gibbons and J. Webb, "Some new results for the queens domination problem," *Australasian Journal of Combinatorics*, vol. 15, pp. 145–160, 1997.

[46] S. Bozóki, P. Gál, I. Marosi, and W. D. Weakley, "Domination of the rectangular queen's graph," https://arxiv.org/abs/1606.02060.

[47] H. Fernau, "Minimum dominating set of queens: a trivial programming exercise?" *Discrete Applied Mathematics*, vol. 158, no. 4, pp. 308–318, 2010.

[48] N. Mohabbati-Kalejahi, H. Akbaripour, and E. Masehian, "Basic and hybrid imperialist competitive algorithms for solving the non-attacking and non-dominating n-queens problems," in *Computational Intelligence: International Joint Conference, IJCCI 2012 Barcelona, Spain, October 5–7, 2012 Revised Selected Papers*, K. Madani, A. D. Correia, A. Rosa, and J. Filipe, Eds., pp. 79–96, Springer, 2015.