

Landscape analysis and efficient metaheuristics for solving the n -queens problem

Ellips Masehian · Hossein Akbaripour ·
Nasrin Mohabbati-Kalejahi

Received: 30 December 2012 / Published online: 3 July 2013
© Springer Science+Business Media New York 2013

Abstract The n -queens problem is a classical combinatorial optimization problem which has been proved to be NP-hard. The goal is to place n non-attacking queens on an $n \times n$ chessboard. In this paper, two single-solution-based (Local Search (LS) and Tuned Simulated Annealing (SA)) and two population-based metaheuristics (two versions of Scatter Search (SS)) are presented for solving the problem. Since parameters of heuristic and metaheuristic algorithms have great influence on their performance, a TOPSIS-Taguchi based parameter tuning method is proposed, which not only considers the number of fitness function evaluations, but also aims to minimize the runtime of the presented metaheuristics. The performance of the suggested approaches was investigated through computational analyses, which showed that the Local Search method outperformed the other two algorithms in terms of average runtimes and average number of fitness function evaluations. The LS was also compared to the Cooperative PSO (CPSO) and SA algorithms, which are currently the best algorithms in the literature for finding the first solution to the n -queens problem, and the results showed that the average fitness function evaluation of the LS is approximately 21 and 8 times less than that of SA and CPSO, respectively. Also, a fitness analysis of landscape for the n -queens problem was conducted which indicated that the distribution of local optima is uniformly random and scattered over the search space. The landscape is rugged and there is no significant correlation between fitness and distance of solutions, and so a local search heuristic can search a rugged plain landscape effectively and find a solution quickly. As a result, it was statistically and

E. Masehian (✉) · H. Akbaripour
Industrial Engineering Department, Tarbiat Modares University, Tehran, Iran
e-mail: masehian@modares.ac.ir

H. Akbaripour
e-mail: h.akbaripour@modares.ac.ir

N. Mohabbati-Kalejahi
Faculty of Industrial Engineering, Amirkabir University of Technology, Garmsar Campus, Iran
e-mail: mohabbati@aut.ac.ir

analytically proved that single-solution-based metaheuristics outperform population-based metaheuristics in finding the first solution of the n -queens problem.

Keywords n -Queens problem · Local search · Simulated annealing · Scatter search · Parameter tuning · TOPSIS method · Fitness analysis of landscape

1 Introduction

The n -queens problem is a classical combinatorial optimization problem in Artificial Intelligence [9]. A solution to the problem obtained by placing n non-attacking queens on an $n \times n$ chessboard considering the chess rules. Although the problem itself has an uncomplicated structure, it has been broadly utilized to develop new intelligent problem solving approaches. Despite the n -queens problem is often studied as a “mathematical recreation”, it has found several real-world applications such as practical task scheduling and assignment, computer resource management (deadlock prevention and register allocation), VLSI testing, traffic control, communication system design, robot placement for maximum sensor coverage, permutation problems, parallel memory storage schemes, complete mapping problems, constraint satisfaction, and other physics, computer science and industrial applications [10, 32, 34]. The variety of these applications indicates the reason of the wide interest on this well-known problem.

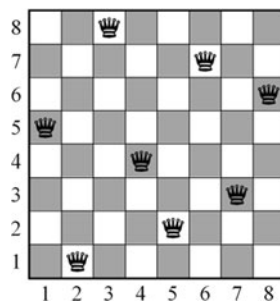
Probably the basic form of the n -queens problem was the 8-queens variant, which was originally proposed in 1848 by the chess player [5], published in the German chess newspaper *Berliner Schachzeitung*. It was republished in 1850 and attracted the attention of the famous mathematician Carl Friedrich Gauss for finding all possible solutions, though he found only 72 of the 92 possible answers. Nauck found all the 92 solutions in the same year [31].

The earliest paper on the general n -queens problem was presented by [24], and the first proof of the possibility of placing n non-attacking queens on an $n \times n$ chessboard is credited to [28]. A thorough review on the problem and its applications is presented by Bell and Stevens [4]. The n -queens problem belongs to the class of Constraint Satisfaction Problems (CSP), and is known as an NP-hard problem. There is a simple reduction to the Maximum Clique problem through a non-attacking queen graph, in which vertices represent chessboard squares and there is an edge between two vertices (squares) if they are neither in the same row, column or diagonal [32]. Also, reduction of the n -queens problem to the Maximum Clique problem and a proof that the n -queens problem is NP-hard are presented in [16]. Furthermore, the exact search enumeration of the first K solutions ($K > 1$) for the n -queens is shown to be NP-hard, in addition to interesting reductions to other NP-hard problems in [4].

There are three variants of the n -queens problem [2]: (1) finding all solutions of a given $n \times n$ chessboard, (2) generating one or more, but not all solutions, and (3) finding only one solution. In the first variant, finding all solutions may be possible for small sizes, but the number of solutions increases exponentially with the problem size, such that the largest instance solved to date is for $n = 26$ with a total number of 2.23×10^{16} solutions (see Table 1), calculated within 271 days on parallel supercomputers in 2009 [33]. A solution to the 8-queens problem (out of 92 solutions) is illustrated in Fig. 1.

Table 1 Size of solution space and the number of solutions for the n -queens problem solved to date

n	Size of solution space ($n!$)	Number of solutions
1	1	1
2	2	0
3	6	0
4	24	2
5	120	10
6	720	4
7	5040	40
8	40320	92
9	362880	352
10	3628800	724
11	39916800	2680
12	479001600	14200
13	6227020800	73712
14	87178291200	365596
15	1307674368000	2279184
16	20922789888000	14772512
17	355687428096000	95815104
18	6402373705728000	666090624
19	121645100408832000	4968057848
20	2432902008176640000	39029188884
21	51090942171709440000	314666222712
22	112400072777607680000	2691008701644
23	25852016738884976640000	24233937684440
24	620448401733239439360000	227514171973736
25	1551121004330985984000000	2207893435808352
26	403291461126605635584000000	22317699616364044

Fig. 1 A solution to the 8-queens problem, presented as [5, 1, 8, 4, 2, 7, 3, 6]

1.1 Related work

According to the extensive bibliography of the n -queens problem of Kosters [22], mathematical and statistical techniques, heuristic and metaheuristic algorithms, both exact and approximate, have been proposed for solving the problem [8, 26, 30]. The

main advantage of metaheuristics compared to exact methods is their ability in handling large-scale instances in a reasonable time [41], but at the expense of losing a guarantee for achieving the optimal solution. Therefore, due to the NP-hardness of the n -queens problem, metaheuristic techniques are appropriate choices for solving it.

In designing metaheuristics for solving optimization problems, the two important search strategies of exploitation (intensification) and exploration (diversification) create two different classes of algorithms [36]. Algorithms in the first class are able to intensify the search in local regions, and are called Single-solution-based metaheuristics (or S-metaheuristics) since a single solution is iteratively improved until a satisfactory solution is obtained. Population-based metaheuristics (P-metaheuristics) are the second class which introduce diversity in exploring the search space by simultaneously and iteratively improving multiple solutions and finally select the best solution among them. Simulated Annealing (SA) [37] and Tabu Search (TS) are examples of S-metaheuristics, and Genetic Algorithm (GA), Differential Evolution Algorithm (DEA) [9] and Ant Colony Optimization (ACO) are examples of P-metaheuristics.

These methods have been widely used for solving the n -queens problem: for instance, Homaifar et al. [14] determined how well the operators of Genetic Algorithms handled very difficult combinatorial and constraint satisfaction problems such as the n -queens problem. They presented computational results for $n < 200$. Also, three metaheuristic algorithms of SA, TS, and GA were used to solve the n -queens problem by Martinjak. They presented test results and upper bound complexity for the problem. Many problem instances with large dimensions were solved and efficiencies of the algorithms were compared. Dirakkhunakon and Suansook [7] compared the results of the classical SA algorithm with the Iterative Improvement Simulated Annealing (IISA) algorithm for the n -queens problem. Their numerical results showed that the modified scheme provides better results than the classical algorithm. Khan et al. [20] proposed a solution for the n -queens problem based on ACO, which was applied to the 8-queens problem and asserted that it is scalable for large n . Their work contains detailed discussions on the problem background and complexity, ACO and experimental graphs.

In this paper the performance of Local Search (LS) and Simulated Annealing S-metaheuristics are compared to that of the Scatter Search (SS) P-metaheuristic (in two versions) in finding the first solution to the n -queens problem. Due to the fact that heuristic methods are parameter sensitive for finding the best solution, a parameter tuning approach based on a combination of TOPSIS and full-factorial design (for SA), and TOPSIS and Taguchi design (for SS) methods is proposed for each algorithm in order to obtain the best set of parameters for reducing both the number of fitness function evaluations (FFE) and the runtimes for solving the n -queens problem. As a result, the LS method surpassed all other proposed methods in this paper, as well as the Cooperative PSO (CPSO) [3] and SA [26] algorithms, which are currently the best algorithms in the literature for finding the first solution to the n -queens problem.

Another contribution of the paper is the fitness analysis of landscape for the n -queens problem using several statistical measures, including the distribution of solutions in the search space, entropy in search space, and fitness distance correlation

and autocorrelation factors, which showed that the problem has a random uniform nature and the local optima are quite scattered over the search space. The landscape is rugged and there is no correlation between fitness and distance of solutions. That's why a local search heuristic can search a flat rugged landscape and find a solution quickly. On the other hand, experimental results also proved that single-solution-based metaheuristics outperform population-based metaheuristics in finding the first solution of the n -queens problem, and thus confirmed the analytical results obtained from the landscape analysis.

The paper is organized as follows: Sects. 2 and 3 respectively present the Local Search and Simulated Annealing S-metaheuristic methods and their experimental results of solving the n -queens problem. In Sect. 4 two versions of the Scatter Search P-metaheuristic and experimental results are described in detail. The parameters of the SA and SS methods are tuned to their best values through TOPSIS-based methods, which are presented in their respective sections. In Sect. 5 the presented methods are further compared with the best methods in the literature for various sizes of the n -queens problem. Section 6 deals with fitness analysis of landscape of the problem, which helps to investigate the nature of the problem and understand why implementing the local search leads to better efficiency and effectiveness. Finally, concluding remarks are presented in Sect. 7.

2 Local search

The Local Search (LS) method (also known as hill-climbing, steepest descent, or gradient descent search) is the oldest and simplest single-solution-based metaheuristics, a form of which was the Simplex method for linear programming developed by George Dantzig in 1947 [1, 36]). Initially, a solution is generated for the problem, and then among all (or some) of its neighbor solutions, the one with the best fitness function value is selected as the new solution. This process repeats iteratively until for the current solution, there is no neighbor with better fitness function value. Local Search is prone to be trapped in local minima, but is used extensively in conjunction with other metaheuristic methods. In this section the implementation of the LS method in the n -queens problem is described.

2.1 Initial solution

In the n -queens problem, if no constraints are considered in locating n queens on an $n \times n$ chessboard, then the number of all possible solutions will be $(n \times n)^n$. However, this can be reduced to n^n by putting only one queen in each row. In addition, by putting a queen in only one column, the search space will be equal to a permutation of n queens, which is $n!$. In this paper, for generating the initial permutation we consider the latter method of locating n queens, and as a result, there will be no attacks in rows and columns, but only in diagonals. The objective function value is calculated by counting the number of attacks, and the optimal solution is a permutation of n queens on the chessboard without any attacks between queens.

Each solution is encoded in the form of a permutation $[\pi(1), \pi(2), \dots, \pi(n)]$, in which the value of $\pi(i)$ indicates the row number and i specifies the column number

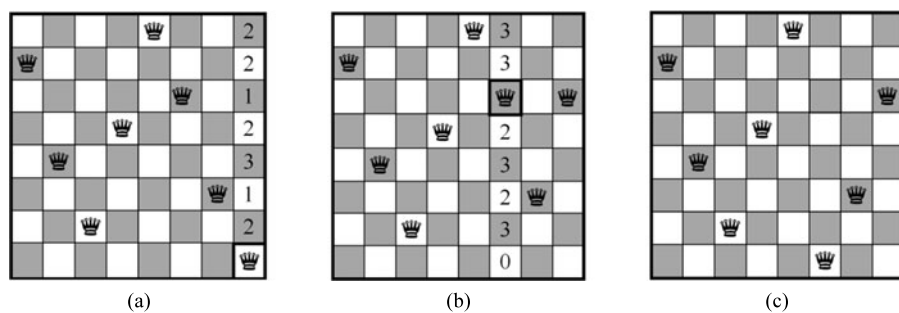


Fig. 2 The Minimum Conflicts method: (a) the 8th column is selected at random and the number of attacks are calculated for all squares in the column. Then the queen in the first row moves to the 6th row (note that it could also move to the third row). (b) The last move overrides the no-conflict rule in the 6th row, and so the queen in the 6th column must move another row, which in this case is the first row with zero conflicts

of a queen on the chessboard (see Fig. 1). Through this scheme, we can easily generate initial permutations with no two queens on the same row or column, letting the conflicts occur merely along the diagonals of the chessboard.

2.2 Neighborhood generation

In the literature, two appropriate neighborhood generation methods have been commonly used for the n -queens problem [26, 34]: Minimum Conflicts (MC) and Random Swap (RS). The Minimum Conflicts method creates a new arrangement of queens by relocating them in a randomly-selected column (or row) on the chessboard. In the selected row or column, all potential moves with the number of attacking queens in each square are calculated and the queen in that row or column moves to the square with minimum number of attacks (conflicts). Figure 2 demonstrated how the Minimum Conflicts operates in order to find a solution for 8-queens problem in two successive stages.

The Random Swap operator is another usual neighborhood generator which exchanges the locations of two random elements of a permutation. For a permutation of size n , the size of this neighborhood is $n(n-1)/2$. Figure 3 illustrates the implementation of the Random Swap to the permutation A, which leads to permutation B.

The Random Swap may or may not decrease the number of conflicts among queens. So, to make the neighborhood generation more goal-directed, we propose a new variant of the swap operator, called Effective Swap, which acts more intelligently than the random swap since it selects the exchange rows by also considering the number of attacks rather than just choosing them randomly. The following details illustrate the function of this new operator.

The Effective Swap operator starts with counting the number of conflicts on the main diagonal of the chessboard. If this number is nonzero, it marks that diagonal for further operations. Otherwise, it proceeds with the subdiagonals immediately above and below the main diagonal. Conflict counting is repeated for these diagonals too, and if no conflicts are found, it proceeds with farther subdiagonals parallel to the main diagonal. In case that still no conflicts are identified, the above procedure is repeated

Fig. 3 Applying the Random Swap operator to the columns 1 and 6

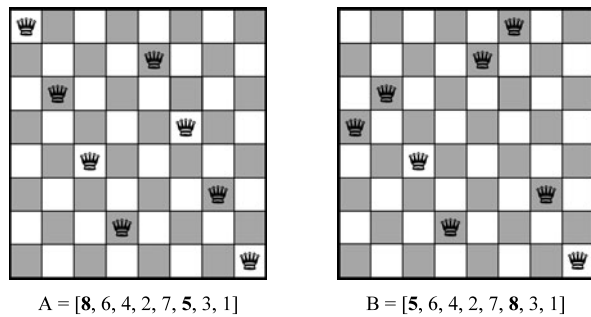
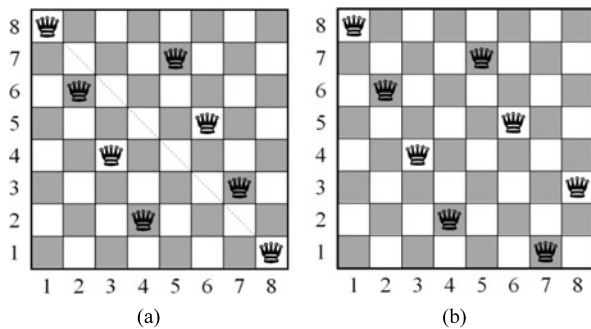


Fig. 4 (a) Before, and (b) after applying the Effective Swap on the chessboard



for the secondary diagonal and its parallel subdiagonals until a conflicting diagonal is found and marked for further operations.

Next, suppose that the marked diagonal has m conflicts. Then the operator performs $m - 1$ random swaps, such that in each swap, one of the queens is selected from the conflicting queens, and the other is a randomly-selected queen not causing any conflict in the marked diagonal. It is worthy to note that performing an Effective Swap does not guarantee an improvement in the fitness function; however, as indicated by our extensive experiments it reduces the number of conflicts far better than the random swap operator.

As an example of Effective Swap, consider a configuration of 8 queens displayed in Fig. 4(a), where there are $m = 2$ conflicting queens on the marked main diagonal, namely $\pi(1)$ and $\pi(8)$, of which one queen is selected randomly, e.g., $\pi(8)$. Now another queen which does not cause conflicts in this diagonal is randomly selected, e.g., $\pi(7)$, and the selected rows are swapped by $\pi(7) \leftrightarrow \pi(8)$, shown in Fig. 4(b).

After applying an Effective Swap, a neighbor permutation is generated, and we check whether any improvement has occurred in the fitness function or not. If so, then this neighbor permutation is kept; otherwise, a new permutation is generated.

The above procedure iterates until the stopping criterion is satisfied. In the n -queens problem this criterion is obtaining the first solution, which is a permutation of queens without any attack between them.

Table 2 Results of performing the Local Search for various sizes of the n -queens problem

n	FFE			$NCCA$	Runtime (secs.)	
	Min	Max	Avg.		Avg.	S.D.
8	18	133	44.5	0.97	0.06	0.01
10	25	361	127.7	1.65	0.08	0.03
25	65	221	160.6	0.76	0.10	0.02
50	170	860	430.6	0.78	0.24	0.13
100	373	970	763.6	0.77	0.55	0.13
200	1012	1574	1290.6	0.68	1.45	0.18
300	1583	2567	1980.3	0.67	3.91	0.52
400	2261	3426	2811.6	0.67	11.96	1.48
500	3320	3891	3553.7	0.68	25.00	1.91
750	4632	5830	5181.7	0.66	91.13	8.20
1000	4881	6961	6474.1	0.65	201.62	9.85
2000	9599	12671	10244.3	0.69	1689.70	10.22

2.3 Experimental results

Table 2 shows the experimental results of solving the n -queens problem at different sizes. Considering the randomness of the neighborhood generation procedure, performing the search on the same problem produced different solutions, and so each instance was run 10 times and the mean and the standard deviation (S.D.) of runtimes and two other performance criteria, the FFE and NCCA, are reported. The algorithm was coded in MatlabTM and run on an IntelTM Core i7 2.00 GHz CPU with 4.00 GB of RAM.

The FFE criterion measures the total number of Fitness Function Evaluations during the whole search, and NCCA stands for Normalized Convergence Curve Area. The convergence curve plots the best-found fitness function value at each iteration, until the final solution is reached. In the n -queens problem, this curve shows how the algorithm reduces the number of conflicts during its execution till it becomes zero.

Inspired by the behavior of the convergence curve, we designed a new performance criterion: the Normalized Convergence Curve Area, which is calculated according to (1), in which N_c^i is the number of conflicts during the i -th evaluation of the fitness function:

$$NCCA = \frac{CCA}{n^2} = \frac{1}{n^2} \sum_{i=1}^{FFE} N_c^i \quad (1)$$

In fact, by calculating the area under a convergence curve we can infer how fast a method reduces the number of conflicts. A relatively small area implies that the algorithm succeeded in reducing the number of conflicts during early iterations. The CCA measures the area under the convergence curve with the number of conflicts plotted along the vertical axis and the number of FFEs along the horizontal axis; but since for large problem sizes the area becomes too large, we divided it to a factor of n^2

and eliminated the impact of problem size, obtaining a normalized value. Examples of convergence curves are illustrated in Fig. 9.

3 Simulated annealing

Simulated Annealing (SA) is a generic probabilistic metaheuristic algorithm which can generate near-optimal solutions with polynomial cost for NP-hard problems. It is often used when the search space is discrete. SA is described as one of the effective S-metaheuristics, first introduced by Kirkpatrick et al. [21].

In SA, the search starts with an initial solution in the state space and iteratively approaches to the optimal solution by investigating the neighbors of each state and accepting a better solution. Up to this point the SA acts similar to the Local Search method; however, SA can accept worse solutions with a probability of

$$P(\Delta E, T) = e^{-\frac{f(s') - f(s)}{T}}, \quad (2)$$

in which T is control parameter that symbolizes the reducing temperature in the metals annealing process, and $f(s)$ and $f(s')$ are respectively the fitness function values of the current and the candidate neighbor solutions. This probability function, which is inspired from the Boltzmann's factor, is reduced as T is updated by a cooling schedule. The SA algorithm progresses until the optimal solution is obtained or a preset maximum number of iterations is reached. The pseudocode in Fig. 5 describes the template of the SA algorithm.

In SA with constant conditions such as initial structure, the initial temperature and temperature decreasing rate, reaching a global optimum is more probable in large neighborhoods than in small ones [40]. Therefore, it is obvious that SA is an appropriate algorithm for handling large-scale n -queens problems. The following subsections describe the main steps of implementing the SA method for the n -queens problem.

The SA algorithm starts with an initial permutation, which is generated by permuting n queens in a string, similar to the procedure used for the Local Search method, thus reducing the search space to $n!$.

In order to find good solutions in SA, the initial temperature is a sensitive parameter. The starting temperature should not be too high to conduct a random search for a period of time, but it should be large enough to make the uphill and downhill transition probabilities. In this paper, three linear relationships between the initial temperature (T_0) and the size of the problem (n) are assumed, which are $T_0 \in \{n, 5n, 10n\}$.

Another important step in SA which plays a critical role in the performance of the problem is the generation of proper neighborhoods for the current permutation, since inadequate neighborhood structure leads to failure in solving the problem. Thus we will consider the three neighborhood generation methods of Minimum Conflicts, Random Swap, and Effective Swap, among which one will be selected as the most appropriate one as a result of parameter tuning.

3.1 Cooling scheme

The cooling scheme has a vital role in the efficiency and the effectiveness of the SA algorithm, and defines the temperature at each step of the algorithm. Slow decrease of

```

 $T = T_{max}$ ; /* Initializing the temperature */
 $s = s_0$ ; /* initial solution */
Best solution =  $s$ 
repeat
  Generation of a random neighbor:  $s'$ 
  If  $\Delta E = f(s') - f(s) \leq 0$ 
    Then  $s = s'$ ; /* Accept the neighbor */
  Else Accept  $s'$  with a probability of  $e^{-(\Delta E/T)}$  and set  $s = s'$ 
  Best solution =  $s$ 
   $T = g(T)$ ; /* Temperature update according to the cooling schedule */
Until Stopping criteria satisfied
Return Best solution

```

Fig. 5 General template of the SA algorithm

the temperature leads to deep exploitation of each solution and hence better solutions, but at the cost of more computational time. The temperature T can be updated in different ways [27], including:

Linear In the linear cooling scheme the temperature T is updated as follows:

$$T = T - \beta, \quad (3)$$

where β is a specified constant value. Or it can be defined as:

$$T_i = T_0 - i \cdot \beta, \quad (4)$$

where T_i represents the temperature at iteration i .

Geometric The geometric cooling scheme is the most popular and effective cooling function and the temperature is updated using the following formula, in which $\alpha \in (0, 1)$:

$$T = \alpha \cdot T. \quad (5)$$

Logarithmic The logarithmic cooling scheme is considered as:

$$T_i = \frac{T_0}{\log(1 + i)}. \quad (6)$$

This schedule is too slow to be applied in practice, but has the proved property of converging to a global optimum [11].

We applied the above three cooling schemes for the proposed SA, and among them the geometric schedule produced better results in equal conditions. However in this scheme the cooling ratio α plays an important role in the success of the search: it must be on one hand large enough to speed up the search, and on the other hand small enough to prevent premature convergence to a low-quality local optimum solution. Therefore a careful parameter tuning was employed to set the best rate for α .

In the next subsection the parameter tuning procedure is explained as an effective approach for choosing the best sets of critical parameters in SA.

Table 3 Presumed levels for some important factors in the SA method

Parameter	Index of levels	Levels
Neighborhood type	1	Random swap
	2	Effective swap
	3	Minimum conflicts
Cooling rate (α)	1	0.90
	2	0.95
	3	0.99
Initial temperature (T_0)	1	n
	2	$5n$
	3	$10n$

3.2 Parameter tuning

As stated before, the parameters of metaheuristic algorithms have a significant effect on the efficiency and effectiveness of the search for a particular problem. There may be many options for these factors for a given problem. Therefore, using an appropriate approach like parameter tuning in order to find the best choice from many alternatives can produce better solution for the given problem. In this section, some levels of the parameters for SA algorithm are introduced. There are two goals in solving the n -queens problem by the proposed algorithms: reducing the numbers of fitness function evaluations, and reducing the runtime. The TOPSIS method is used to cope with both objectives at the same time. Finally, the results of the TOPSIS-based parameter tuning are displayed for the SA algorithm.

3.2.1 Parameter levels

Three factors are considered as the most important parameters in the SA: neighborhood type, cooling rate, and initial temperature, and therefore are chosen for being tuning carefully. Three alternatives are proposed for the neighborhood type, including Random swap, Effective swap, and Minimum conflicts operators. Also, three levels for the Cooling rate (α) and Initial temperature (T_0) are suggested for parameter tuning, as given in Table 3.

3.2.2 The TOPSIS method

Hwang and Yoon [15] developed the TOPSIS method to assess alternatives prior to multiple-attribute decision making. In the TOPSIS, the distance to the ideal solution and negative-ideal solution according to each alternative is considered, and then the best alternative is selected which is the nearest one to the ideal solution and the farthest one from the negative-ideal solution. The TOPSIS structure for aggregating the more important objectives in solving the n -queens problem can be explained as follows [38]:

1. Alternative performance matrix creation: The structure of the alternative performance matrix can be expressed as follows:

$$D = \begin{matrix} & \begin{matrix} FFE & Runtime \end{matrix} \\ \begin{matrix} R_1 \\ R_2 \\ \dots \\ R_m \end{matrix} & \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ \dots & \dots \\ x_{m1} & x_{m2} \end{bmatrix} \end{matrix} \quad (7)$$

In the proposed problem, the numbers of FFEs and Runtime are the objectives (X_j , $j = 1, 2$) which are related to alternative performances. Possible alternatives (run experiments) are denoted as R_i , $i = 1, \dots, m$; and x_{ij} is the performance of R_i with respect to the objective X_j .

2. Normalization of the performance matrix: For this purpose the following transformation formula is used, in which p_{ij} represents the normalized performance of R_i with respect to the objective X_j . The matrix form of p_{ij} is represented as P , with $i = 1, 2, \dots, m$ and $j = 1, 2$.

$$p_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^m x_{ij}^2}}, \quad P = [p_{ij}]. \quad (8)$$

3. Multiplying the performance matrix by its related weights: Each column of the matrix P is multiplied by weights associated with each objective FFE (w_{FFE}) and runtime (w_T). The weighted performance matrix V is obtained as follows:

$$V = \begin{bmatrix} w_{FFE} \cdot p_{11} & w_T \cdot p_{12} \\ w_{FFE} \cdot p_{21} & w_T \cdot p_{22} \\ \dots & \dots \\ w_{FFE} \cdot p_{m1} & w_T \cdot p_{m2} \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \\ \dots & \dots \\ v_{m1} & v_{m2} \end{bmatrix}, \quad (9)$$

in which v_{ij} represents the weighted normalized performance of R_i with respect to X_j for $i = 1, 2, \dots, m$ and $j = 1, 2$. Here we set $w_{FFE} = 0.6$ and $w_T = 0.4$.

4. Determination of ideal and negative-ideal solutions: The ideal value set V^+ and the negative-ideal value set V^- are determined as follows for minimizing both objectives simultaneously, in which $J' = \{j = 1, 2 | v_{ij}, \text{ a smaller response is desired}\}$:

$$V^+ = \{(\min v_{ij} | j \in J'), i = 1, 2, \dots, m\} = \{v_1^+, v_2^+\}, \quad (10)$$

$$V^- = \{(\max v_{ij} | j \in J'), i = 1, 2, \dots, m\} = \{v_1^-, v_2^-\}, \quad (11)$$

5. Calculation of separation measures: The separation (distance) of each alternative from the ideal solution (S_i^+), as well as the separation of each alternative from the negative-ideal solution (S_i^-) are given as follows:

$$S_i^+ = \sqrt{\sum_{j=1}^2 (v_{ij} - v_j^+)^2}, \quad S_i^- = \sqrt{\sum_{j=1}^2 (v_{ij} - v_j^-)^2}. \quad (12)$$

Table 4 Full factorial design of experiments for parameter tuning of SA

No.	Neighborhood type	Cooling rate	Initial temperature	C_i (TOPSIS index)
1	1	1	1	0.0530
2	1	1	2	0.3713
3	1	1	3	0.0527
4	1	2	1	0.2180
5	1	2	2	0.1154
6	1	2	3	0.3778
7	1	3	1	0.1936
8	1	3	2	0.3395
9	1	3	3	0.1396
10	2	1	1	0.9978
11	2	1	2	0.9858
12	2	1	3	0.9912
13	2	2	1	1.0000
14	2	2	2	0.9865
15	2	2	3	0.9896
16	2	3	1	0.9721
17	2	3	2	0.9756
18	2	3	3	0.9559
19	3	1	1	0.8438
20	3	1	2	0.9513
21	3	1	3	0.9083
22	3	2	1	0.9014
23	3	2	2	0.9243
24	3	2	3	0.8816
25	3	3	1	0.8862
26	3	3	2	0.8500
27	3	3	3	0.9435
\bar{C}	$\bar{C}_1 = 0.2068$ $\bar{C}_2 = 0.9838$ $\bar{C}_3 = 0.8989$	$\bar{C}_1 = 0.6839$ $\bar{C}_2 = 0.7105$ $\bar{C}_3 = 0.6951$	$\bar{C}_1 = 0.6740$ $\bar{C}_2 = 0.7222$ $\bar{C}_3 = 0.6934$	

6. Calculation of relative closeness to the ideal solution and ranking the preference order: The relative closeness C_i to the ideal solution can be expressed as follows:

$$C_i = \frac{S_i^-}{S_i^+ + S_i^-} \quad (13)$$

where C_i lies between 0 and 1. The closer C_i is to 1, the higher is the priority of the i -th run experiment. Because of the multiple levels of each parameter, \bar{C}_i is calculated as the mean of relative closeness to the ideal solution for each parameter per level. Numerical results of using the TOPSIS for the n -queens problem with tuned parameters are presented in the next section.

Table 5 Results of running SA for various sizes of the n -queens problem

n	FFE			NCCA	Runtime (s)	
	Min	Max	Avg.		Avg.	S.D.
8	41	221	85.0	1.83	0.07	0.02
10	1	889	165.0	2.62	0.09	0.07
25	220	819	378.1	1.81	0.18	0.06
50	266	695	492.4	1.29	0.27	0.05
100	573	1042	826.8	1.07	0.60	0.12
200	1057	2033	1517.6	0.90	1.88	0.31
300	1775	2803	2125.0	0.79	4.01	0.72
400	2347	3865	3109.1	0.80	13.75	1.93
500	2844	4121	3635.2	0.77	26.19	3.22
750	4745	6589	5406.4	0.72	96.39	9.56
1000	6164	8242	7028.1	0.71	217.66	18.65
2000	10213	15147	12975.6	0.69	1729.35	23.80

3.2.3 Parameter level selection

It is clear from the Table 3 that each SA parameter has 3 levels. Thus, $3 \times 3 \times 3 = 27$ experiments are required for the full factorial design. In this paper, we replicated each experiment five times and the relative closeness to the ideal solution (C_i) were calculated for them as reported in Table 4.

For choosing the best values of parameters, the average TOPSIS index (\bar{C}_i) for each parameter must be calculated and the level with the highest \bar{C}_i value (i.e., closest to 1) among them is selected as the best level for that parameter. Consequently, the best level for all the three parameters will be level 2, and the parameters are set as: Neighborhood generator = Effective Swap operator, Cooling Rate (α) = 0.95, and Initial temperature = 5 times the number of queens. It should be noted that tuning for n -queens problem was done for $n = 300$, but the results can be extended for other sizes as well.

3.3 Experimental results

The SA algorithm was run 10 times for each size of the problem and the average and standard deviation of runtimes, as well as the NCCA and FFE criteria were calculated, as presented in Table 5. For the FFE, the minimum, maximum, and average values are reported.

4 Scatter search

Scatter Search is an evolutionary and population-based metaheuristic, first introduced by F. Glover [13]. The algorithm starts by generating an initial population by considering two key factors: quality and diversity of solutions. Several strategies can be applied to get a population with these properties. In general, greedy procedures are used

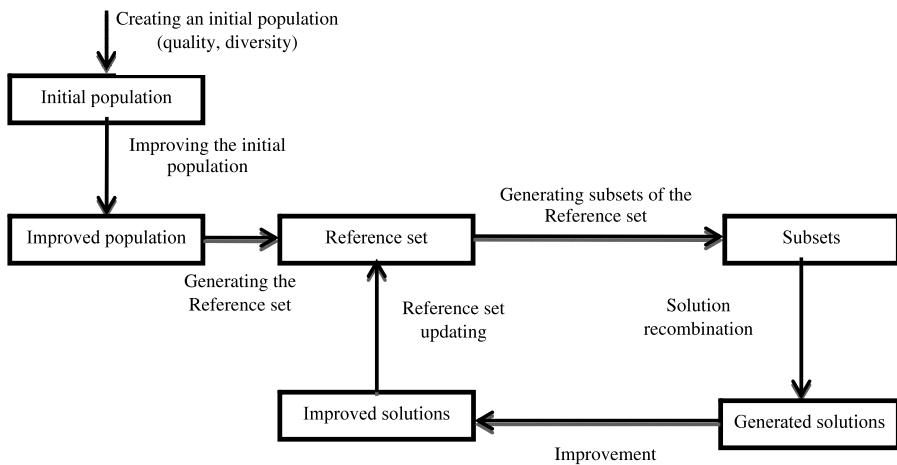


Fig. 6 Illustration of the search mechanism in the Scatter Search algorithm [36]

to diversify the search, and a simple heuristic procedure is applied to the solutions to refine them. In the next step, some of the solutions are selected as a Reference Set and combined to provide inputs to an improvement process of solutions that is based on a simple S-metaheuristic like local search. Then, the Reference Set is updated with the purpose of getting better solutions not similar to those already in the Reference Set. This procedure iterates until a stopping criterion is satisfied. Figure 6 illustrates the general procedure of the Scatter Search algorithm. The following subsections describe the details of our implementation of this algorithm to the n -queens problem.

4.1 Generation, diversification and improvement of initial solutions

The Scatter search starts with a set of initial solutions, which are required to be as scattered as possible within the solution space. Thus, the use of a systematic procedure to generate these solutions is essential. Regarding the n -queens as a permutation problem, the generator of combinatorial objects described in Glover [12] is considered, which operates as follows:

A permutation P is used as a starting point to generate subsequent permutations. Let's define the subsequence $P(h : s)$ where s is a positive integer between 1 and h , such that

$$P(h : s) = [s, s + h, s + 2h, \dots, s + rh] \quad (14)$$

where r is the largest nonnegative integer such that $s + rh \leq n$. Finally, the permutation $P(h)$ for $h < n$ is defined to be

$$P(h) = [P(h : h), P(h : h - 1), \dots, P(h : 1)] \quad (15)$$

For example, assume $h = 4$ and a seed permutation in the 8-queens problem is $P = [1, 2, 3, 4, 5, 6, 7, 8]$. The recursive application of $P(4 : s)$ for $s = 4, \dots, 1$ results in subsequences $P = [4, 8]$, $P = [3, 7]$, $P = [2, 6]$, and $P = [1, 5]$. Hence, $P(4) = [4, 8, 3, 7, 2, 6, 1, 5]$.

Half of the initial population is generated by the above method and the other half by random permutations. So, the initial population contains permutations with proper quality and diversity. In each random permutation there are no conflicts in rows or columns of the chessboard. This method not only reduces the search space considerably, but also generates initial permutations with acceptable quality. Afterwards, a heuristic algorithm taken from the work of Kale [19] is used to improve the quality of each permutation of the initial population as follows:

First the chessboard is divided into three hypothetical and (approximately) equal vertical groups of columns, each with almost $n/3$ columns. Then the initial permutation string S is processed from left to right: if an element has a value between $n/6 < \pi(i) < 5n/6$, it is sequentially placed in the first (left) and third (right) groups; otherwise, it is placed in the second (center) group. Note that in each group the original order of elements is preserved. As a result, in the new permutation S' the queens in lateral (resp. central) columns will be placed in the middle (resp. top and bottom) rows. For example, for a 12-queen initial permutation $S = [2, 6, 12, 8, 1, 5, 10, 7, 3, 9, 11, 4]$, employing this method will yield $S' = [6, 8, 5, 10|2, 12, 1, 11|7, 3, 9, 4]$. If the number of conflicts in S' is less than in S , it is saved as an improved solution; otherwise, solution S remains in the initial population.

4.2 Reference set selecting and updating

Diversity and quality of candidate solutions are two critical factors in the Reference Set. Therefore, it is suggested in the literature [23] to collect the elements of the Reference Set by selecting the best and diverse solutions to avoid similarities in solutions. In our proposed method the set of improved permutations obtained from the improvement method is sorted based on their objective function values. The Reference Set is constructed from two types of permutations: 50 % of permutations with better objective function values from the top of the list, and 50 % of the worse permutations from the bottom of the list. Its size is equal to B , which is calculated as in (16), in which $0 < \beta < 1$ and PS indicates the Population Size:

$$B = \beta \times PS. \quad (16)$$

Note that in the literature the Reference Set of the SS method should contain both high quality and diverse solutions. However, due to the fact that in the n -queens problem a multitude of global solutions exist, the diversity of a permutation cannot be measured relative to a single optimal solution. Therefore, we selected the worst permutations of the initial population to represent the most diverse permutations. As a result, by defining the Reference Set in such a manner a handful of high quality and diverse permutations are collected for being processed further in the next step.

4.3 Subset generation

In this step all subsets of size r (here we set $r = 2$) are generated in order to create various possible mixtures of quality (Q) and diversity (D). Regarding that half

of the Reference Set contains high-quality, and the other half highly-diverse solutions, $C(n/2, 2)$ (C means combination) of the generated subsets will contain Q-Q solutions, $C(n/2, 2)$ will contain D-D solutions, and $C(n, 2) - 2 \times C(n/2, 2)$ will contain Q-D pairs of solutions [29]. In this way a proper intensification and diversification of the problem space is attained, which will be further manipulated in the next step for creating even better solutions.

4.4 Solutions recombination

After producing a subset of permutations by the subset generation method, they are recombined to create new permutations using different combination methods, as discussed by Marti et al. [25]. In this paper the following two procedures are considered as combination operators:

Partially Matched Crossover (PMX): In this operator, in general, two genotypes (solution encodings, in the form of permutation strings of length n) are selected as parents A and B, and two crossover positions are picked randomly along the solutions. Then, all the chromosomes (elements) of Parent A lying between these two points are exchanged with the chromosomes of Parent B at the same positions, and vice versa.

For example, for the 8-queens strings in Fig. 7, taking the Parents A and B, the two crossover limits are fixed at 4th and 6th positions, and the dark area indicates the pairs which must undergo exchange. As a result, in both parents the following swaps take place: $7 \leftrightarrow 4$, $3 \leftrightarrow 1$, and $8 \leftrightarrow 2$, which create two new children.

Ordered Crossover (OX): In this method only one offspring is generated from two parents. First a substring from the Parent A (shown in dark in Fig. 8) is selected randomly and an offspring is produced by copying the substring into its corresponding position. Then these selected elements are deleted from the Parent B. The resulting sequence contains the elements that the offspring needs. The crossover is finished by placing the remaining elements into the vacant positions of the offspring from left to right, in order of their appearance in the Parent B. The procedure is demonstrated in Fig. 8.

Regardless of the type of crossover applied, the next generation will be selected from the best permutations of the pool (i.e. collection of parents and children), with the size of the population maintained.

Fig. 7 An example of parents and children in the Partially Matched Crossover (PMX)

Parent A:	2	4	6	7	3	8	5	1
Parent B:	8	5	3	4	1	2	7	6
Child 1:	8	7	6	4	1	2	5	3
Child 2:	2	5	1	7	3	8	4	6

Fig. 8 Illustration of the Order Crossover (OX) operator

Parent A:	8	7	2	5	1	4	6	3
Parent B:	2	5	8	7	4	1	3	6
Child:	2	8	7	5	1	4	3	6

4.5 Improvement method

In order to improve a trial candidate permutation in the elite population for generating a local optimum, a local search algorithm is suggested to be applied. For this purpose, a neighborhood of each permutation is produced by applying the Effective Swap operator. If any improvement is occurred in the fitness function, this neighbor is kept; otherwise, another neighbor is generated. This procedure iterates until a stopping criterion is satisfied. The stopping criterion contains a parameter T to control the depth of the local search, set by:

$$T = k \cdot n \quad (17)$$

where k is a constant and n is the size of the problem. After each iteration of the local search, the value of T is updated by (18), in which after extensive trial and error experiments, ε was set to 0.01.

$$T = (1 - \varepsilon) \cdot T \quad (18)$$

The local search procedure iterates until T reaches a lower bound like T_{min} . The algorithm with the mentioned features is named SS-1 in this paper. On the other hand, the n -queens problem has multiple optimal solutions (with a fitness function value of zero, meaning no conflicts), and the number of these solutions increases exponentially as n grows. Therefore, if the local search is given more time to transform an initial permutation, it can converge to an optimal solution much faster. For this purpose, whenever the newly generated neighbor causes an improvement in the fitness function value, a rewarding mechanism is enforced to update the T , as in (19). It can be seen that:

$$T = (1 + \varepsilon) \cdot T \quad (19)$$

In this rewarding mechanism, the temperature increment was set equal to its decrement in the cooling procedure, resulting in a factor of 1.01. This factor delays the convergence and causes the search to deeply exploit seemingly good permutations. As a result, such a dynamic definition of T causes an effective search of the space, as the algorithm spends more time on exploiting an appropriate permutation, and less time on non-promising ones. We call this algorithm SS-2 in this paper. The remaining steps are the same for both the SS-1 and SS-2 algorithms. The performance of two algorithms is compared in the Sect. 4.7.

After the above improvement, the Reference Set is updated with the upgraded permutations according to the Sect. 4.2, and the Updating-Recombination-Improvement procedure (the cycle in Fig. 6) is repeated until the stopping criterion is satisfied, which is when the first solution (i.e., a permutation of n queens on the chessboard with zero attacks) is found.

4.6 Parameter tuning

Like any other metaheuristic algorithm, setting the best combination of parameters in the SS will yield higher efficiency and effectiveness in solving various problems. Similar to the parameter tuning procedure described in Sect. 3.2 for the SA algorithm,

Table 6 Parameter levels in the Scatter Search algorithm

Parameter	Index of levels	Levels
PS (Population size)	1	50
	2	100
	3	150
	4	200
k (Constant factor)	1	1
	2	5
	3	10
	4	20
β (Constant coefficient)	1	0.05
	2	0.10
	3	0.15
	4	0.20
Recombination type	1	PMX
	2	OX

here the levels of important factors in SS will be introduced. But then a TOPSIS-Taguchi-based parameter tuning method is applied to reach the best set of parameters by considering the two objectives of FFEs and runtimes.

The influential parameters in our SS algorithm for the n -queens problem include the population size PS , the constant factor β in (16), the constant coefficient k in (17), and the recombination type, as indicated in Table 6, according to which the PS , k and β have four levels each, and the recombination type has two levels. It is noted that all these parameters are equally used in both SS-1 and SS-2 algorithms. Although the full factorial design requires $4 \times 4 \times 4 \times 2 = 128$ experiments, the Taguchi method can be used to reduce the number of experiments. Genichi Taguchi (1924–2012) was a Japanese engineer and statistician who developed a method for designing experiments to investigate the effect of process parameters on mean and variance of the process characteristics which define the performance of the process [35].

The experimental design proposed by Taguchi involves using orthogonal arrays to organize the parameters affecting the process, as well as the levels at which they should vary. Instead of having to test all possible combinations like the full factorial design, the Taguchi method tests only limited combinations of parameter levels. This allows for collection of necessary data to determine which factors affect the process most with minimal experiments, thus saving time and resources. The Taguchi method works best when there are intermediate numbers of variables, few interactions exist between variables, and only a few variables contribute significantly to the outcome.

Using the Taguchi orthogonal array selector, $L_{16} (4^3 \times 2^1)$ was chosen as the appropriate array for parameter tuning of the SS. We replicated each experiment five times and calculated the average of responses for TOPSIS index computation. The result of parameter tuning is reported in Table 7, which shows that the best levels for the PS , k , β , and Recombination type are 1, 3, 2 and 1, respectively, meaning that $PS = 50$, $k = 10$, $\beta = 0.1$, and Recombination type = PMX.

Table 7 The L_{16} Taguchi design of experiments for parameter tuning of SS

No.	PS (Population size)	k (Constant factor)	β (Constant coefficient)	Recombination type	C_i (TOPSIS index)
1	1	1	1	1	0.7268
2	1	2	2	1	0.8644
3	1	3	3	2	0.8352
4	1	4	4	2	0.6810
5	2	1	2	2	0.7116
6	2	2	1	2	0.6350
7	2	3	4	1	0.7570
8	2	4	3	1	0.9200
9	3	1	3	1	0.6627
10	3	2	4	1	0.6384
11	3	3	1	2	0.8384
12	3	4	2	2	0.8139
13	4	1	4	2	0.0244
14	4	2	3	2	0.5043
15	4	3	2	1	0.8268
16	4	4	1	1	0.7644
\bar{C}	$\bar{C}_1 = 0.7769$ $\bar{C}_2 = 0.7559$ $\bar{C}_3 = 0.7384$ $\bar{C}_4 = 0.5300$	$\bar{C}_1 = 0.5314$ $\bar{C}_2 = 0.6605$ $\bar{C}_3 = 0.8144$ $\bar{C}_4 = 0.7948$	$\bar{C}_1 = 0.7412$ $\bar{C}_2 = 0.8042$ $\bar{C}_3 = 0.7306$ $\bar{C}_4 = 0.5252$	$\bar{C}_1 = 0.7701$ $\bar{C}_2 = 0.6305$	

4.7 Experimental results

The results of running the two algorithms SS-1 and SS-2 are respectively reported in Tables 8 and 9 for various sizes of the n -queens problem, after parameter tuning.

As noted before, the two algorithms are different in their population improvement methods. In the SS-1, the number of FFEs, NCCA and average runtimes increase exponentially by the growth of the problem size, and consequently the results were possible to be reported only up to $n = 300$. The Tables also reveal that the SS-2 converges more quickly to the optimum solution than the SS-1, and shows how a small innovative improvement in the Local Search can result in significantly better solutions. It was interesting to observe that the NCCA for SS-2 remained almost constant for large problem sizes.

5 Comparisons

In the preceding section three algorithms were introduced for solving the n -queens problem and their numerical results were presented for various sizes of the problem. Table 10 summarizes and compares the performance measures of the proposed algorithms, including average number of FFEs, NCCA, and the mean and standard

Table 8 Results of running SS-1 for various sizes of the n -queens problems

n	<i>FFE</i>			<i>NCCA</i>	Runtime (s)	
	Min	Max	Avg.		Avg.	S.D.
8	1	129	59.0	1.11	0.12	0.01
10	8	303	209.4	4.07	0.16	0.06
25	51	351	207.9	0.80	0.20	0.04
50	198	653	383.1	0.72	0.32	0.07
100	448	2300	888.5	1.69	0.78	0.37
200	837	41716	27752.5	52.87	28.95	16.74
300	43860	50642	45407.0	42.75	84.32	13.76

Table 9 Results of running SS-2 for various sizes of the n -queens problems

n	<i>FFE</i>			<i>NCCA</i>	Runtime (s)	
	Min	Max	Avg.		Avg.	S.D.
8	29	493	132.0	1.50	0.11	0.05
10	101	648	218.5	1.99	0.16	0.06
25	121	320	176.1	0.77	0.19	0.04
50	101	725	340.2	0.80	0.36	0.07
100	101	906	484.9	0.60	0.64	0.10
200	944	1618	1262.7	0.62	1.75	0.27
300	1730	2537	2259.5	0.71	4.58	0.50
400	2136	3494	2713.3	0.67	12.13	1.72
500	2880	4231	3559.1	0.66	25.65	2.85
750	4472	6225	5117.3	0.67	92.52	9.77
1000	6118	7418	6839.1	0.66	211.93	17.51
2000	11125	15284	12981.4	0.66	1787.59	21.10

Table 10 Comparing the performance of the three proposed algorithms

Criterion	LS	SA	SS-2
Average number of FFEs	2755.28	3145.36	3007.00
Average <i>NCCA</i>	0.80	1.77	0.86
Average runtime	168.82	174.20	178.13
S.D. of runtime	2.72	4.88	4.50

deviation of runtimes (note that SS-1 is not included in the Table since it failed to produce solutions for large sizes in due time).

The results show that although SS-2 and SA have acceptable performance, the Local Search method outperforms the other two algorithms in all measures. Indeed, single solution-based metaheuristics deal with the n -queens problem better than the population-based metaheuristics in finding the first solution.

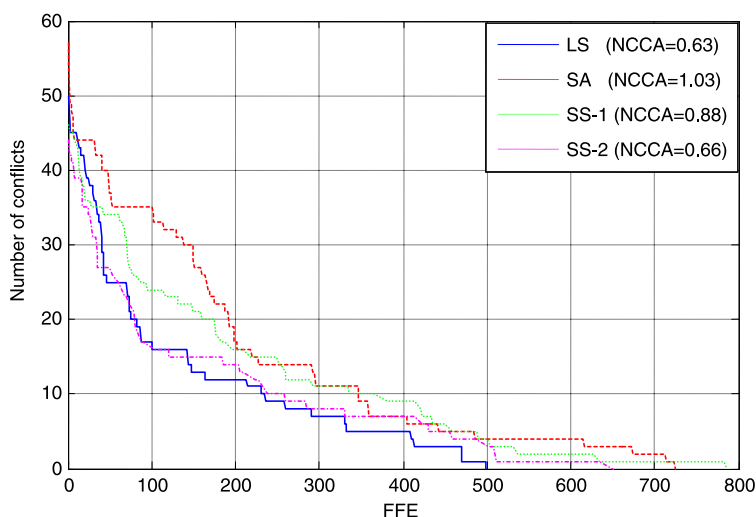


Fig. 9 Comparing the average NCCA of the proposed four algorithms for $n = 100$

Figure 9 illustrates the average Normalized Convergence Curve Area (NCCA) of the four proposed algorithms (LS, SA, SS-1 and SS-2) when solving the 100-queens problem. The NCCA measure demonstrates how an algorithm converges to the optimal solution. In this figure, the number of FFEs and the number of conflicts are shown along the horizontal and vertical axes, respectively. It is observed that there is no significant difference between the convergence speeds of the LS and SS-2, as their NCCAs are quite close.

In order to further support our finding about the strength of S-metaheuristics in solving the n -queens problem, additional experimentation is required. On the other hand, Martinjak and Golub [26] represented three metaheuristic algorithms (SA, TS, and GA) for solving the n -queens problem, and reported minimum, maximum and average number of iterations for each algorithm, but not the runtimes. Their results showed that SA outperforms the TS and GA methods. Also, Amooshahi et al. [3] presented a new Cooperative PSO (CPSO) method to solve permutation problems, including the n -queens. First an initial random population of particles is generated, each particle containing information about the locations of n queens on an $n \times n$ chessboard. Then each particle is divided into n equal sub-swarms, and each sub-swarm is changed into one sub-particle. Sub-particles use the standard PSO method to update their velocities and positions according to the best local experience of each sub-particle and the best position among all particles. Through a number of experiments, they compared the CPSO with implementations of standard PSO, SA, TS and GA algorithms (reported in Martinjak and Golub [26]) and showed that the CPSO outperformed all those metaheuristics in terms of the number of Fitness Function Evaluations (FFE).

As a result, the average number of FFEs obtained by the LS—as our superior method—is compared with the best algorithms in the literature, namely, the SA presented in [26] and the CPSO presented in [3]. The results are displayed in Table 11,

Table 11 Comparing average numbers of FFEs for LS, SA and CPSO methods

n	LS (proposed)	SA [26]	CPSO [3]
8	44.5	492.8	225.8
10	127.7	947.8	540.4
50	430.6	2848.6	2764.2
100	763.6	7872.7	5063.5
200	1290.6	21708.2	9184.5
300	1980.3	24636.2	14559.6
500	3553.7	56629.7	23799.6
750	5181.7	88953.0	34765.2
1000	6474.1	126401.7	47299.8
2000	10244.3	314373.0	95235.9

which shows that the LS method (with the Effective Swap operator) dominates the other two algorithms, with its average number of FFEs being approximately 21 and 8 times less than the SA and CPSO algorithms, respectively. As a result, LS outperforms the best algorithms in the literature.

For statistically demonstrating that S-metaheuristics have better performance than P-metaheuristics for the n -queens problem, a landscape fitness analysis is conducted.

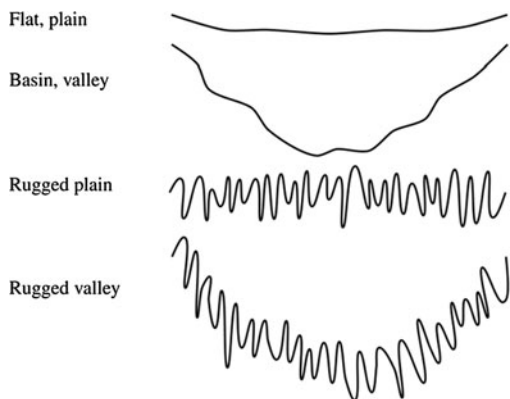
6 Analysis of landscape

No one can claim that a specific algorithm is the best one for solving all types of optimization problems. In the other words, if the performance of algorithm A is better than algorithm B for a specific problem, there may be another problem in which B acts better than A [39]. Therefore, in order to find out what type of metaheuristics is suitable for a specific problem, the landscape of the problem's search space should be analyzed. In fact, in designing a metaheuristic for an optimization problem the properties of the problem's landscape influence the effectiveness of the metaheuristic for a given instance. These properties are types of solution representation, neighborhood, and the objective function, which completely define the landscape of a problem.

If the search space is considered as a floor, then the landscape of the problems can be made of valleys, plains, peaks, canyons, cliffs, plateaus, basins, etc., as shown in Fig. 10. Depending on the shape of the landscape, specific types of search methods will be more effective.

By computing and analyzing indices such as ruggedness and distribution of the fitness function which explain the distribution of local optima (solutions with zero conflict) over the landscape of the n -queens problem, we will be able to form an idea about the shape of the landscape and thus can deduce what type of metaheuristic works well in that context. In this paper, landscape studies have been based on the 100-queens problem (as an average size), where the Local Search algorithm presented in Sect. 2 was run for 1000 uniformly-sampled random starting permutations (the set

Fig. 10 Presentation of various landscapes using geographical metaphors [36]



of starting permutations is called population U). For each starting permutation in U , the Local Search is run until a local optimum is achieved. As a result, 1000 local optimums were generated, the set of which is called population O . Regarding the exact numbers of local optimum solutions found for $n \leq 26$ (Table 1), we can be sure that there exist at least 1000 local optimal solutions for the 100-queens problem.

Distribution and correlation measures are the statistical measures that have been employed in the analysis of the landscape. Distribution measures are used to study the topology of local optima solutions over the landscape, and include (1) distribution in the search space, and (2) entropy in the search space. Correlation measures analyze the ruggedness of the landscape and explain the correlation between the quality of solutions and their relative distance, and include (3) Autocorrelation function, and (4) Fitness distance correlation. These measures are described in the following.

6.1 Distribution in the search space

For a general population P of solutions, the average distance $dmm(P)$ and the normalized average distance $Dmm(P)$ are calculated as follow:

$$dmm(P) = \frac{\sum_{s_1 \in P} \sum_{s_2 \in P, s_1 \neq s_2} dist(s_1, s_2)}{|P| \cdot (|P| - 1)}, \quad (20)$$

$$Dmm(P) = \frac{dmm(P)}{diam(S)}. \quad (21)$$

$Dmm(P)$ indicates the concentration of the population P in the search space S and the cardinality of P is shown as $|P|$. Particularly, a weak distance specifies that the solutions in P are clustered in a small region of the search space. The diameter of a general population P is the maximal distance between elements of P :

$$diam(P) = \max_{s_1, s_2 \in P} dist(s_1, s_2) \quad (22)$$

The distance $dist(s_1, s_2)$ is calculated as the sum of all absolute differences between the positions of all items in solution strings s_1 and s_2 , which is proposed by

Table 12 The values of distribution measures for the n -queens problem

n	P	O	$dmm(U)$	$Dmm(U)$	$dmm(O)$	$Dmm(O)$	Δ_{Dmm}
100	1000	1000	3333.4	0.7731	3320.9	0.7821	-0.011

Table 13 The values of entropy measures for the n -queens problem

n	P	O	$ent(U)$	$ent(O)$	Δ_{ent}
100	1000	1000	0.9890	0.9865	0.0026

Campos et al. [6]. This measure is dependent on the labeling of the items.

$$dist(s_1, s_2) = \sum_{i=1}^n |s_1(i) - s_2(i)|. \quad (23)$$

The Δ_{Dmm} indicator represents the variation of the average distance between a uniform population U (i.e. permutations uniformly sampled in the search space) and the population of local optimal solutions O :

$$\Delta_{Dmm} = \frac{(Dmm(U) - Dmm(O))}{Dmm(U)}. \quad (24)$$

By applying the above equations and measures for 1000 local optima of the 100-queens problem, the following results are generated and reported in Table 12.

The relatively large values of $Dmm(U)$ and $Dmm(O)$ indicate that the local optimum solutions in the population O are distributed in different directions over the search space.

6.2 Entropy in the search space




The concept of entropy refers to the diversity of a specified population in the search space. If the entropy is weak for a problem space, the solutions are concentrated. On the other hand, high entropy indicates distribution of solutions over the search space. For the n -queens problem the entropy of a general population P is measured as follows:

$$ent(P) = \frac{-1}{n \log n} \cdot \sum_{i=1}^n \sum_{j=1}^n \left(\frac{n_{ij}}{|P|} \log \left(\frac{n_{ij}}{|P|} \right) \right), \quad 0 \leq ent(P) \leq 1 \quad (25)$$

in which n_{ij} is the number of times a queen i is assigned to location j (of a permutation) in the population P , and $|P| = 1000$ is the cardinality (size) of P . After applying a local search algorithm on each permutation of the population, the entropy variation Δ_{ent} between the uniform population U and the local optima population O is presented by (26). The numerical results of Entropy are displayed in Table 13.

$$\Delta_{ent} = \frac{(ent(U) - ent(O))}{ent(U)}. \quad (26)$$

Table 14 Fitness landscape structures according to the variation of the distance and the entropy [36]

Entropy Δ_{Ent}	Low variation	High variation	High variation
Distribution Δ_{Dmm}	Low variation	Low variation	High variation
Landscape			
	Uniform	Multimassif	One-Massif

The entropy gives information about the dispersion of optimal solutions but does not tell about the size of the concentrations, as scattered concentrations or a single concentration may have the same low entropy. On the other hand, distribution measures give complementary information about the concentration of optimal solutions in a single region. Therefore, by considering both the variations of distribution and entropy between the initial population U and the final optimal population O (i.e., Δ_{Dmm} and Δ_{ent}) it is possible to form an idea about the distribution of the local optima in the search space. Indeed, as Table 14 illustrates, the approximate distribution of local optima can be categorized into three main groups: Uniform, Multimassif, and One-massif. For our problem, considering the small values of Δ_{Dmm} and Δ_{ent} it can be inferred that the optimal solutions are uniformly distributed in the search space.

6.3 Autocorrelation function

The Autocorrelation function $\rho(d)$ is a correlation indicator used to measure the ruggedness of a landscape, in which d denotes distance in measuring the correlations of solutions in the search space. When $d = 1$ the correlations are measured between neighbor solutions.

$$\rho(d) = \frac{\sum_{s,t \in S \times S, \text{dist}(s,t)=d} (f(s) - \bar{f}) \cdot (f(t) - \bar{f})}{n \cdot \sigma_f^2} \quad (27)$$

If $\rho(d) \approx 0$ then the landscape of the problem is rugged since the variation of fitness between two neighbors is equal on average to the variation between any two solutions. On the other hand, if $\rho(d) \approx 1$ then the landscape is smoother since two neighbors have a similar fitness value but are distant.

In n -queens problem the value of the Autocorrelation function is $\rho(d) \approx 0.07$, and thus the landscape of the problem is rugged.

6.4 Fitness–distance correlation

This indicator shows how much the fitness of a permutation has a relation with its distance to the global optimum. According to the nature of the n -queens problem, all local optima are global optima since all have zero conflicts among queens.

In global optimum the relationship between the fitness and the distance affects the search complexity [17, 18]. For fitness distance analysis, $F = \{f_1, f_2, \dots, f_n\}$, which

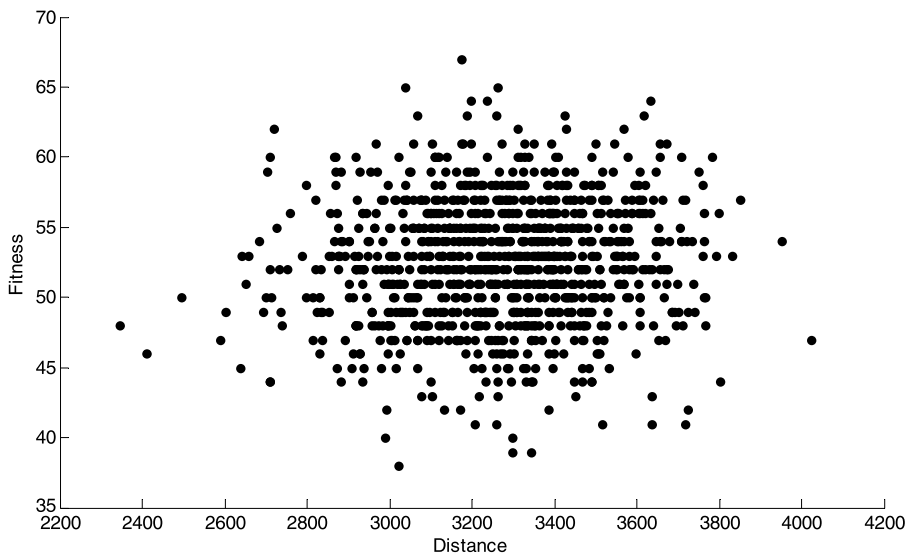


Fig. 11 The fitness-distance plot of the 100-queens problem, using the effective swap neighborhood structure

is the fitness values (i.e., number of conflicts) of the permutations in the uniform population U , and $D = \{d_1, d_2, \dots, d_n\}$, which is the distances of the permutations in U to their corresponding local optimums, should be calculated in order to find the correlation coefficient, r :

$$r = \frac{\text{cov}(F, D)}{\sigma_f \sigma_d}, \quad (28)$$

$$\text{cov}(F, D) = \frac{1}{n} \sum_{i=1}^n (f_i - \bar{f})(d_i - \bar{d}). \quad (29)$$

in which the mean and standard deviation of fitness values are denoted by \bar{f} and σ_f , and the mean and standard deviation of distances to the global optimum are denoted by \bar{d} and σ_d , respectively. The correlation coefficient may be a large positive, a large negative or near zero value, in which cases the problem is called straightforward, misleading, and difficult, respectively. Since for our analysis we calculated the correlation coefficient $r = 0.083$, the problem is a difficult one.

For further analysis of the Fitness-Distance Correlation (FDC), we plotted the fitness-distance diagram of the problem (Fig. 11) which illustrates the fitness of permutations in U (i.e., the number of conflicts) against their distance to their achieved optimal solutions in O .

The analyses described in this section reveals that the n -queens problem has a random uniform nature, its local optima are very scattered over the search space, the range of their quality is small, and their average costs are rather good. The correlation measures show that the landscape of the problem is rugged, and there is no correlation between the fitness and distance, and problem is difficult to solve. In such

conditions, the Local Search method (or other S-metaheuristics like SA) encounters a rugged plain landscape (see Fig. 10) and can find a solution quickly by starting from almost anywhere and converge to a nearby local optimum. This fact validates our experiments which showed the superiority of the LS method over other methods. The LS performed even better than the SA, which loses time in lowering the temperature. Of course, the Effective Swap neighbor-generating operator has also its share in the success of the LS method.

On the other hand, although population-based metaheuristics concurrently search the landscape from numerous locations, their convergence toward the first solution with zero conflict is slow because they employ time-consuming new solution generation operators which actually cause no added value in terms of efficiency. It is noted that in the Table 10 the relatively small average NCCA of the Scatter Search method ($= 0.86$) over the SA method ($= 1.77$) is largely due to the local search component used for improving the solutions of the Reference Set (Sect. 4.5).

The above explanations are true when the goal is to find the first solution. However, the landscape analysis shows that for finding more than one solutions of the n -queens problem, P-metaheuristics are more appropriate than S-metaheuristics since they concurrently evolve a whole population of chromosomes toward multiple solutions.

7 Conclusions

The n -queens problem is a well-known combinatorial optimization problem with many real-world applications. There are three approaches in finding solutions for the problem: finding all possible solutions (completely known for $n \leq 26$); finding one or more, but not all, solutions; and finding only one solution. Regarding the nature of metaheuristic algorithms which do not guarantee to find all global solutions, they can be used to find just one or some of the solutions. In this paper, like other works dealing with the problem, metaheuristic algorithms were used to find the first solution, and the developed Local Search (LS), Simulated Annealing (SA), and Scatter Search (SS) metaheuristics are used for finding the first solution of the n -queens problem.

Due to significant effects of parameters on the effectiveness of metaheuristics, TOPSIS-full-factorial and TOPSIS-Taguchi based parameter tuning methods are proposed to select the best set of parameters for each algorithm. Experimental result showed that the average number of Fitness Function Evaluations of the LS was approximately 21 and 8 times less than the best algorithms in the literature, i.e., SA [26] and CPSO [3], respectively. Based on a fitness analysis of landscape, the n -queens problem proves to have a random uniform nature, where local optima are distributed over the search space. The landscape has a rugged plain form and there is no correlation between the fitness and distance of solutions. Therefore, a local search heuristic can quickly find a solution by starting from a random point and converging to one of the many local optimal solutions nearby it. The fact that S-metaheuristics perform better than P-metaheuristics in finding the first solution of the n -queens problem has been shown experimentally.

References

1. Aarts, E.H.L., Lenstra, J.K.: *Local Search in Combinatorial Optimization*. Wiley, New York (1997)
2. Abramson, B., Yung, M.: Divide and conquer under global constraints: a solution to the N -queens problem. *J. Parallel Distrib. Comput.* **6**(3), 649–662 (1989)
3. Amooshahi, A., Joudaki, M., Imani, M., Mazhari, N.: Presenting a new method based on cooperative PSO to solve permutation problems: a case study of n -queen problem. In: 3rd International Conference on Electronics Computer Technology (2011)
4. Bell, J., Stevens, B.: A survey of known results and research areas for n -queens. *Discrete Math.* **309**, 1–31 (2009)
5. Bezzel, M.: Proposal of 8-queens problem, *Berliner Schachzeitung* 3, 363 (1848)
6. Campos, V., Laguna, M., Mart, R.: Context-independent scatter search and tabu search for permutation problems. *INFORMS J. Comput.* **17**, 111–122 (2005)
7. Dirakkhunakon, S., Suansook, Y.: Simulated annealing with iterative improvement. In: International Conference on Signal Processing Systems (2009). doi:[10.1109/ICSPS.2009.61](https://doi.org/10.1109/ICSPS.2009.61)
8. Draa, A., Talbi, H., Batouche, M.: A quantum inspired genetic algorithm for solving the N -queens problem. In: Proceedings of the 7th International Symposium on Programming and Systems, Algiers, pp. 145–152 (2005)
9. Draa, A., Meshoul, S., Talbi, H., Batouche, M.: A quantum-inspired differential evolution algorithm for solving the N -queens problem. *Int. Arab J. Inf. Technol.* **7**, 21–27 (2010)
10. Erbas, C., Sarkeshik, S., Tanik, M.M.: Different perspectives of the n -queens problem. In: Proceedings of the 1992 ACM Annual Conference on Communications, pp. 99–108. ACM Press, New York (1992)
11. Geman, S., Geman, D.: Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.* **6**, 721–741 (1984)
12. Glover, F.: A template for scatter search and path relinking. In: Hao, J.K., Lutton, E., Ronald, E., Schoenauer, D., Snyers, D. (eds.) *Lecture Notes in Computer Science*, vol. 1363, pp. 13–54 (1997)
13. Glover, F.: Heuristics for integer programming using surrogate constraints. *Decis. Sci.* **8**, 156–166 (1977)
14. Homaifar, A., Turner, J., Ali, S.: The n -queens problem and genetic algorithms. In: Proceedings IEEE Southeast Conference, vol. 1, pp. 262–267 (1992)
15. Hwang, C.L., Yoon, K.: *Multiple Attribute Decision Making-Method and Applications, a State-of-the-Art Survey*. Springer, New York (1981)
16. Jagota, A.: Optimization by reduction to maximum clique. In: IEEE International Conference on Neural Networks (1993)
17. Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: Proceedings of the 6th International Conference on Genetic Algorithms, pp. 184–192. Morgan Kaufmann, San Francisco (1995)
18. Jones, T.: *Evolutionary algorithms, fitness landscapes and search*. PhD thesis, University of New Mexico, Albuquerque, NM (1995)
19. Kale, L.V.: An almost perfect heuristic for the n non-attacking queens problem. *Inf. Process. Lett.* **34**, 173–178 (1990)
20. Khan, S., Bilal, M., Sharif, M., Sajid, M., Baig, R.: Solution of n -queen problem using ACO. In: IEEE 13th International Multitopic Conference, Art. No. 5383157 (2009).
21. Kirkpatrick, S., Gelet, C.D., Vecchi, M.P.: Optimization by simulated annealing. *Science* **220**, 621–630 (1983)
22. Kusters, W.: n -queens bibliography. Retrieved May 4 (2012). <http://www.liacs.nl/~kusters/nqueens/>
23. Laguna, M., Marti, R.: *Scatter Search: Methodology and Implementations in C*. Kluwer Academic, Boston (2003)
24. Lionnet, F.J.E.: Question 963. *Nouv. Ann. Math., Ser. 2* **8**, 560 (1869)
25. Marti, R., Laguna, M., Campos, V.: Scatter search vs. genetic algorithms: an experimental evaluation with permutation problems. In: Rego, C., Alidaee, B. (eds.) *Adaptive Memory and Evolution: Tabu Search and Scatter Search*. Kluwer Academic, Dordrecht (2004)
26. Martinjak, I., Golub, M.: Comparison of heuristic algorithms for the N -queen problem. In: Proceedings of the ITI 2007 29th Int. Conf. on Information Technology Interfaces, Cavtat, Croatia, pp. 25–28 (2007)
27. Nouraniy, Y., Andresenz, B.: A comparison of simulated annealing cooling strategies. *J. Phys. A, Math. Gen.* **31**, 8373–8385 (1998)

28. Pauls, E.: Das Maximalproblem der Damen auf dem Schachbrette, II, Deutsche Schachzeitung. Organ fur das Gesamte Schachleben **29**(9), 257–267 (1874)
29. Rego, C., Leão, P.: A scatter search tutorial for graph-based permutation problems. Research Paper HCES-10-00, Hearin Center for Enterprise Science, University of Mississippi, MS 38677, USA (2009)
30. Rivin, I., Zabih, R.: A dynamic programming solution to the n -queens problem. Inf. Process. Lett. **41**, 253–256 (1992)
31. Russell, S.J., Norvig, P.: Artificial Intelligence a Modern Approach Prentice-Hall, Englewood Cliffs (1995)
32. Segundo, P.S.: New decision rules for exact search in N -queens. J. Glob. Optim. **51**, 497–514 (2011). doi:[10.1007/s10898-011-9653-x](https://doi.org/10.1007/s10898-011-9653-x)
33. Sloane, N.J.A.: The On-Line Encyclopedia of Integer Sequences (2012). <http://oeis.org/A000170>
34. Sosic, R., Gu, J.: Efficient local search with conflict minimization. IEEE Trans. Knowl. Data Eng. **6E**, 661–668 (1994)
35. Taguchi, G., Yokoyama, Y.: Taguchi Methods: Design of Experiments. Am. Supplier Inst. Press, Millersburg (1993)
36. Talbi, E.-G.: Metaheuristics from Design to Implementation. Wiley, Hoboken (2009)
37. Tambouratzis, T.: A simulated annealing artificial neural network implementation of the n -queens problem. Int. J. Intell. Syst. **12**, 739–752 (1997)
38. Tong, L.I., Wang, Ch.H., Chen, H.C.: Optimization of multiple responses using principal component analysis and technique for order preference by similarity to ideal solution. Int. J. Adv. Manuf. Technol. **27**, 407–414 (2005)
39. Wolpert, D.W., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)
40. Xinchao, Z.: Simulated annealing algorithm with adaptive neighborhood. Appl. Soft Comput. **11**, 1827–1836 (2011)
41. Yang, X.-S.: Nature-Inspired Metaheuristic Algorithms. Luniver Press (2010)