Anchor University Journal of Science and Technology (AUJST)
A publication of the Faculty of Science Applied and Health Science, Anchor University Lagos
URL: journal.aul.edu.ng
In AJOL: https://www.ajol.info/index.php/aujst
Vol. 7 No 1, January 2026, Pp. 28-37
ISSN: 2736-0059 (Print); 2736-0067 (Online)

A Comparative Review of Trajectory-Based Metaheuristic Algorithms

[1]Asegunloluwa E. Babalola

[1]Department of Computing, Faculty of Science, Anchor University Lagos, Nigeria.

***Corresponding author Email**: Adeosuntobi995@gmail.com

Submitted 01 October, 2025
Accepted 01 November, 2025

Competing Interests.
The authors declare no competing interests

ABSTRACT

Optimization is a central challenge in scientific discovery, engineering design, and decision-making. While population-based metaheuristics such as Genetic Algorithms and Particle Swarm Optimization have received significant attention, Trajectory-based algorithms remain relatively underrepresented despite their simplicity and efficiency. Trajectory algorithms operate by iteratively refining a single solution, tracing a path through the search space, and balancing exploration with exploitation through localized decision rules. This review study revisits four classical trajectory algorithms; Simulated Annealing (SA), Hill Climbing (HC), Tabu Search (TS), and the Great Deluge Algorithm (GDA), and provides a comparative review of their mechanisms, strengths, and limitations. Simulated Annealing balances exploration and exploitation through a temperature-controlled acceptance rule, while Hill Climbing represents a purely greedy search that quickly converges to local optima. Tabu Search improves upon this by using adaptive memory structures to avoid cycling and diversify the search, and the Great Deluge Algorithm employs a deterministic threshold (water level) that steadily rises to control solution acceptance. Each algorithm is analyzed with reference to its pseudocode, showing how exploration and exploitation are achieved in practice. The review highlights why these algorithms, once overshadowed by population-based methods, are gaining renewed attention. Their low memory footprint, computational efficiency, and ease of hybridization make them suitable for modern contexts.

Keywords. Metaheuristics; Trajectory-based algorithms; Optimization; Simulated Annealing; Tabu Search; Great Deluge Algorithm.

1. Introduction

Optimization lies at the heart of scientific discovery, engineering design, and decision-making, as it involves selecting the best solution from a set of feasible alternatives to maximize or minimize a particular objective function (Tartibu, 2025). With the increasing complexity and data-rich nature of modern systems, traditional approaches such as linear programming and convex analysis often struggle to address high-dimensional, non-convex, and dynamically changing problems. These limitations have led to the rise of metaheuristics, which provide flexible and robust approaches for solving challenging optimization tasks.

Metaheuristics are high-level algorithmic frameworks designed to guide underlying heuristics in efficiently exploring large search spaces and approximating global optima (Darvishpoor et al., 2023). They have proven particularly effective in handling combinatorial, multi-objective, and black-box optimization problems where analytical solutions are not feasible. Over the past decades, metaheuristics have found applications in diverse domains such as logistics, scheduling, energy systems, biomedical engineering, finance, and artificial intelligence (Lameesa et al., 2024). Recent surveys further emphasize the rapid expansion of the field: between 2019 and 2024, more than 150 new metaheuristic algorithms were introduced, highlighting both the creativity and fragmentation that characterize this research area (Dokeroglu et al., 2024; Wang et al., 2025).

Metaheuristics are generally divided into two categories: population-based and trajectory-based algorithms (Almufti et al., 2023). Population-based approaches, such as Genetic Algorithms (GAs), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Differential Evolution (DE), operate by evolving a population of candidate solutions through cooperation and information exchange. While powerful in exploration, these approaches are often computationally demanding.

Trajectory-based algorithms, on the other hand, focus on the iterative improvement of a single candidate solution. Classical examples include Simulated Annealing (SA), Hill Climbing (HC)

https://dx.doi.org/10.4314/aujst.v7i1.4

, Tabu Search (TS), Record-to-Record Travel (RRT), and the Great Deluge Algorithm (GDA). These methods trace a continuous path or trajectory through the solution space, making localized adjustments that gradually steer the search toward an optimum. Their simplicity, low memory requirements, and relatively fast convergence make them particularly well suited for constrained environments such as embedded systems, edge computing, and real-time applications.

The designation trajectory algorithms arise from their operational principle. Rather than evolving a population, they guide a single solution along a continuous path through the search space. This metaphor draws from classical optimization and control theory, where trajectory optimization refers to identifying a path that maximizes performance while satisfying dynamic and boundary constraints, such as in robotics or aerospace engineering. In metaheuristics, this path-oriented approach emphasizes stepwise improvement influenced by the current state, local neighborhoods, and adaptive perturbations, effectively tracing a trajectory across the solution landscape.

Despite their early prominence, trajectory algorithms became less popular over time, especially in the 1990s and 2000s, when population-based algorithms dominated optimization research. The growing appeal of evolutionary algorithms, swarm intelligence, and other population-based methods lay in their stronger global exploration ability, robustness across diverse problem landscapes, and ease of hybridization with other techniques (Selvarajan, 2024). In contrast, trajectory methods were sometimes perceived as prone to premature convergence and limited exploration since they relied on a single search path. The research community's preference for algorithms that could balance exploration and exploitation on a larger scale further marginalized trajectory-based approaches during this period.

In recent years, however, trajectory algorithms have gained renewed attention due to their efficiency and adaptability. Their lightweight nature and structure integrates seamlessly with neural networks and reinforcement learning frameworks. Moreover, they have shown strong potential as components in hybrid and ensemble strategies, complementing population-based methods to achieve more balanced optimization outcomes.

Against this backdrop, this paper provides a comprehensive and up-to-date review of trajectory algorithms, reconnecting their classical foundations with recent innovations. It analyzes their theoretical underpinnings, examines modern adaptations, and compares their mechanisms for balancing exploration and exploitation.

2. **Literature Review**

This study focuses on four trajectory-based algorithms—Simulated Annealing, Hill Climbing, Tabu Search, and the Great Deluge Algorithm because they represent the historical foundations, methodological diversity, and contemporary relevance of this class of metaheuristics. Each embodies a distinct strategy such as probabilistic acceptance (SA), deterministic exploitation (Hill Climbing), memory-based search (TS), and threshold-driven acceptance (GDA).

2.1. Simulated Annealing (SA)

Simulated Annealing (SA) was introduced by Kirkpatrick et al., (1983), inspired by the physical process of annealing in metallurgy (Franzin & Stützle, 2023). Annealing involves heating a material and slowly cooling it under controlled conditions to increase crystal size and reduce structural defects. This gradual cooling process strengthens the material, whereas rapid cooling produces an amorphous structure, representing a metastable state similar to a local minimum in optimization (Guilmeau et al., 2021)..

In the SA algorithm, the objective function corresponds to the energy of the material, and a fictitious temperature parameter guides the search. This parameter allows the algorithm to accept not only better solutions but occasionally worse ones, providing an escape from local optima. The cooling schedule, which defines how temperature decreases over iterations, is a critical component: rapid cooling risks premature convergence, while overly overly slow cooling increases computational cost. Figure. 1 shows the

https://dx.doi.org/10.4314/aujst.v7i1.4

pseudocode of SA.

SA is modeled as a stochastic process based on Markov chains, where the probability of the next state depends only on the current state. This property makes it effective for both discrete and continuous optimization problems. Its greatest advantage lies in avoiding local optima, unlike purely deterministic or gradient-based approaches. However, its disadvantages remain the need for careful tuning of cooling schedules and relatively high computational intensity (Mahdi et al., 2017).

Today, SA remains a foundational trajectory-based algorithm, widely studied both as a standalone method and as a component in hybrid frameworks. While it is no longer the dominant metaheuristic, it continues to serve as a benchmark for evaluating newer algorithms due to its simplicity and strong theoretical foundations. Recent research emphasizes hybridization, where SA is combined with evolutionary algorithms, swarm intelligence, or reinforcement learning to improve performance in large-scale and high-dimensional problems (Jian et al., 2020). It is also being adapted for modern application domains, including cloud computing resource allocation (Abdulhamid et al., 2016), energy system optimization (Wang & Li, 2022), and machine learning tasks such as feature selection and hyperparameter tuning.

Start

Initialize population and parameters

Generate randomly an initial optimal state S_i,

Calculate fS_i

Select an initial temperature T_0

Select a terminal temperature T_f or a total number of temperature t_{max}

Set temperature change counter $t = 1$

While $T_i < T_f$ or $t = t_{max}$

Set repetition counter $L = 0$

Repeat until $L = L_t = \beta_t$

Generate new state S_j, a neighbor of (S_i)

Calculate $\Delta E = f(S_j) - f(S_i)$

If $\Delta E < 0$, **then**

$S_i = S_j$

Else if random (0,1) < $Exp\left(-\frac{\Delta E}{K_b T}\right)$, **then**

$S_i = S_j$ where K_b is Boltzmann's constant

End if

$L = L + 1$

End repeat

$t = t + 1$

$T_i = \alpha T_i$, where α is the cooling rate

End While

Figure 1. Pseudo code of Simulate Annealing

https://dx.doi.org/10.4314/aujst.v7i1.4

2.2. Hill Climbing

Hill Climbing is one of the simplest and earliest search and optimization algorithms, primarily applied to single-objective functions (Russell & Norvig, 2016). It operates as an iterative improvement technique, beginning with a randomly generated solution and incrementally modifying it to explore neighboring states. At each step, the algorithm evaluates candidate neighbors and selects the one with the best value to replace the current solution. This process continues until no further improvement is possible as depicted in Figure 2.

The algorithm is effective for reaching a local optimum, though it cannot guarantee convergence to a global optimum. A local optimum is a solution that cannot be improved by any of its neighbors, whereas a global optimum is the best possible solution across the entire search space. Because Hill Climbing only explores the uphill direction, always moving to better neighbors, it is highly susceptible to getting stuck in local optima (Palmer, 2018).

Hill Climbing uses a parameter-free search procedure, where the initial candidate solution is generated randomly, and unary operations are used to produce offspring. Its major strength lies in its low computational requirements, as it only evaluates the current solution and its immediate neighbors. This efficiency makes it appealing in constrained environments. However, its main drawback is premature convergence, as it easily becomes trapped in suboptimal solutions.

Although Hill Climbing is no longer considered a cutting-edge optimization algorithm, it remains significant as a foundational technique in artificial intelligence and heuristic search. Modern research frequently uses Hill Climbing as a baseline for evaluating more advanced metaheuristics or integrates it into hybrid algorithms to strengthen local exploitation.

2.3. Tabu Search

Tabu Search (TS) was developed by Fred Glover in 1986 (not 1896, as often mistakenly cited) as an enhancement of Hill Climbing to overcome the problem of entrapment in local optima (Hanafi et al., 2023), as shown in Figure 3. The term tabu originates from Polynesian tradition, where it refers to something sacred and prohibited. In the algorithmic context, this metaphor was adopted to denote prohibited or restricted moves.

The defining feature of TS is its use of adaptive memory structures, most notably the Tabu List. These short-term memory records recently visited solutions or attributes of moves, preventing the search from revisiting them and thereby avoiding cycles (Glover et al., 2018). Beyond short-term memory, TS incorporates intermediate-term memory to guide the search toward promising regions and long-term memory to encourage diversification, balancing intensification and exploration of the search space.

This memory-based approach distinguishes TS from simple trajectory-based methods like Hill Climbing. TS achieves a dynamic balance between intensification (exploiting good areas of the search space) and diversification (exploring new areas), by learning from past

Begin
Initialize population and parameters
$x_k = LB_k + UB_k - LB_k) \times U(0,1), \forall_k = 1,2,\ldots N$ (The initial state of x)
Calculate $f(x)$
While (until termination condition)
$x' = improved\,\mathrm{N}(x)$
If $f(x') \leq f(x)$ then
$x = x'$
End if
End While

Figure 2. Pseudocode of Hill Climbing

https://dx.doi.org/10.4314/aujst.v7i1.4

moves. These mechanisms make it highly versatile and effective in solving complex optimization problems.

Modern research often integrates TS with other metaheuristics or machine learning techniques to improve global exploration while retaining its strong local search ability (Silva et al., 2020). It is also frequently applied to large-scale scheduling, network routing, and combinatorial optimization in logistics and supply chains (Lin et al., 2021). The adaptability of TS through memory structures makes it especially well-suited for problems requiring balance between exploration and exploitation. As such, it remains a benchmark trajectory-based algorithm and a critical component in the design of hybrid optimization frameworks.

2.4. Great Deluge Algorithm (GDA)

The Great Deluge Algorithm (GDA) was introduced by Dueck (1993) as a trajectory-based local search method inspired by a natural metaphor (Acan & Ünveren, 2015). The analogy envisions a person caught in a great flood, climbing upward to avoid getting their feet wet. Initially, the person can move freely across the landscape, but as the water level rises, low-lying areas become inaccessible, forcing the person to search for paths toward higher ground until eventually reaching a peak (Arbaoui et al., 2022).

In GDA, solutions are accepted if their objective values are less than or equal to a dynamically changing threshold, called the level. The level begins with the objective function value of the initial solution and is iteratively increased (or decreased, depending on problem orientation) by a constant factor β as the algorithm progresses as depicted Figure 4. This mechanism allows the algorithm to accept worse solutions than the current best, provided they meet the level criterion, which prevents premature convergence to local optima. Unlike Hill Climbing, which strictly accepts better neighbors, or Simulated Annealing, which relies on probabilistic acceptance rules, GDA's acceptance is deterministic and guided by the rising water level analogy.

Although elegant in design, classical GDA has limitations, most notably its tendency to stagnate in local minima if the level adjustment is not tuned carefully. To address this, several variants have been proposed, including the the Nonlinear Great Deluge Algorithm, Extended

```
Begin
    Select an initial solution; S0
    Initialize Tabu List; TabuList = null
    Set maxTabusize
    While (until termination condition ( ) )
        Initialize candidate list; candidate_list = null
            For (sCandidate ∈ sNeighbourhood)
                If ( sCandidate ∉ TabuList )
                    Candidate_List = Candidate_list + sCandidate
                End
            End
        sCandidate =BestCandidate ( Candidate_list)
            If (fitness(sCandidtae) > fitness(sBest))
                Set  sBest = sCandidate
                Add its structures to the TabuList
                    While (size (TabuList) > maxTabusize)
                        Remove older structures from TabuList
                    End
            End
    End
Return (sbest)
```

Figure 3. Pseudo code of Tabu Search

https://dx.doi.org/10.4314/aujst.v7i1.4

Great Deluge Algorithm, and Flex-Deluge Algorithm, which introduce adaptive or flexible adjustments to the water level to balance exploration and exploitation (Khalfi et al., 2023; Aldeeb et al., 2019).

Today, GDA is valued for its simplicity, minimal parameter requirements, and ability to serve as a baseline or building block for more sophisticated hybrid methods. While population -based and probabilistic metaheuristics often overshadow it, GDA continues to play a role in optimization research, particularly where deterministic local improvement and low computational overhead are desired.

3. Discussion

In Simulated Annealing (SA), this balance is achieved through a temperature-controlled acceptance mechanism. At high temperatures, exploration dominates as even worse solutions may be accepted, while cooling gradually shifts the focus to exploitation. This dual role has enabled SA to remain competitive as a general-purpose optimizer, particularly when integrated into hybrid frameworks for deep learning and energy optimization (Liang et al., 2020; Wang & Li, 2022).

Start

Initialize population and parameters

Generate random solution: Sol

Set $SolutG = Sol$

Set level = $f(SolutG)$

Set estimated quality solution: $Estimated.Quality$

Set number of iteration: $IterNo$

Calculate increase rate $\beta = \frac{Estimated.Quality}{IterNo}$

Set Iteration $= 0$

While $(Iteration < IterNo)$

Generate a random solution sol_{new} in the neighborhood of Sol

Calculate $fSol_{new}$

If $f(Sol_{new}) > f(SolutG)$ then

$SolutG = Sol_{new}$

$Sol = Sol_{new}$

End If

If $f(Sol_{new}) > Level$ **then**

$Sol = Sol_{new}$

$f(Sol) = f(Sol_{new})$

End If

If $f(Sol_{new}) \leq Level$ **then**

$Level = Level + \beta$

End If

$Iteration = Iteration + 1$

End While

Return $SolutG$

Figure 4: Pseudocode of Great Deluge Algorithm

https://dx.doi.org/10.4314/aujst.v7i1.4

The enduring relevance of trajectory algorithms lies in their design around the twin principles of exploration and exploitation, which remain the foundation of metaheuristic search. Exploration enables an algorithm to sample widely across the solution space, reducing the risk of premature convergence to poor-quality solutions. Exploitation, on the other hand, allows the search to intensify around promising regions, refining candidate solutions to move closer to the global optimum. An effective balance between these two behaviors is essential: too much exploration leads to randomness and inefficiency, while excessive exploitation risks becoming trapped in local optima (Eiben & Smith, 2015).

Different trajectory algorithms implement this balance in distinct ways, and examining their mechanisms reveals both their strengths and limitations. Among them, Simulated Annealing (SA) stands out as a classical and widely applied method because it models this balance explicitly through its temperature-controlled acceptance mechanism. In its pseudocode in Figure 1, at line 10 a new neighbor state is generated, introducing diversity and enabling exploration of new regions. At line 12, if the new solution is better $(\Delta E < 0$), it is always accepted, representing exploitation through direct improvement.

However, lines 13–14 allow the acceptance of worse solutions with a probability dependent on the current temperature. When the temperature is high, this probability is large, making exploration dominant as the algorithm can escape local optima. As the algorithm advances, the cooling schedule at line 19 lowers the temperature, decreasing the acceptance of worse solutions and steadily shifting the search toward exploitation.

In theory, this mechanism creates smooth progression from global exploration in the early stages to local refinement in the later stages. In practice, however, the balance achieved by SA is sensitive to parameters such as the initial temperature (line 5), the number of iterations at each temperature (line 8), and the cooling rate (line 19). If these are poorly tuned, the algorithm may converge prematurely or waste excessive computation on random exploration. These sensitivities have motivated modern hybridizations of SA, particularly in fields such as deep learning and energy optimization, where robust exploration–exploitation trade-offs are critical.

In Hill Climbing, the balance between exploration and exploitation is far more limited than in Simulated Annealing because the algorithm follows a strictly greedy strategy. At line 3 of Figure 2, the process begins with a randomly generated initial state x_k, which ensures that the search starts from some arbitrary point in the solution space. The objective function of this candidate solution is then calculated (line 4). From this point forward, the search follows a very simple rule: at each iteration (line 6), an improved neighbor solution x' is generated from the neighborhood of the current state. If this neighbor improves upon the current solution (line 7), it is immediately accepted (line 8). Otherwise, the algorithm rejects it and continues searching.

This mechanism means that Hill Climbing performs pure exploitation, it always intensifies the search in the direction of improvement but does not allow for the acceptance of worse solutions. Unlike Simulated Annealing's acceptance rule (lines 13–14 in SA pseudocode), there is no probabilistic mechanism to promote exploration. As a result, once the algorithm reaches a local optimum, where no neighboring solution is better, it halts, even if better solutions exist elsewhere in the search space.

The strength of this approach lies in its simplicity and efficiency. Hill Climbing evaluates only one solution at a time and uses minimal computational resources. However, its inability to escape local optima highlights its weakness. In practice, this makes Hill Climbing useful not as a standalone optimizer but as a local search component within hybrid frameworks, where more exploration algorithms provide the global search ability and Hill Climbing serves to refine promising solutions.

In Tabu Search, exploration and exploitation are managed using adaptive memory rather than probabilistic acceptance rules. The process begins by selecting an initial solution (line 2) and initializing the Tabu List to empty (line 3). The Tabu List serves as a short-term memory of forbidden moves or solution structures, preventing the search from cycling back to recently visited states. A maximum size for this memory is also set at the outset (line 4).

https://dx.doi.org/10.4314/aujst.v7i1.4

During each iteration of the main loop (line 5), a list of candidate solutions is built from the neighborhood of the current solution. However, only those candidates that are not present in the Tabu List are considered (lines 7–9). This restriction forces the algorithm to explore new regions of the search space rather than repeatedly revisiting the same local configurations, thereby providing a structured mechanism for exploration.

From the filtered candidate list, the best candidate is selected (line 12). If its fitness is better than the best-so-far solution, the current best is updated (line 14), and the move or structure leading to that candidate is added to the Tabu List (line 15). This represents exploitation, since the algorithm intensifies the search by always adopting improving moves when they are available. To keep memory size manageable, older entries are removed whenever the Tabu List exceeds its maximum length (lines 16–17).

Through this cycle, Tabu Search balances the two metaheuristic pillars. Exploitation is achieved by greedily adopting the best candidate solutions, while exploration is enforced by forbidding recently used moves, pushing the search into less explored areas. The Tabu List thus provides a deterministic but adaptive exploration strategy, in contrast to the probabilistic approach of Simulated Annealing.

The strength of this design lies in its ability to escape local optima systematically, a weakness of Hill Climbing. Its main limitation, however, is sensitivity to the size of the Tabu List: if the list is too small, the algorithm risks cycling; if it is too large, it may forbid potentially useful moves and slow convergence. Nonetheless, this balance of memory-based exploration with greedy exploitation explains why Tabu Search remains one of the most successful trajectory algorithms for combinatorial optimization problems such as scheduling, routing, and logistics.

Furthermore, the Great Deluge Algorithm (GDA) approaches the balance between exploration and exploitation through the metaphor of a rising water level. The process begins by generating an initial random solution (line 3) and setting it as the best-so-far solution (line 4). The corresponding objective value defines the initial level (line 5), while an estimated target quality and the number of iterations is used to calculate the increase rate β (line 8). This β determines how quickly the water level rises during the search.

In each iteration (line 9), a new neighbor solution is generated, and its fitness is evaluated (line 10). If the new solution is strictly better than the best-so-far (lines 11–15), it is accepted and updates the current solution as well as the global best, which reflects exploitation. However, GDA also allows for the acceptance of solutions that are not necessarily better, as long as their fitness is greater than the current water level (lines 16–18). This introduces exploration, because it enables the algorithm to move into different regions of the search space rather than halting at the first local optimum. When the fitness of the candidate does not exceed the level (line 21), the level itself is incremented by β (line 22). This steady rise in the level reduces the likelihood of accepting worse solutions over time, shifting the algorithm toward exploitation as the search progresses. The iteration counter is then updated (line 24) until the maximum number of iterations is reached.

This mechanism allows GDA to mimic the exploration–exploitation transition seen in Simulated Annealing, but instead of relying on probability (as in SA's acceptance rule at lines 13–14), GDA uses a deterministic threshold. Early in the search, many solutions are admissible because the water level is low, fostering exploration. As the level rises, the acceptance window narrows, forcing the algorithm to exploit better-quality regions of the solution space.

The strength of GDA lies in its simplicity and low computational overhead, since it does not require complex probability calculations. However, its performance is highly sensitive to the choice of β; if the level rises too quickly, exploration is prematurely cut short, while a very slow rise can make the search inefficient. To address this, variants such as Extended GDA and Nonlinear GDA introduce adaptive level adjustments that attempt to fine-tune this balance. Today, GDA is used in scheduling, timetabling, and hybrid metaheuristic frameworks, where its deterministic acceptance rule complements more exploratory algorithms.

https://dx.doi.org/10.4314/aujst.v7i1.4

4. Conclusion

Despite renewed interest, trajectory algorithms face several enduring challenges. First, their susceptibility to local optima remains a limitation, especially in rugged fitness landscapes. Second, parameter sensitivity such as cooling schedules in SA or level increments in GDA, often requires expert tuning, reducing accessibility for non-specialists. Third, scalability remains an issue; while efficient for small and medium problems, trajectory algorithms often lag behind population-based methods in large-scale or highly multimodal domains.

Trajectory algorithms differ fundamentally from population-based methods such as Genetic Algorithms or Particle Swarm Optimization. The latter maintain diverse populations, naturally promoting exploration, but often at higher computational cost. Trajectory algorithms, in contrast, emphasize single-solution efficiency, which can be advantageous in resource-constrained environments. However, their single-solution focus limits their ability to escape highly multimodal landscapes unless they are hybridized with global exploration strategies.

References

Tartibu, L. K. (2025). *Multi-objective Optimization Techniques in Engineering Applications: Advanced Methods for Solving Complex Engineering Problems* (Vol. 1184). Springer Nature.

Darvishpoor, S., Darvishpour, A., Escarcega, M., & Hassanalian, M. (2023). Nature-inspired algorithms from oceans to space: A comprehensive review of heuristic and meta-heuristic optimization algorithms and their potential applications in drones. *Drones*, *7*(7), 427.

Lameesa, A., Hoque, M., Alam, M. S. B., Ahmed, S. F., & Gandomi, A. H. (2024). Role of metaheuristic algorithms in healthcare: a comprehensive investigation across clinical diagnosis, medical imaging, operations management, and public health. *Journal of computational design and engineering*, *11*(3), 223-247.

Wang, C. H., Hu, K., Wu, X., & Ou, Y. (2025). Rethinking Metaheuristics: Unveiling the Myth of "Novelty" in Metaheuristic Algorithms. *Mathematics*.

Dokeroglu, T., Canturk, D., & Kucukyilmaz, T. (2024). *A survey on pioneering metaheuristic algorithms between 2019 and 2024*. arXiv.

Almufti, S. M., Shaban, A. A., Ali, Z. A., Ali, R. I., & Fuente, J. D. (2023). Overview of metaheuristic algorithms. *Polaris Global Journal of Scholarly Research and Trends*, *2* (2), 10-32.

Selvarajan, S. (2024). A comprehensive study on modern optimization techniques for engineering applications. *Artificial Intelligence Review*, *57*(8), 194.

Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, *220*(4598), 671-680.Franzin, A., & Stützle, T. (2023). Simulated Annealing. *Introduction to Computational Intelligence*, 29.

Guilmeau, T., Chouzenoux, E., & Elvira, V. (2021). Simulated annealing: A review and a new scheme. In *2021 IEEE statistical signal processing workshop (SSP)* (pp. 101-105). IEEE.

Mahdi, W., Medjahed, S. A., & Ouali, M. (2017). Performance analysis of simulated annealing cooling schedules in the context of dense image matching. *Computación y Sistemas, 21*(3), 493–501.

Abdulhamid, S. I. M., Abd Latiff, M. S., Abdul-Salaam, G., & Hussain Madni, S. H. (2016). Secure scientific applications scheduling technique for cloud computing environment using global league championship algorithm. *PloS one*, *11*(7), e0158102.

Wang, X., & Li, J. (2022). Application of simulated annealing in renewable energy system optimization. *Energy Reports, 8,* 11234–11245.

Russell, S., & Norvig, P. (2016). *Artificial intelligence: A modern approach* (3rd ed.). Pearson.

Jian, J. R., Zhan, Z. H., & Zhang, J. (2020). Large-scale evolutionary optimization: A survey and experimental comparative study. *International Journal of Machine Learning and Cybernetics*, *11*(3), 729-745.

Palmer, R. (2018). Optimization on rugged landscapes. *Molecular Evolution on Rugged Landscapes*, 3-25.

https://dx.doi.org/10.4314/aujst.v7i1.4

Hanafi, S., Wang, Y., Glover, F., Yang, W., & Hennig, R. (2023). Tabu search exploiting local optimality in binary optimization. *European Journal of Operational Research*, *308*(3), 1037-1055.

Glover, F., Laguna, M., & Martí, R. (2018). Principles and strategies of tabu search. In *Handbook of Approximation Algorithms and Metaheuristics* (pp. 361-377). Chapman and Hall/CRC.

Lin, Y., Xu, J., & Sun, X. (2021). Tabu search-based metaheuristics for vehicle routing and logistics optimization: A review. *European Journal of Operational Research, 293*(2), 401–417.

Acan, A., & Ünveren, A. (2015). A two-stage memory powered Great Deluge algorithm for global optimization. *Soft Computing*, *19*(9), 2565-2585.

Arbaoui, B., Wahid, J., & Abdul-Rahman, S. (2022). A Modified Great Deluge Algorithm with Dual Decay Rate for School Timetabling. *International Journal of Intelligent Engineering & Systems*, *15*(6).

Khalfi, S., Caraffini, F., & Iacca, G. (2023). Metaheuristics in the Balance: A Survey on Memory-Saving Approaches for Platforms with Seriously Limited Resources. *International Journal of Intelligent Systems*, *2023*(1), 5708085.

Aldeeb, B. A., Al-Betar, M. A., Abdelmajeed, A. O., Younes, M. J., AlKenani, M., Alomoush, W., ... & Alqahtani, M. A. (2019). A comprehensive review of uncapacitated university examination timetabling problem. *International Journal of Applied Engineering Research*, *14*(24), 4524-4547.

https://dx.doi.org/10.4314/aujst.v7i1.4