DOI: 10.24996/ijs.2024.65.1.41

ISSN: 0067-2904

Improved Naked Mole-Rat Algorithm Based on Variable Neighborhood Search for the N-Queens Problem

Samer Alaa Hussein*, Idrees A. Zahid
Information Technology Center, University of Technology, Baghdad, Iraq

Received: 26/11/2022 Accepted: 11/2/2023 Published: 30/1/2024

Abstract
Solving problems via artificial intelligence techniques has widely prevailed in different aspects. Implementing artificial intelligence optimization algorithms for NP-hard problems is still challenging. In this manuscript, we work on implementing the Naked Mole-Rat Algorithm (NMRA) to solve the n-queens problems and overcome the challenge of applying NMRA to a discrete space set. An improvement of NMRA is applied using the aspect of local search in the Variable Neighborhood Search algorithm (VNS) with 2-opt and 3-opt. Introducing the Naked Mole Rat algorithm based on variable neighborhood search (NMRAVNS) to solve N-queens problems with different sizes. Finding the best solution or set of solutions within a plausible amount of time is the main goal of the NMRAVNS algorithm. The improvement of the proposed algorithm boosts the exploitation capability of the basic NMRA and gives a greater possibility, with the emerging search strategies, to find the global best solution. This algorithm proved successful and outperformed other algorithms and studies with a remarkable target. A detailed comparison is performed, and the data results are presented with the relevant numbers and values. NMRA and NMRAVNS comparisons are implemented and recorded. Later on, a comparison between the Meerkat Clan Algorithm, Genetic Algorithm, Particle Swarm Optimization, and NMRAVNS is tested and presented. Finally, NMRAVNS is evaluated against the examined genetic-based algorithm and listed to prove the success of the proposed algorithm. NMRAVNS outperformed previous findings and scored competitive results with a high number of queen sizes, where an average time reduction reached about 87% of other previous findings.

Keywords: naked mole-rat algorithm, n-queens problems, optimization, artificial intelligence, variable neighborhood search.

خوارزمية فئران الخلد المجردة المحسنة استنادًا إلى البحث المتغير في الجوار لمشكلة مسائلة ملكات الشطرنج

سامر علاء ، ادريس علاء
مركز تكنولوجيا المعلومات, الجامعة التكنولوجية, بغداد, العراق

الخلاصة
لقد ساد حل المشكلات عن طريق تقنيات الذكاء الاصطناعي على نطاق واسع في جوانب مختلفة. لا يزال تطبيق خوارزميات تحسين الذكاء الاصطناعي لمشاكل NP الصعبة يمثل تحديًا. في هذه المخطوطة ، نعمل

Email: samer.a.hussein@uotechnology.edu.iq

على تطبيق Naked Mole-Rat Algorithm (NMRA) لحل مشاكل n-queens والتغلب على التحدي المتمثل في تطبيق NMRA على مجموعة فضاء منفصل. يتم تطبيق تحسين NMRA باستعمال جانب البحث المحلي في خوارزمية بحث الجوار المتغير (VNS) مع (opt-2) و (opt-3) تقديم خوارزمية Naked mole-rat استنادًا إلى البحث المتغير في الأحياء المجاورة (NMRAVNS) لحل مشاكل n-queens ذات الأحجام المختلفة. الهدف الرئيسي لخوارزمية NMRAVNS هو إيجاد الحل الأفضل أو مجموعة الحلول خلال فترة زمنية معقولة. يعزز تحسين الخوارزمية المقترحة من قدرة استغلال NMRA الأساسية ويعطي إمكانية أكبر مع استراتيجيات البحث البارزة للعثور على أفضل حل شامل. أثبتت هذه الخوارزمية نتائج ناجحة وتفوقت على الخوارزميات والدراسات الأخرى المستعملة بهدف رائع. تم إجراء مقارنة مفصلة وعرضت نتائج البيانات مع الأرقام والقيم المعتبرة. يتم تنفيذ وتسجيل مقارنة NMRA و NMRAVNS. في وقت لاحق ، تم اختبار وتقديم مقارنة بين خوارزمية عشيرة السراقيط، والخوارزمية الجينية ، وتحسين سرب الجسيمات ، و NMRAVNS. أخيرًا ، يتم تقييم NMRAVNS مقابل الخوارزمية الجينية المدروسة وإدراجها لإثبات نجاح الخوارزمية المقترحة. تفوقت NMRAVNS على النتائج السابقة وحققت نتائج تنافسية مع عدد كبير من أحجام n-queens حيث وصل متوسط تقليل الوقت إلى حوالي 87 ٪ من النتائج السابقة الأخرى.

1. Introduction

Technology evolves as life expectancy increases. As the amount of artificial intelligence (AI) embedded in various aspects of modern life grows exponentially, various AI techniques for solving, thinking about, and overcoming humanity’s daily problems begin to emerge. Some issues and problems require a swarm intelligence algorithm (SI), which is a particular branch of AI, in order to meditate on a specific issue. SI algorithms mimic natural creatures in which a group cooperates as an individual to achieve its goals [1]. Applying a swarm intelligence algorithm is still challenging and problematic for a specific set of concerns, as it consumes a considerable amount of time.

As a result, a metaheuristic approach is introduced in order to produce the best solution in a reasonable amount of time and space [2]. Advanced swarm algorithms with a metaheuristic approach are mostly applied to solve a hard problem that consumes a considerable amount of time. These are known as "NP-hard problems," and their solutions can be found in non-polynomial time. Solving NP-hard problems still requires a strategic and appropriate approach to finding the best or optimum solutions. Optimization algorithms and techniques are increasingly being used to solve such problems. A number of optimization algorithms have been introduced and proven to be effective in achieving specific goals.

The N-queens problem is one of those NP-hard problems that this manuscript embraces. Given the difficulty of the n-queens problem and the level of complexity it reaches as the value of n increases, it represents a large solution space, particularly the discrete one, to employ improvement and enhancement. The naked mole-rat algorithm is one of the optimization algorithms that has proven its efficiency in finding solutions. Considering the effectiveness of the naked mole-rat algorithm in optimizing the continuous solution space, therefore, an adaptation to work with a discrete solution space is required. Using a variable neighborhood search algorithm (VNS), which employs a local search strategy with multiple alternatives within a discrete search space [3], an integration of VNS with the aforementioned naked mole-rat algorithm is introduced, and an improvement is presented to solve the NP-hard problem. This integration is enhanced and employed in the proposed Naked Mole-Rat Algorithm based on Variable Neighborhood Search (NMRAVNS).

The major highlights of this work are summarized as follows:

- Based on the Naked-Mole Rat algorithm and Variable Neighborhood Search, a new enhanced algorithm is derived.

- Adaptation in the improved algorithm is introduced to tackle the discrete space problem.
- High exploitation has been achieved via employing variable neighborhood search within the improved algorithm.
- Time-wise performance was significantly boosted compared to the traditional algorithms.

In this paper, we will discuss the current knowledge, existing solutions, and our proposed solution for the raised NP-hard problem. Related work is discussed in the next subsection of the introduction, exploring the NMRA literature review. The NP-hard problem and the N-queen problem are presented in the introduction's second subsection. Improvements and enhancements applied to the current algorithms will be presented in the methodology section. The improved algorithm (NMRAVNS) is being tested. Comparisons will be presented in the discussion and results analysis sections. There will also be an explanation of the findings and a results analysis. Finally, a conclusion is induced in the conclusion section.

1.1 NMRA Literature Review

Several algorithms were introduced to imitate the swarm behavior in nature to solve problems in real life and were employed as swarm intelligence optimization algorithms. Among those swarm optimization algorithms are particle swarm optimization (PCO) [4], artificial bee colony (ABC) [5], grey wolf optimization (GWO) [6], and others. The naked mole-rat algorithm (NMRA) is among those swarm optimization algorithms that were introduced, researched, and confirmed for their competitiveness. Numerous types of research involve NMRA, enhancement, and improvement.

The authors of the [7] manuscript discussed how to overcome stagnation in local optima in differential evolution (DE) and poor exploration in NMRA by proposing a newly designed algorithm that updates the NMRA worker phase by incorporating deferential evolution and leaving all parameters alone. Singh et al. [8] used the generated weight map by the modified NMRA to reserve important information in the final image without the use of artifacts; the proposed fNMRA is built on a feature-level fusion when the refinement of the weight maps is done.

[9] proposed an enhanced NMRA (ENMRA) to mitigate the current algorithm limitations like poor exploration, slow convergence, and a deadlock in local optima. In [9], they implemented the search strategy of the grey wolf optimization algorithm (GWO) to improve the exploration for the basic NMRA, differential evolution equations were used to improve exploitation, and a neighborhood search strategy was used to prevent the algorithm from settling in local optima by applying neighborhood search around the best-utilized individual. This paper studies the NMRA's limitations in exploration, improves its search strategy to overcome this limitation, and applies the improved NMRA to an NP-hard problem to verify its enhancement and robustness.

1.2 N-Queens Problem

Consider the NP-Hard problem of n-queens and how difficult, if not impossible, it is to solve using traditional algorithms, as explained in the Sharma et al. study [10]. Since the n-queens problem requires a distribution of the queens on every row so that no queen can attack others in any possible move, applying such conditions makes it hard to find the best solution for the board: placing all the queens in positions such that no other queen can threaten others by moving vertically, horizontally, or diagonally and within a plausible time.

N-queens problems have different variations depending on the value of (n); it could be a 4-queen problem, an 8-queens problem, a 12-queens problem, etc. The important aspect of the n-queens problem deployed in this study is its ability to solve and deal with discrete problem space. Alternatively, other algorithms are used to handle continuous space problems, specifically in communication, network traffic, voice-over-IP (VoIP), etc. The discrete characteristic of n-queens is utilized to deduce the solution for the current discrete problem space.

1.3 Naked mole-rat algorithm

Nature provides the most effective problem solvers and mitigation strategies. One of the swarms' intelligences inspired by nature is the naked mole-rat algorithm (NMRA). This algorithm imitates naked mole-rate creatures and their mating strategies and foraging styles as a group. As described in [11], the naked mole-rat algorithm implements the four rules of the naked mole-rat animals and applies three phases. The four rules are as follows: They live in groups of 70 to 80 rats and can number up to 295; a single queen leads each group and divides them into workers and breeders. Minor tasks are done by a worker, and the best worker joins the breeder subgroup, while the breeder is meant only for the mating and breeding processes. The best-performing member of the breeder group mates with the queen.

The NMRA and its implementation were divided into three phases based on these rules: internalization, the worker phase, and finally the breeder phase. Phase is designed to represent exploration patterns in the NMRA and help the algorithm find good solutions in its search pool.

While the breeder phase defines the exploitation process, its main goal is to provide local search ability within a specific part of the search domain. Collectively, both phases—the worker and breeder phases—combined to define the heuristic algorithm. The NMRA worker phase mechanism is defined by the following equation:

$$W_i^{(t+1)} = W_i^t + \lambda(W_j^t - W_k^t) \tag{1}$$

where w_i^t refers to the *i*th worker within the *t*th iteration, $w_i^{(t+1)}$ is defined as the new solution or worker, λ is the factor of mating, and w_j^t and w_k^t are two random solutions selected from the worker's pool. The K value is obtained from a uniform distribution in the range of [0, 1].
While the breeder phase is defined by equation 2:

$$b_i^{(t+1)} = (1 - \lambda)b_i^t + \lambda(d - b_i^t) \tag{2}$$

Here in equation 2, b_i^t refers to the breeder *i* within the iteration *t*. Breeders' mating frequency is controlled via λ factor, it also helps in identifying and recognizing a new breeder $b_i^{(t+1)}$ in the next coming iteration.

The naked mole-rat algorithm is used in optimization, using different approaches and solutions. NMR is used to optimize energy efficiency in a wireless protocol in WSN [12], used in localization optimization [13], and in antenna design as multi-objective NMRA [14]. The NMR algorithm is also used in feature selection, as in [15]. An enhanced version of the NMR algorithm is used to find the optimal solutions for wireless sensor networks in an underground infrastructure [16]. The naked mole-rat algorithm, like many other swarm algorithms, mimics nature to solve or achieve what it needs, as introduced by Solgotra et al. in their article [11].

Solgotra et al. also mentioned that, according to the experimental results and numbers, the NMR algorithm is very competitive with other heuristic and cutting-edge algorithms. The NMR algorithm, like many other swarm intelligence algorithms, uses intelligence to meet its needs in nature. Accordingly, we studied and implemented the NMR algorithm to solve the N-queens problems, whose hardness and complexity make them not solvable in the traditional ways, at least not in a reasonable time. For a better understanding of NMRA, a pseudo-code of the Naked Mole Rat Algorithm is presented in Algorithm 1:

Algorithm 1. NMRA Algorithm

```
Input: No. of naked mole rats (N), No. of breeders (B: N/5), No. of workers (W: N−B), breeding probability (bp).
Output: Best Solution.
BEGIN
Initialize population of N naked mole rats.
Evaluate the fitness of the population.
d = the best solution
Divide the population into breeders and workers.
WHILE ($t < t_{max}$) DO
FOR i=1 to W DO
perform worker phase: $w_i^{(t+1)} = w_i^t + \lambda (w_j^t - w_k^t)$
evaluate $w_i^{(t+1)}$
END FOR
FOR i=1 to B DO
IF rand > bp
perform breeder phase: $b_i^{(t+1)} = (1- \lambda) w_i^t + \lambda (d - b_i^t)$
evaluate $b_i^{(t+1)}$
END FOR
combine the new worker and breeder population
evaluate the population
Update d if there is a better solution
t=t+1
END WHILE
Return the best solution d
END
```

2. Methodology

The methodology discussed in this manuscript is divided into three sections. The first section explains how to apply NMRA to the N-Queen problem, beginning with solution representation and moving on to the implementation phases, while the second section presents the improvement of NMRA. Finally, the third part will explain parameters and their tuning values for the NMRA algorithm while applying it to solving NP problems.

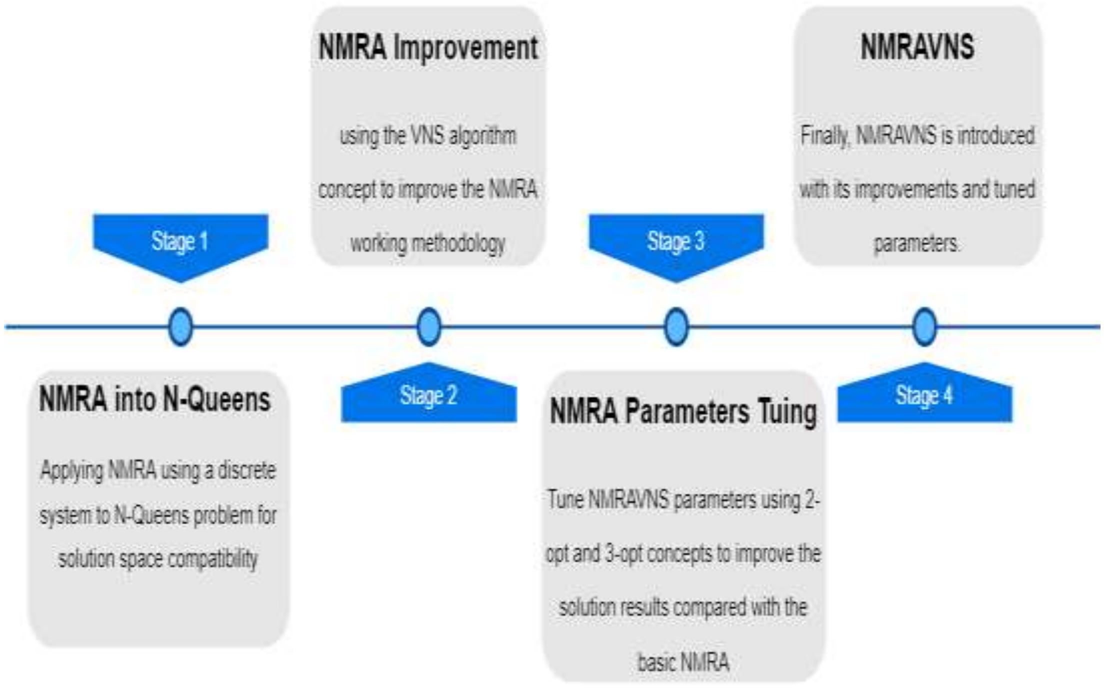
NMRA Improvement
using the VNS algorithm concept to improve the NMRA working methodology
NMRAVNS
Finally, NMRAVNS is introduced with its improvements and tuned parameters.
Stage 1
Stage 3
Stage 2
Stage 4
NMRA into N-Queens
Applying NMRA using a discrete system to N-Queens problem for solution space compatibility
NMRA Parameters Tuing
Tune NMRAVNS parameters using 2-opt and 3-opt concepts to improve the solution results compared with the basic NMRA

Figure 1: Working methodology stages

Figure 1 depicts the stages of the working methodology used in this manuscript. For system compatibility, the first stage involves applying the NMRA algorithm to the N-Queens problem using discrete solution space. The second stage involved NMRA improvement by merging VNS algorithm concepts into the working methodology of the NMRA. Stage 3 includes parameter tuning and adjusting. As the whole concept has been practically implemented using the MATLAB programming environment, NMRA parameters have been tuned, and 2-opt and 3-opt concepts have also been integrated to obtain the best solution space. The fourth stage is having the NMRAVNS algorithm improved and testing the results by comparing the output for different n-queens using the unimproved NMRA. Results obtained using MATLAB were incredibly enhanced and improved, and all the comparisons and results' discussions have been elaborated on in the third section of this paper, titled "Discussion and Results Analysis."

2.1 Applying NMRA to N-queens problem

2.1.1 Solution Representation

As its name suggests, the N-queens problem can have different sizes; Figure 2 illustrates two of them, the 8-queens problem and the 4-queens problem, in Figures 2 (a) and (b), respectively. Other variations of the n-queens problem do exist, and the higher the value of (n), which represents the chessboard dimension, the harder it is to solve. For example, the 16-queens problem is more difficult to solve than an 8-queens problem and hence requires extra time to find the best solutions for the queens' places. There is a direct relationship between the degree of the n-queens problem, i.e., the value of n, and the complexity level it has. Correspondingly, the amount of time required is also increased with a higher value of n. The reason for this direct relationship between the value of n and the complexity level is that for each extra level, the already-placed queens will have a bigger potential effect than any new placement. For example, as Figure 2(a) shows, the queen placed in the 8th cell added a no placement cell that goes all the way across the board horizontally, vertically, and diagonally in all directions. In this case, the restriction added up to the 8 rows beneath the queen.

In Figure 2 (b), the degree of n-queens is 4, implying less complexity. As shown in Figure 2(b), the queen placed in the cell (4b) also has a placement restriction horizontally, vertically, and diagonally, except this time its restriction extended only for 4 rows, which is the degree of this n-queen and hence its complexity for placement of other queens down the board. The time required to find that safe placement solution is much shorter than in the previous eight-queen example.

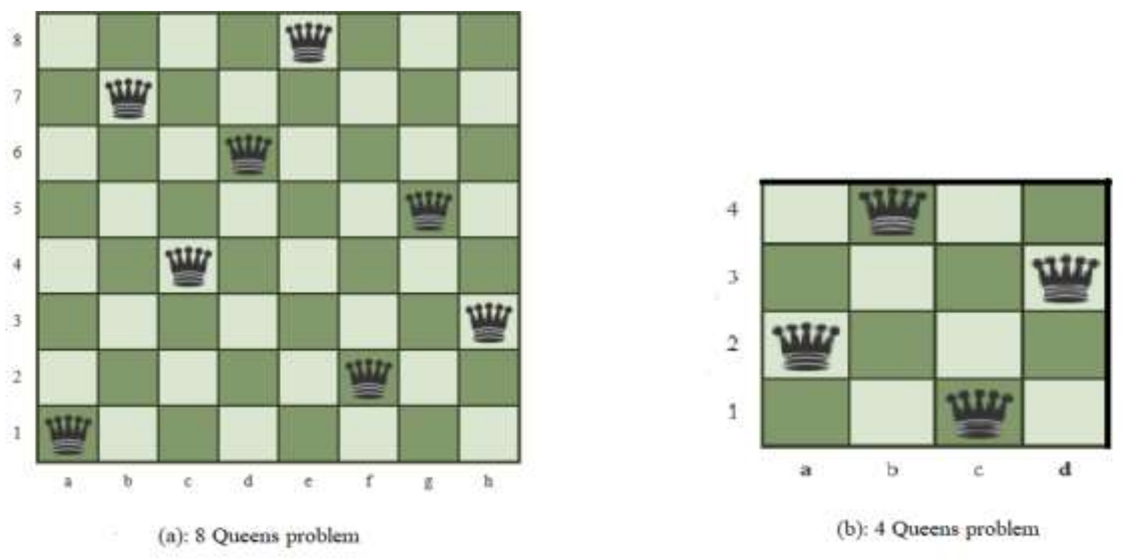

Figure 2: n-queens problem with two examples.

As for NMRA to be applied and utilized to find the solution for N-Queen’s problems, a solution must be presented in a compatible system with regard to the problem itself, as discussed thoroughly above in the introduction. Because N-queens' problems are represented by discrete values for their queens' positions on the board, a solution representation would be presented as a vector of discrete values within the range of the queen problem, i.e., if the problem being solved is for 8 queens, then a range of discrete integer values from 1 to 8 is presented, and similarly for a 16 queen problem, a range of discrete integer values from 1 to 16 is presented.

2.1.2 Implementation Phases

The NMRA algorithm simply tries to find the optimal solution for a general optimization problem, and it does that by working with two categories: breeder and worker categories [17]. It always searches for the best solution within the worker category and moves it to the breeder, and it moves the worst from the breeder to the worker.

In our implementation of the NMRA, we have to work with the same categories but from a discrete standpoint instead of a continuous one. An interpretation and adjustment have to be made for this shift in order to apply the mentioned algorithm to the discrete space set and try to solve the studied problems.

The proposed implementation contains three phases, starting with the first phase of “***subtracting solutions***,” moving to the second phase of “***multiplication by lambda,”*** and then the final phase of "***adding solutions***.” Figure 3 depicts the phases of implementation.

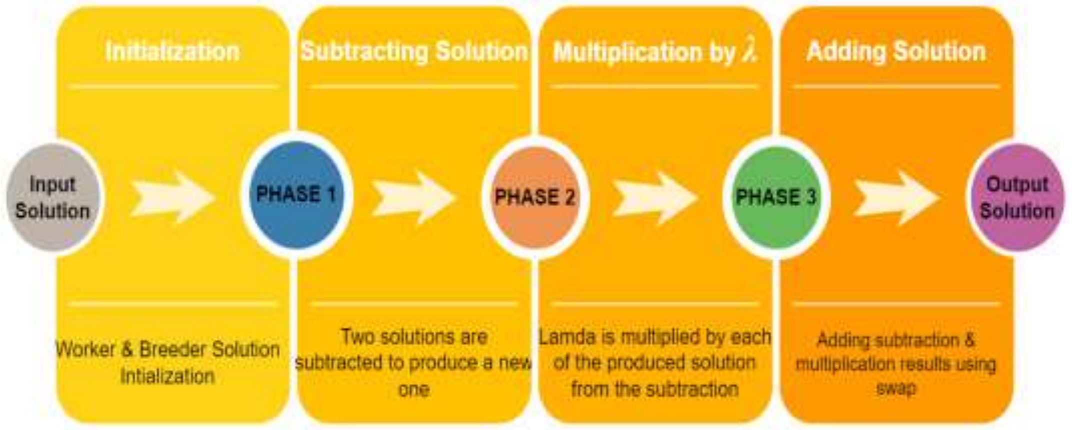
Initialization
Subtracting Solution
Multiplication by λ
Adding Solution
Input Solution
PHASE 1
PHASE 2
PHASE 3
Output Solution
Worker & Breeder Solution Intialization
Two solutions are subtracted to produce a new one
Lamda is multiplied by each of the produced solution from the subtraction
Adding subtraction & multiplication results using swap

Figure 3: NMRA to N-Queens Implementation phases

Figure 3 illustrates briefly the phases used for NMRA implementation into an N-queens problem. Prior to applying the first phase, solution initialization is adopted for the worker and breeder solutions. using those solutions as input for the first phase, i.e., the subtraction phase, where two solutions are subtracted to produce a new one. The third phase then integrates the output of the subtraction phase by multiplying each value of the produced vector with a random value selected from the range of [0, 1] called lambda, denoted by (λ). Using the swap operation concept, the final phase combined both results from the previous phases.

The aforementioned process was denoted by phases, which included several steps. The newly proposed approach for discrete solution implementation is listed in the following detailed phases for a broader understanding of impact:

PHASE 1: Subtracting Solutions:

The NMRA implementation of the first phase starts after the initialization of the initial generation, and the following steps are applied to produce the next generation using equation (1), where phase 1 represents the subtraction portion of equation (1), i.e., $(w_j^t - w_k^t)$

- ***Step 1*****:** Two solutions produced out of the next generation using equation (1) are randomly selected, i.e., S_j and $S_{k.}$
- ***Step 2*****:** The subtracting process is applied by starting an element-wise comparison of the two vectors, i.e., solutions. An element from either solution (S_i, S_k) is picked in case their values are identical, and a zero is placed for a non-matching value to produce the new subtracted solution called $S_{sub.}$

Figure 4 explains phase one of NMRA implementations with the steps mentioned above. S_j and S_k are presented with an example's values. S_{j1} represents the first value in that solution, and S_{j8} indexes the last value of the S_j solutions. Indexing applies for S_k and S_{sub} accordingly.

S_{j1} S_{j2} .... S_{j8}

S_j	2	5	3	7	1	8	4	6
	⇩	⇩	⇩	⇩	⇩	⇩	⇩	⇩

S_{k1} S_{k2} .... S_{k8}

S_k	3	5	6	7	1	2	4	8
S_{sub}	0	5	0	7	1	0	4	0

S_{sub1} S_{sub2} .... S_{sub8}

Figure 4 : PHASE 1: Subtracting Solution Example

PHASE 2: Multiplication by Lambda:

Phase 2 represents the multiplication part of (λ) by $(w_j^t - w_k^t)$ produced in phase 1.

- ***Step 1*:** Lambda (λ) is produced by generating a random floating-point number between 0 and 1 using equation (2):

$$\lambda\,(R) \;\; where\; R \in \{0, \ldots, 1\} \qquad (3)$$

- ***Step 2****:* A new vector is generated with the exact length of the S_{sub} solution obtained in phase 1 and a random floating point number using equation 3 for each element of that vector.
- ***Step 3:*** Using the newly generated lambda (*λ)* from equation (3) as the threshold, initiate element-wise processing for the newly produced vector in step 2 of this phase.
- ***Step 4:*** Produce a New Solution S_{mul} for this generation by applying the following:

If the value of Lambda *(λ)* is bigger than the element of the newly created vector in step (2) then place zero in that index. Otherwise, place the correspondent value produced from phase (1) in that index, i.e., the S_{sub} value.

Steps 1 through 4 for phase 2 are depicted with an example in Figure 5 for a better understanding.

Let Lambda λ = 0.5

S_{sub}	*0*	*5*	*0*	*7*	*1*	*0*	*4*	*0*
rand	*0.81*	*0.9*	*0.12*	*0.91*	*0.27*	*0.09*	*0.63*	*0.54*
S_{mul}	*0*	*5*	*0*	*7*	*0*	*0*	*4*	*0*

Figure 5 : PHASE 2: Multiplication by Lambda Example

PHASE 3: Adding Solutions:

Phase 3 translates the final part of equation (1), the addition part. Where the final result of phase 1's subtraction output alongside the multiplication process of phase 2 is added to W_i^t using the following procedure:

- ***Step 1:*** Mark any non-zero values' positions in S_j that have equal values with S_{mul}.
- ***Step 2****:* Swap the values for those marked positions in Step 1 to produce the final "added solutions," S_{add}.

Figure 6 demonstrates phase 3 with its three steps and an example value for the mentioned solutions and procedures.

S_i	*6*	*1*	*3*	*2*	*4*	*7*	*8*	*5*
S_{mul}	*0*	*5*	*0*	*7*	*0*	*0*	*4*	*0*

- Select non-zero values positions in $\boldsymbol{S_j}$ *that have identical values with* $\boldsymbol{S_{mul}}$

6	*1*	*3*	*2*	*4*	*7*	*8*	*5*

a- Swap the values for those marked positions in the previous step to produce the final the added solution $\boldsymbol{S_{add}}$

	6	*1*	*3*	*2*	*4*	*7*	*8*	*5*
S_{add}	*6*	*5*	*3*	*7*	*8*	*2*	*4*	*1*

Figure 6: PHASE 3: Adding Solutions

As discussed previously, these steps interpret equation (1), the worker part of the NMRA, where two new workers are selected and the steps in each phase from phase 1 to phase 3 are applied to introduce the best worker among the two randomly selected workers. Whereas the breeder part of NMRA is represented by equation (2), a breeder is updated and compared with the best breeder for mating purposes, and the least-performed breeder is reallocated as a worker.

For a complete evaluation of the performance of the improved NMRA algorithm, we applied it to the n-queens problem. From this point, we have settled the parameters for both algorithms: NMRA and the Improved NMRA, i.e., NMRAVNS. Comparisons would be made in the discussion section, later on, to evaluate the performance from several perspectives.

2.2. NMRA Improvement

An improvement is introduced to the implementation of NMRA on discrete values via VNS, where the newly proposed algorithm has emerged. The improvement is represented by implementing the variable neighborhood search (VNS) mechanism for the NMRA to search and find the best solutions from the current population. VNS is mainly utilized for its local search approach [18]. We have implemented the local search aspect via a two-position swap operation, or what is known as a “2-opt,” to look for local optimum, and if it does not find it, a three-position swap, known as a “3-opt,” is applied to better optimize the solution. NMRAVNS has adopted the VNS mechanism in the searching phase for the best solutions from a discrete value set. A pseudo-code of the NMRA implementation for discrete space is presented in Algorithm 2.

Algorithm 2. NMRAVNS

```
Input: No. of naked mole rats (N), No. of breeders (B: N/5), No. of workers (W: N−B), breeding probability (bp).
Output: Best Solution.
BEGIN
Initialize population of N naked mole rats.
Evaluate the fitness of the population.
d = the best solution
Divide the population into breeders and workers.
WHILE ($t < t_{max}$) DO
FOR i=1 to W DO
perform worker phase:  $w_i^{(t+1)} = w_i^t + \lambda (w_j^t - w_k^t)$
evaluate $w_i^{(t+1)}$
END FOR
FOR i=1 to B DO
IF rand > bp
perform breeder phase: $b_i^{(t+1)} = (1- \lambda) w_i^t + \lambda (d - b_i^t)$
evaluate $b_i^{(t+1)}$
END FOR
combine the new worker and breeder population
evaluate the population and find the current best solution $d_{CB}$
Perform VNS Algorithm on $d_{CB}$ and compute $S_{VNS}$
IF $f(S_{VNS}) < f(d_{CB})$
d = $S_{VNS}$
END IF
t=t+1
END WHILE
Return the best solution d
END
```

We use variable neighborhood search (VNS) local search to support performance, enhance local convergence, and minimize the stochastic search probability for better results and outcomes within the NMRA. A proven successful metaheuristic search algorithm is the Variable Neighborhood Search (VNS) algorithm. a widely used search algorithm in optimization problems to find the local optimum. Basically, it employs a local search to find the local optimum. We have implemented VNS local search by using a 2- or 3-opt swap procedure over the best current solution to find an improved solution, i.e., the local optimum [19]. The VNS algorithm is presented in Algorithm 3:

Algorithm 3. VNS algorithm

```
Input: current best solution ($S_{CB}$).
Output: Improved Solution.
BEGIN
k=1
WHILE ($k \leq 2$) DO
IF (k=1)
                    Perform (2-opt) operation on CB, (n) times to obtain Local best solution $S_{LBS}$.
                    IF $f(S_{LBS}) < f(S_{CB})$
                              $S_{CB} = S_{LBS}$
                              k=1
ELSE
          k=k+1
END IF
          ELSE
                    Perform (3-opt) operation on CB, (n) times to obtain Local best solution $S_{LBS}$.
IF $f(S_{LBS}) < f(S_{CB})$
                              $S_{CB} = S_{LBS}$
ELSE
          k=k+1
                    END IF
END IF
END WHILE
return ($S_{CB}$)
```

2.3 NMRAVNS Parameters description, tuning and testing

NMRA is applied to N-Queens problems as illustrated in Section 2.1. Parameters' descriptions and values are recorded. The improved algorithm, NMRAVNS, was applied to the N-queens problem to compare the results. The values set for the NMRA and NMRAVNS are presented in Table 1.

Table 1: Parameters description of NMRA and NMRAVNS algorithms for solving n-queens problems

Algorithm	Parameters	Values
NMRA	Initializations	Random
	Representations	Integer String which is n long
	Population size	10,20, 30, ..100
	Breeding probability	0.5
	Maximum iterations	100,500
NMRAVNS	Initializations	Random
	Representations	Integer String which is n long
	Population size	10,20,30,…,100
	Breeding probability	0.5
	Maximum iterations	100,500
	Max Generation of (2-opt) and (3-opt) neighbors	Population size/2

We can see in Table 1 that both algorithms start by initializing their solutions from a random set. While the representation process of the initialized solutions is an integer string that is n long, the population size begins at 10 and increases in 10-fold increments until it reaches 100. The likelihood of breeding is 50%. Maximum iterations are between 100 and 500, according to the queen size applied. It is 100 for the 8-queen problem and 500 for the 16-queen problem. For the improved NMRA, the maximum number of 2-opt and 3-opt neighbors that it can reach is half the population size. Implementation of the NMRAVNS algorithm over different values of n-queens and results produced by using MATLAB software version R2021b.

3. Discussion and Results Analysis

In this section, the results of the four phases of the proposed methodology are presented and discussed.

3.1 NMRAVNS Improvement time complexity-wise & iteration-wise

NMRAVNS has been applied for 8 and 16 queens, respectively, to test and evaluate its performance from a time-wise perspective and number of iterations with regard to the NMRA algorithm. With population size starting at 10 solutions and stopping at 100 with 10 folds, the number of iterations as well as time for reaching the best solution in seconds have been recorded and presented in Table 2 for both NMRA and NMRAVNS.

Table 2 : Comparison between basic NMRA and the proposed algorithm NMRAVNS for solving 8 and 16 queens problem.

Queen No.	Pop. Size	NMRA		NMRAVNS	
		Iteration No.	**Time (sec)**	**Iteration No.**	**Time (sec)**
8	10	17	0.00323	2	0.00223
	20	9	0.00353	2	0.00147
	30	6	0.00433	3	0.00275
	40	10	0.00728	1	0.00224
	50	5	0.00547	1	0.0021
	60	4	0.00538	1	0.00279
	70	5	0.00726	1	0.00332
	80	4	0.00556	1	0.0027
	90	3	0.00488	1	0.00368
	100	3	0.00592	1	0.00308
16	10	283	0.04906	40	0.00915
	20	224	0.09872	18	0.0091
	30	210	0.12345	38	0.02469
	40	205	0.15742	6	0.0065
	50	119	0.11724	26	0.02914
	60	124	0.15207	20	0.02621
	70	133	0.18125	14	0.02239
	80	125	0.192827	8	0.01505
	90	71	0.12358	13	0.02568
	100	69	0.134807	3	0.00947
Time Avg.			**0.069163**		**0.010187**

Table 2 discusses NMRAVNS enhancement in terms of NMR using two N-queens problems (8 and 16). A comparison was made from two points of view to illustrate the results' improvement: time-wise and in terms of the number of iterations required to reach the best solution. For reliable and more accurate results, an average has been taken for each recorded value instead of picking the best one. This average is taken out of ten trials for better casting and increased accuracy.

We clearly see in Table 2 for the 8-queen problem that NMRAVNS outperforms NMRA regarding the time required to find the best solution as well as the number of iterations. Time-wise performance enhancement roughly ranges from 0.001 (second) to 0.005 (second). The number of iterations required for each fold of the population size is incredibly decreased for the 8-queen problem and for the 10-fold population size, with a decrement in the number of iterations ranging from 15 iterations for the same population size to a 2-iteration difference for the lowest enhancement.
Table 2 shows that NMRAVNS outperformed NMRA in terms of time and number of iterations for the 16-queen problem. The biggest time difference was recorded in a population of 80 with a 0.177777-second difference in 16 queen problems. The largest number of iterations being dropped when NMRAVNS is applied is 243 iterations in the first fold with a population size of 10 within 16 queen problems.

For an overall evaluation and understanding of the current comparison in Table 2, an average execution time is calculated for the recorded time for each fold of the population until the best solution is produced by both algorithms, i.e., NMRA and NMRAVNS, for the two cases of 8 and 16 queens, respectively. A difference of 0.058976 is found when NMRAVNS is implemented, as its average time recorded was 0.010187, while for NMRA it was 0.069163. Mentioning the average time for both comparison parties, the base NMRA and the improved algorithm NMRAVNS, we can use equation 3 to calculate the time reduced upon implementing the improved algorithm and for the N-queen cases mentioned in Table 2, where we will use the average time for both aforementioned algorithms.

$$\text{Time Reduction (TR)} = \left(\frac{Base\ Time - Improved\ Time}{Base\ Time}\right) * 100 \tag{4}$$

Upon applying equation (4) to the numbers obtained in Table 2, where base time represents the average time used to solve N-queens problems employing NMRA, which is 0.069163, and improved time is the average time used for the same instances employing NMRAVNS, a value of 85.27102642, which is an 85.27% improvement in time reduction, is obtained when comparing the basic NMRA and the improved NMRAVNS.

3.2 NMRAVNS vs other algorithms for solving 8 & 16 queens problem

Variant comparisons have to be made to obtain reliable and precise results, which in turn gives a better evaluation for the implementation of the proposed algorithm. From this point of view, a comparison has been made between NMRAVNS and other related algorithms such as the Meerkat Clan Algorithm (MCA), Genetic Algorithm (GA), and Particle Swarm Optimization (PSO), all of which have solved NP-hard problems such as n-queens problems and others as a common ground for comparison [20]–[22]. These mentioned algorithm solutions applied to the n-queens problem are addressed in [23].

Time and number of iterations were used as comparison metrics for the mentioned algorithms and NMRAVNS. When an 8-queen problem is addressed, NMRAVNS outperforms PSO and GA algorithms in both metrics (time and number of iterations). As per MCA, it's also outperformed via NMRAVNS in time regards, and most of the iteration numbers are also outperformed, or at least have the same number of iterations for both. For the 16-queen problem, it is obviously noticeable that NMRAVNS is incredibly faster than MCA, GA, and PSO for the denoted population size. Generally, NMRAVNS's average time is 0.01018 seconds, while the closest average time was obtained from MCA, which recorded 0.0449 seconds. GA results in 2.7215 second as an average time, while PSO average time was 62.18117.

Table 3: Comparison between NMRAVNS and some algorithms for solving 8 and 16 queens problem.

Queen No.	Pop. Size	NMRAVNS		MCA [23]		GA [23]		PSO [23]	
		Iteration No.	**Time (sec)**	**Iteration No.**	**Time (sec)**	**Iteration No.**	**Time (sec)**	**Iteration No.**	**Time (sec)**
8	10	2	0.00223	5	0.028	12	1.8485	13	1.1977
8	20	2	0.00147	2	0.013	7	1.7053	4	1.3741
8	30	3	0.00275	1	0.017	3	1.543	2	1.5269
8	40	1	0.00224	2	0.027	3	1.3966	4	1.4405
8	50	1	0.0021	2	0.014	5	1.4659	3	1.3814
8	60	1	0.00279	1	0.032	2	1.4373	3	1.4725
8	70	1	0.00332	1	0.022	2	1.9066	4	1.621
8	80	1	0.0027	2	0.02	3	1.9255	3	1.8541
8	90	1	0.00368	1	0.017	3	2.3777	4	2.2561
8	100	1	0.00308	1	0.019	3	2.4925	3	2.1666
16	10	40	0.00915	20	0.029	60	1.9999	23547	61.9999
16	20	18	0.0091	24	0.049	70	2.9999	18444	95.9999
16	30	38	0.02469	17	0.053	39	2.3333	10003	76.9999
16	40	6	0.0065	14	0.061	31	3	143366	147.6666
16	50	26	0.02914	14	0.069	25	2.3333	15150	194.6665
16	60	20	0.02621	10	0.105	24	2.6666	13247	203.9999
16	70	14	0.02239	8	0.061	35	4.6666	3853	69.6666
16	80	8	0.01505	8	0.067	36	4.9999	4443	91.3333
16	90	13	0.02568	8	0.096	30	4.6666	7160	167.3333
16	100	3	0.00947	6	0.1	39	6.6666	4548	117.6666
Time Avg.			**0.01018**		**0.0449**		**2.7215**		**62.18117**

A visual graph depicting the improvement listed in Table 2 is presented in Figures 7 and 8 for time-wise improvement for ten sets of population size variation for both the 8 queens and 16 queens problems, respectively.

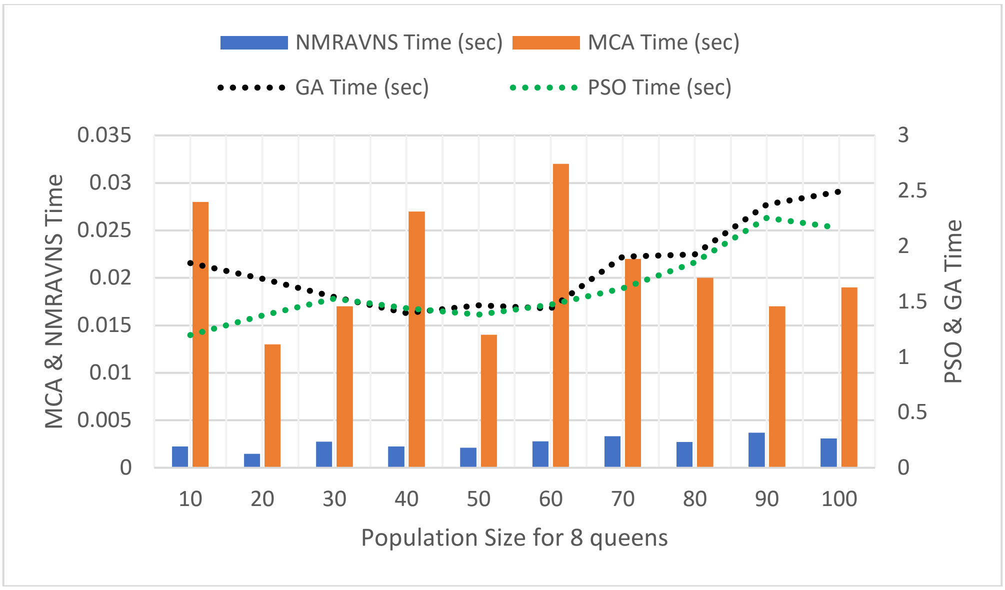

Figure 7: NMRAVNS improvement for 8 queens

Figure 7 illustrates the improvement NMRAVNS has achieved from a time-wise aspect in order to find the best solution compared to the MCA, GA, and PSO algorithms. This figure depicts Table 3 in a graphical illustration to focus more on the time elapsed for NMRAVNS during its implementation to apply the methodology stages. Considering PSO and GA's high values compared to NMRAVNS and MCA's low values, that issue presented both PSO and GA as outliers. To clearly demonstrate NMRAVNS improvements in charts and to overcome the outlier issue, a secondary vertical axis is added to handle the outliers' values in PSO and GA.

Two variations of chart types have been used in figure 7 to give more coordination and focus related to the primary vertical axis named (MCA & NMRAVNS Time) and the secondary vertical axis named (PSO & GA Time). It is clearly obvious that NMRAVNS, labeled with a blue clustered column, reached the best solution in a much less plausible time than MCA, labeled with an orange clustered column, and both are related to the primary vertical axis on the left of figure 7. NMRAVNS also outperforms GA and PSO labels with a dotted black line and a dotted green line, respectively, and they are both related to the secondary vertical axis to the right of figure 7. Figure 7 depicts the testing for 10 population slots in the 8 queens problem.

Similarly, a visual chart is depicted in figure 8 for the improvement made and listed in Table 3, and it compares NMRAVNS to MCA, GA, and PSO as in Figure 7, except this time it is tested for the 16 queens problem. The secondary vertical axis is also used to overcome the outliers issue discussed in Figure 7, and variations of chart type are also employed to highlight the difference and focus more on the different values depicted in the figure.

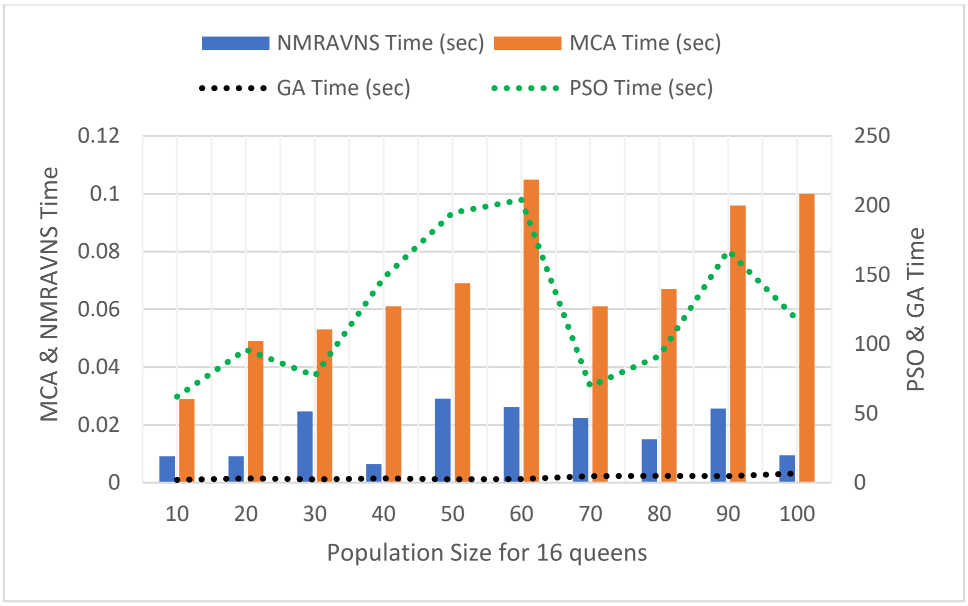

Figure 8 : NMRAVNS improvement for 16 queens

3.3 NMRAVNS vs other algorithms for solving N queens problem

The performance and result enhancement of NMRAVNS implementation compared to n-queens problems via other algorithms is also demonstrated. Searching literature on solving the n-queens problem by applying optimization algorithms, Jane and Prasad [21], discussed the n-queens problem and proposed a new algorithm for finding the set of solutions considering the previous findings. A comparison is listed in Table 4 with the newly proposed genetic algorithm NPGA [21], which already outperforms SPRSO, Per-PSO, Old-GA, Lijo V.P., and Jasmin T. Jose. Our discrete implementation of NMRAVNS outperformed the findings in [21] and their NPGA, as evidenced by the NPGA average time of 2.096675, whereas the NMRAVNS average time for the same 8, 13, 14, 16, 30, 40, 50, and 100 queens is 0.2719. The induvial time for each queen size mentioned earlier is listed in Table 4 along with the findings of NPGA and NMRAVNS.

Table 4 : Comparison between NMRAVNS and some algorithms for solving n-queens problem

Queen No.	NMRAVNS Time (sec)	SRPSO [21] Time (sec)	Per-PSO [21] Time (sec)	Old-GA [21] Time (sec)	NPGA [21] Time (sec)	Lijo V. P. and Jasmin T. Jose [21]
8	0.00147	N/A	N/A	N/A	0.0307	N/A
13	0.0809	N/A	N/A	N/A	0.0492	44.14
14	0.14243	N/A	N/A	N/A	0.0527	87.56
16	0.0065	N/A	N/A	N/A	0.0972	N/A
30	0.06239	6.59	10.32	17.29	0.342	N/A
40	0.10911	23.73	34.3	35.66	0.7885	N/A
50	0.34003	40.12	53.25	54.43	1.0443	N/A
100	1.43237	N/A	N/A	N/A	14.3688	N/A
Time Avg.	**0.2719**				**2.096675**	

To calculate the reduced time when employing NMRAVNS versus the proposed algorithm by [21], i.e., NPGA, we implement equation (4) to calculate time reduction for the average time required for NMRAVNS and NPGA for the n-queens cases as mentioned in Table 4. Using equation (4), a time reduction of 87.03185 seconds was obtained by setting the base time for NPGA to 2.09675 seconds and improving the time representing NMRAVNS to 0.2719 seconds. Evaluating the reduced time, about 87% of the time has been reduced by applying NMRAVNS over NPGA for the same instances of n-queens cases mentioned in Table 4.

4. Conclusion

The conclusion for this manuscript is divided into four parts. The first is that using NMRA to derive discrete values for the N-queens problem is possible. The second conclusion is regarding the high performance NMRAVNS has recorded toward solving this NP-hard problem. The discussion and results analysis sections demonstrate that NMRAVNS is applicable to solving N-queen's problems. Furthermore, improvements are highly remarkable time- and iteration-wise when compared to the NMRA. Third, a conclusion that NMRAVNS outperformed MCA, GA, and PSO for 8 and 16 queen problems is reached. Lastly, NMRAVNS findings outperformed previously listed findings via their NPGA and their compared algorithms by reducing 87% of the average time required to find the same solutions for 8, 13, 14, 16, 30, 40, 50, and 100 queen problems.

References

[1] M. Schranz *et al.*, "Swarm Intelligence and cyber-physical systems: Concepts, challenges and future trends," *Swarm Evol. Comput.*, vol. 60, p. 100762, Feb. 2021, doi: 10.1016/ J.SWEVO.2020.100762.

[2] N. M. Hijazi, H. Faris, and I. Aljarah, "A parallel metaheuristic approach for ensemble feature selection based on multi-core architectures," *Expert Syst. Appl.*, vol. 182, p. 115290, Nov. 2021, doi: 10.1016/J.ESWA.2021.115290.

[3] P. Hansen, N. Mladenović, J. Brimberg, and J. A. M. Pérez, "Variable Neighborhood Search," *Int. Ser. Oper. Res. Manag. Sci.*, vol. 272, pp. 57–97, 2019, doi: 10.1007/978-3-319-91086-4_3.

[4] R. Ramteke, S. Singh, and A. Malik, "Optimized routing technique for IoT enabled software-defined heterogeneous WSNs using genetic mutation based PSO," *Comput. Stand. Interfaces*, vol. 79, Jan. 2021, doi: 10.1016/j.csi.2021.103548.

[5] H. Jahangir and D. Rezazadeh Eidgahee, "A new and robust hybrid artificial bee colony algorithm – ANN model for FRP-concrete bond strength evaluation," *Compos. Struct.*, vol. 257, p. 113160, Feb. 2021, https://doi.org/10.1016/J.COMPSTRUCT.2020.113160.

[6] Y. Fu, H. Xiao, L. H. Lee, and M. Huang, "Stochastic optimization using grey wolf optimization with optimal computing budget allocation," *Appl. Soft Comput.*, vol. 103, p. 107154, May 2021, https://doi.org/10.1016/j.asoc.2021.107154.

[7] R. Salgotra, U. Singh, G. Singh, N. Mittal, and A. H. Gandomi, "A self-adaptive hybridized differential evolution naked mole-rat algorithm for engineering optimization problems," *Comput. Methods Appl. Mech. Eng.*, vol. 383, Sep. 2021, https://doi.org/10.1016/j.cma.2021.113916.

[8] S. Singh, N. Mittal, and H. Singh, "A feature level image fusion for IR and visible image using mNMRA based segmentation," *Neural Comput. Appl. 2022*, pp. 1–18, Jan. 2022, https://doi.org/10.1007/S00521-022-06900-7.

[9] P. Singh, R. P. Singh, Y. Singh, J. Shafi, and M. F. Ijaz, "An enhanced naked mole rat algorithm for optimal cross-layer solution for wireless underground sensor networks," *Mathematics*, vol. 9, no. 22, p. 2942, Nov. 2021, https://doi.org/10.3390/math9222942.

[10] S. Sharma and V. Jain, "Solving N-Queen Problem by Genetic Algorithm using Novel Mutation Operator," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1116, no. 1, p. 012195, 2021, https://doi.org/10.1088/1757-899x/1116/1/012195.

[11] R. Salgotra and U. Singh, "The naked mole-rat algorithm," *Neural Comput. Appl.*, vol. 31, no. 12, pp. 8837–8857, Dec. 2019, https://doi.org/10.1007/S00521-019-04464-7.

[12] V. Nehra, A. K. Sharma, and R. K. Tripathi, "NMR inspired energy efficient protocol for heterogeneous wireless sensor network," *Wirel. Networks*, vol. 25, no. 6, pp. 3689–3700, Aug. 2019, https://doi.org/10.1007/s11276-019-01963-2.

[13] P. Singh, P. Singh, and N. Mittal, "Optimized localization using naked mole-rat algorithm in dynamic wireless sensor networks," *Int. J. Commun. Syst.*, vol. 34, no. 10, Jul. 2021, https://doi.org/10.1002/dac.4832.

[14] G. Singh and U. Singh, "Multi-objective Naked Mole-Rat Algorithm for UWB Antenna Design," *IETE J. Res.*, 2021, https://doi.org/10.1080/03772063.2021.1912657.

[15] S. Guha, A. Das, P. K. Singh, A. Ahmadian, N. Senu and R. Sarkar, "Hybrid Feature Selection Method Based on Harmony Search and Naked Mole-Rat Algorithms for Spoken Language Identification From Audio Signals", *IEEE Access*, vol. 8, pp. 182868-182887, 2020, https://doi.org/10.1109/ACCESS.2020.3028121.

[16] P. Singh, R. Singh, Y. Singh, J. Shafi, and M. Ijaz, 2021, "An Enhanced Naked Mole Rat Algorithm for Optimal Cross-Layer Solution for Wireless Underground Sensor Networks," *Mathematics*, vol. 9, no. 22, pp. 2942, 2021, https://doi.org/10.3390/math9222942.

[17] P. Singh, N. Mittal, U. Singh, and R. Salgotra, "Naked Mole-Rat Algorithm with Improved Exploration and Exploitation Capabilities to Determine 2D and 3D Coordinates of Sensor Nodes in WSNs," *Arab. J. Sci. Eng.*, vol. 46, no. 2, pp. 1155–1178, Feb. 2021, https://doi.org /10.1007/S13369-020-04921-9.

[18] S. Nunes Bezerra, M. J. F. Souza, S. R. de Souza, and V. Nazário Coelho, "A VNS-Based Algorithm with Adaptive Local Search for Solving the Multi-Depot Vehicle Routing Problem," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11328 LNCS, pp. 167–181, 2019, https://doi.org/10.1007/978-3-030-15843-9_14.

[19] X. Xu, J. Li, and M. C. Zhou, "Bi-Objective Colored Traveling Salesman Problems," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 6326–6336, Jul. 2022, https://doi.org /10. 1109/TITS.2021.3086625.

[20] S. A. Hussein and A. Y. Yousif, "An Improved Meerkat Clan Algorithm for Solving 0-1 Knapsack Problem", *Iraqi J. Sci.*, vol. 63, no. 2, pp. 773–784, Feb. 2022, https://doi.org /10. 24996/ijs.2022.63.2.32.

[21] V. Jain and J. S. Prasad, "Solving N-queen Problem Using Genetic Algorithm by Advance Mutation Operator," *Int. J. Electr. Comput. Eng.*, vol. 8, no. 6, p. 4519, 2018, https://doi.org /10. 11591/ijece.v8i6.pp4519-4523.

[22] M. K. Marichelvam, M. Geetha, and Ö. Tosun, "An improved particle swarm optimization algorithm to solve hybrid flowshop scheduling problems with the effect of human factors – A case study," *Comput. Oper. Res.*, vol. 114, p. 104812, Feb. 2020, https://doi.org/10.1016 /J.COR.2019.104812.

[23] S. M. Ali, N. T. Mahmood, and S. A. Yousif, "Meerkat clan algorithm for solving N-queen problems," *Iraqi J. Sci.*, vol. 62, no. 6, pp. 2082–2089, 2021, https://doi.org /10.24996/ijs. 2021.62.6.33.