Proceedings of the 4th International Conference on Applied Artificial Intelligence and Computing (ICAAIC-2025)
IEEE XPlore Part Number: CFP25BC3-ART; ISBN: 979-8-3315-6587-9

Introducing Party Competition: An Efficient High-Speed Metaheuristic for Solving the Binary Knapsack Problem

Yashar Salami, IEEE Professional Member
Faculty of Computer and Information Technologies, Cappadocia University, Ürgüp, Nevsehir, Turkey
yashar.salami@kapadokya.edu.tr

2025 4th International Conference on Applied Artificial Intelligence and Computing (ICAAIC) | 979-8-3315-6587-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICAAIC64647.2025.11330771

***Abstract*: With the growing demand for high-speed optimization algorithms, particularly for complex and time-sensitive problems, developing an effective high-speed metaheuristic approach presents a significant challenge in this field. This paper introduces a novel metaheuristic method aimed at enhancing both speed and efficiency in solving optimization problems, especially the binary knapsack problem. The simulation results reveal that the proposed algorithm significantly outperforms Genetic Algorithms (GA) and Imperialist Competitive Algorithms (ICA) in addressing the binary knapsack problem. Notably, the high convergence speed of this new method not only enhances its effectiveness but also establishes it as a highly efficient option for tackling complex and time-critical optimization challenges.**

Keywords—Party Competition, Meta-Heuristic, Optimization, Knapsack.

I. INTRODUCTION

With the rapid development of the Internet of Things and the Internet of Vehicles, enormous amounts of data are continuously exchanged among connected devices and vehicles [1][2]. To maintain secure and reliable communication, blockchain technology has become an effective solution for enhancing trust and transparency in these networks [3][4]. However, integrating blockchain with IoT and IoV introduces several optimization challenges related to processing speed, resource allocation, and energy efficiency, which require intelligent computational approaches to address effectively[5]. In the complex realm of optimization problems, finding optimal solutions for large and intricate issues often presents a significant challenge [6].

In the complex realm of optimization problems, finding optimal solutions for large and intricate issues often presents a significant challenge [7][8]. Metaheuristic algorithms are recognized as powerful and practical tools in this domain, especially for solving complex and large-scale problems. These algorithms draw inspiration from natural processes and complex systems to provide optimal or near-optimal solutions within reasonable time frames [9]. Unlike precise analytical methods that may require extensive computational time and become impractical for high-dimensional problems, metaheuristic algorithms leverage general search and optimization techniques to deliver efficient solutions in a reasonable amount of time [10]. The performance of metaheuristic algorithms depends significantly on their ability to generate high-quality results while reducing computational time[11]. These algorithms utilize search and optimization techniques for various problems, enabling them to provide acceptable results within reasonable timeframes[12]. Their unique characteristics have led to widespread use across diverse industries, including network design, production planning, resource management, and financial issues. Metaheuristic algorithms are specifically designed to tackle complex and large-scale problems, capable of delivering substantial improvements in system and process performance [13] [14].

Metaheuristic algorithms play a vital role in providing efficient and near-optimal solutions to complex problems, helping researchers and organizations achieve their objectives with limited resources. Their use improves system performance while reducing computational cost and time [15]. Many of these challenges fall under NP (Nondeterministic Polynomial time) problems, where finding an exact optimal solution is computationally expensive and often impractical within a reasonable timeframe [16][17].

Metaheuristic algorithms have been specifically developed to address issues related to NP problems[18]. These algorithms are generally designed to find near-optimal solutions within a reasonable time frame and can be effectively applied to NP problems [19]. By employing local and global search techniques, metaheuristic algorithms can provide optimal and efficient solutions over time, considering the unique characteristics of NP problems [20]. Several metaheuristic algorithms, including GA, are widely used to solve complex and NP-hard problems [21]. Inspired by natural evolution and selection principles, this algorithm employs selection, crossover, and mutation operators to generate near-optimal solutions [22]. Bee Algorithm: This algorithm explores various regions of the solution space to find optimal solutions, inspired by honeybees' foraging behavior [23]. Simulated Annealing (SA): This algorithm mimics physical annealing by adjusting the temperature and the probability of accepting worse solutions during the early stages of optimization [24]. Particle Swarm Optimization (PSO): Inspired by the behavior of bird flocks or fish schools, this algorithm explores the solution space by simultaneously evaluating multiple solutions [25]. ICA: Inspired by historical processes of competition and colonization, this algorithm

explores the solution space through a dynamic interaction among competing solutions, emulating the strategic efforts of nations vying for resources [26].

Metaheuristic algorithms are effective for global optimization but often suffer from low processing speed, which poses challenges in real-time applications and resource-constrained environments. Although methods such as GA, SA, and PSO can achieve high-precision solutions, their computational complexity and large search spaces often reduce efficiency and increase execution time.

In real-time processing, rapid response time and efficiency are critical even under limited computational resources. Algorithms capable of delivering results quickly while utilizing limited resources are highly needed. Standard metaheuristic algorithms may struggle to keep pace in such situations due to their lengthy processing times and complex computations. Therefore, developing and presenting a meta-heuristic method at a high speed is one of the most critical challenges in this field.

In this article, we introduce a serious metaheuristic method to address this challenge: party competition. Party competition is vital in democratic systems, where political parties compete for public support and influence public policies. This competition can occur either directly in public elections or indirectly by affecting legislative and political processes. The success of a party in political competition depends on its ability to attract and retain members, deliver effective programs, and communicate effectively with the public. This competitive process can inspire new metaheuristic approaches.

A. Paper Contribution

The primary contribution of this paper is the development of a novel metaheuristic algorithm inspired by human collective intelligence, designed to enhance speed and efficiency in solving complex optimization problems, particularly the binary knapsack problem. The proposed algorithm achieves faster convergence by leveraging principles of cooperation and intelligent behavior observed in human problem-solving. It provides more accurate solutions in a shorter time than GA and ICA. This algorithm, addressing the demand for rapid, effective solutions to time-sensitive problems, sets new benchmarks in high-speed optimization and highlights the value of collective intelligence in designing metaheuristic algorithms.

B. Paper organization

The rest of the paper, Section 2 Problem Statement, introduces the key issues and objectives of the research. Section 3 Methodology definition of the methodologies. Section 4This section details the experimental setup and presents the results of applying the algorithms. Finally, the Conclusion summarizes the findings.

II. Problem Statement

A. Binary Knapsack Problem

The Binary Knapsack Problem is a prominent and intricate optimization challenge. The goal is to select a subset of items such that the total weight of the selected items does not exceed the knapsack's capacity while maximizing their total value. Each item in this problem can be included in the knapsack or excluded, which makes it a binary (0/1) problem. Due to its NP-hard nature, finding the optimal solution for large-scale instances is computationally demanding and requires sophisticated techniques and specialized algorithms.

B. Definition of NP-hard Problems

NP-hard problems, a diverse class of computational problems, are inherently infeasible to solve optimally within polynomial time constraints. These problems typically require exponential time to solve, rendering traditional exact algorithms impractical due to their high computational complexity. As a result, addressing NP-hard problems effectively often involves using advanced optimization algorithms and metaheuristic techniques designed to produce near-optimal solutions within feasible time frames.

C. Binary Knapsack Problem and NP-hardness

The Binary Knapsack Problem is NP-hard, particularly when dealing with a large number of items and a substantial knapsack capacity. The exponential growth in complexity associated with finding the optimal solution makes exact approaches impractical for large instances. Thus, metaheuristic algorithms are widely employed to provide effective solutions. These algorithms are adept at exploring large and complex solution spaces. They can deliver near-optimal solutions efficiently, making them a valuable tool for addressing the challenges posed by NP-hard problems, such as the Binary Knapsack Problem.

III. Methodology

Political parties are fundamental organizations in democratic systems that unite individuals with shared ideologies, goals, and interests. These entities aim to influence public policies, governance, and social norms by pursuing specific programs and objectives that reflect their core principles. Parties serve as platforms for political participation, allowing members to engage in strategic activities, advocate for various causes, and compete in elections.

The primary functions of a political party include formulating policies, representing diverse segments of society, and providing a framework for political discussion and decision-making. Parties consist of various members, including leaders who guide the party's direction and lower-tier members who support its activities and goals. Each party operates based on its unique ideology, shaping its political and social orientation.

Party competition is a crucial process in democratic systems, in which political parties vie for public support, influence public policy, and achieve electoral success. This competition can occur directly in public elections or indirectly through legal and political processes. In these contests, parties

attract voters by presenting innovative programs, policies, and solutions.

The success of a party in political competition depends on various factors, including its ability to attract and retain members, offer effective and practical programs, and communicate effectively with the public. Successful parties can influence public policies, implement positive societal changes, and garner broad support from diverse groups. Success in party competition means winning elections, achieving long-term objectives, and implementing policies.

Party competition strengthens democracy and enhances political processes by offering diverse options and competing for public support, thereby advancing societal progress.

Inspired by this concept of party competition, we have developed a new metaheuristic method, which is described as follows:

A. Initial Population Definition

Population: A collection of "individuals," each representing a potential solution to the optimization problem. Each individual in the population has different values for decision variables.

Parties: The population is divided into several parties. Each party consists of a leader and several members. The party's leader is the individual with the lowest cost (or best performance) within that party.

B. Member Update (Attraction)

Movement Towards Leader: Each party member moves towards their party leader. This movement is regulated by an attraction coefficient (c1) and is randomly adjusted. The amount and direction of movement towards the leader are controlled to simulate the optimal positions.

Leader Update: If a member performs better than the party's current leader after moving, they are selected as the new leader. This process is conducted periodically and independently within each party.

C. Revolution (Jump)

Applying Jump: To explore new areas in the search space, some members of the population experience significant random changes in their positions. These changes are governed by a revolution coefficient (c2) and a probability of jumping, which prevent the search from becoming homogeneous and allow it to cover a broader space.

D. Intra-Party Competition

New Leader Selection: If a member performs better than the party leader, that member is assigned the leadership position. This process is performed independently within each party and regularly to ensure optimal leadership.

E. Inter-Party Competition

Member Absorption: Stronger parties can absorb members from weaker parties. Member absorption is based on the performance and merit of the party and its members, thereby enhancing the power and efficiency of stronger parties.

Disbanding Weak Parties: If a party fails to attract new members and does not improve its performance, it is disbanded, and its members are transferred to other parties. This process helps optimize the overall population structure.

F. Stopping Criteria

Criteria for Termination: The algorithm terminates when a specified stopping criterion is met, such as reaching a certain number of iterations or achieving a particular cost level. These criteria are defined to evaluate the algorithm's effectiveness and termination point. Figure 1 shows the pseudocode of the Party algorithm.

```
Initialize thepopulation with N individuals
Divide thepopulation into M parties
For each iteration:
  For each party:
    Update members by moving towards the leader
    Apply revolution to some members
    Perform intra-party competition
  Perform inter-party competition:
    Strong parties absorb members from weak parties
    Disband weak parties if necessary
  Check stopping criteria
Return the best solution found
```

Fig.1 . Pseudocode of the Party algorithm.

IV. SIMULATION

A. Environment simulation

The simulations were conducted on a Dell E6430 laptop equipped with an Intel Core i7 processor and 8GB of RAM, providing sufficient computational power for complex calculations. MATLAB 2013 was used as the simulation environment due to its robust numerical analysis and data processing capabilities, making it well-suited for this research.

B. Parameter settings

Table 1 details the parameter settings for the optimization algorithms used in this study. These parameters were consistently applied across all algorithms to ensure fair simulations and comparisons. The selected settings effectively evaluated the proposed method's performance against existing approaches, allowing for a clearer understanding of its relative strengths and weaknesses in optimization tasks.

Table 1 Parameter Settings.

Algorithm	Parameter	Value
Hundredth Monkey Effect	knowledge_threshold	10
	Threshold_value	1e-35
Genetic	Mutation	0.8
	Crossover	0.2
ICA	Revolution	0.3
	Assimilation	0.5
	alpha	1

C. Simulation result

The convergence behavior of the proposed Party algorithm, illustrated in Figure 2, shows that it rapidly approaches the optimal solution during the early iterations and reaches stability much faster than the other two algorithms. This fast convergence demonstrates the algorithm's strong exploitation ability and makes it highly suitable for applications that require quick responses and low execution time.

As shown in Figure 3, the Genetic Algorithm (GA) also successfully finds optimal solutions; however, its convergence rate is comparatively slower than that of the Party algorithm. While GA is effective in maintaining population diversity and achieving high accuracy, it generally requires more iterations and computational resources, making it more appropriate for problems where precision is prioritized over speed.

The convergence pattern of the Imperialist Competitive Algorithm (ICA), shown in Figure 4, indicates that it takes considerably longer to converge than both the Party and GA algorithms. Although ICA can effectively explore the search space, its slower convergence rate suggests it may struggle with large-scale or time-sensitive problems that demand rapid optimization.

Overall, the obtained results clearly demonstrate that the Party algorithm exhibits superior convergence behavior in solving the binary knapsack problem, significantly outperforming both the GA and ICA. Its ability to reach the optimal region rapidly with minimal computational cost highlights its strong balance between exploration and exploitation. Therefore, the Party algorithm can be regarded as a promising and efficient optimization approach, particularly suitable for real-time or large-scale applications where convergence speed and computational efficiency are critical.

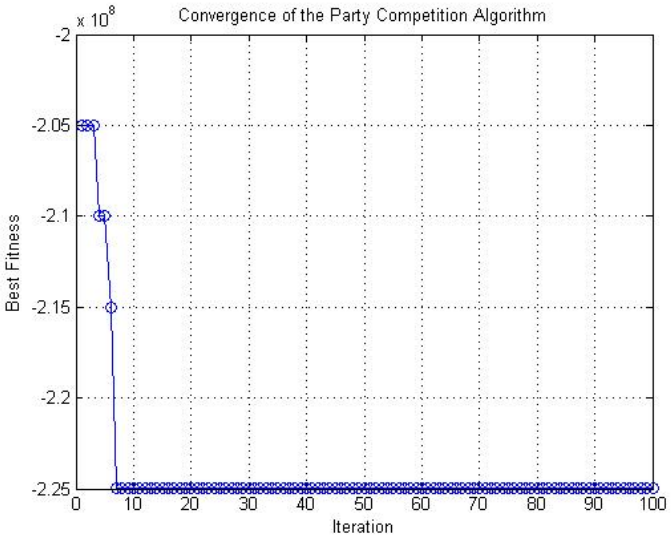

Fig. 2 Convergence of the party algorithm.

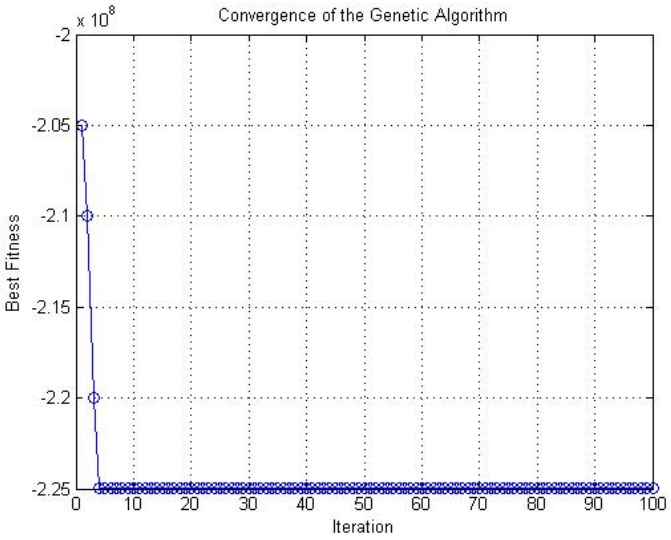

Fig. 3 Convergence of the GA algorithm.

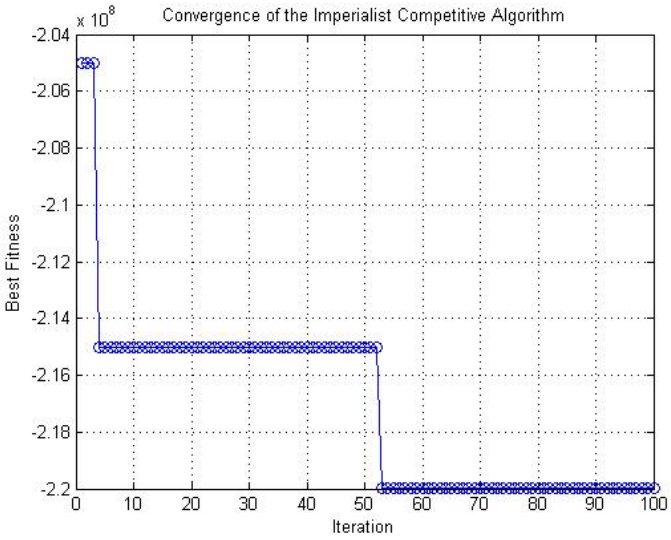

Fig. 4 Convergence of the ICA algorithm.

D. Discussion

The experimental results compare the execution time of three different algorithms used to solve the knapsack problem: the Party algorithm, the GA, and the ICA algorithm. The results, as presented in Table 2, reveal notable differences in Time execution across these Algorithms.

Table 2 Time execution.

Algorithm	Time execution
Party	0.497134 seconds
GA	135.179082 seconds
ICA	114.739302 seconds

The Party algorithm stands out with the fastest execution time of just 0.497134 seconds. This indicates a high efficiency level, making it particularly advantageous for optimization problems where quick solutions are essential. The Party algorithm's rapid performance suggests it could be preferred in scenarios that demand real-time processing or where computational resources are limited.

On the other hand, the GA exhibits the longest execution time, taking 135.179082 seconds to solve the problem. While GA is well-known for its ability to find optimal solutions in complex problem spaces, its relatively slow execution time may pose a limitation in time-sensitive applications. Despite its effectiveness, the requirement for significant computational time may limit its application in certain practical settings.

The ICA algorithm, with an execution time of 114.739302 seconds, offers a middle ground. Although slightly faster than GA, it still takes longer than the Party algorithm. ICA's performance suggests that, while it may be suitable for applications where precision is paramount, its slower execution speed compared to Party could be a drawback in scenarios where rapid decision-making is critical.

Overall, the results underscore the Party algorithm's superior speed, making it a compelling option for solving optimization problems where execution time is a crucial factor. In contrast, while GA and ICA offer robust optimization capabilities, their longer execution times may limit their practicality in situations where efficiency and speed are prioritized.

V. Conclusion

The analysis and experimental results demonstrate that the proposed metaheuristic algorithm significantly surpasses both GA and ICA in terms of convergence speed, computational efficiency, and overall performance for the binary knapsack problem. By achieving faster convergence and maintaining stable optimization behavior, the algorithm effectively addresses complex, large-scale problems with reduced execution time. Its strong balance between exploration and exploitation enhances solution quality without compromising speed. Therefore, the proposed algorithm stands out as a reliable and high-performance optimization technique, particularly well-suited for time-sensitive, real-time, and computationally demanding applications. These findings highlight its potential to establish a new benchmark for fast and efficient metaheuristic optimization.

References

[1] Y. Salami, "SO-ITS: a secure offloading scheme for intelligent transportation systems in federated fog-cloud," *Iran J. Comput. Sci.*, 2025, doi: 10.1007/s42044-025-00318-9.

[2] Y. Salami, "SOBT-UF: Secure Offloading in Blockchain Infrastructure for Intelligent Transportation Systems Using 5G-Enabled UAVs Within a Fog-Edge Computing Federation," in *2024 19th Iranian Conference on Intelligent Systems (ICIS)*, IEEE, 2024, pp. 217–222. doi: 10.1109/ICIS64839.2024.10887460.

[3] Y. Salami and S. Hosseini, "BSAMS: Blockchain-Based Secure Authentication Scheme in Meteorological Systems," *Nivar*, vol. 47, no. 120–121, pp. 181–197, 2023, doi: https://doi.org/10.30467/nivar.2023.415722.1260.

[4] Y. Salami, F. Taherkhani, Y. Ebazadeh, M. Nemati, V. Khajehvand, and E. Zeinali, "Blockchain-Based Internet of Vehicles in Green Smart City: Applications and Challenges and Solutions," *Anthropog. Pollut.*, vol. 7, no. 1, pp. 87–96, 2023, doi: 10.22034/AP.2023.1978624.1144.

[5] Y. Salami, V. Khajehvand, and E. Zeinali, "LSMAK-IOV: Lightweight Secure Mutual AKE Scheme in Fog-Based IoV," in *2024 10th International Conference on Artificial Intelligence and Robotics (QICAR)*, IEEE, 2024, pp. 1–5. doi: 10.1109/QICAR61538.2024.10496659.

[6] F. A. Quinton, P. A. S. Myhr, M. Barani, P. Crespo del Granado, and H. Zhang, "Quantum annealing applications, challenges and limitations for optimisation problems compared to classical solvers," *Sci. Rep.*, vol. 15, no. 1, p. 12733, 2025.

[7] S. M. Saranya, S. Mohanapriya, and D. Komarasamy, "Bio-inspired meta-heuristic algorithm for solving engineering optimization problems based on computational intelligence," in *Computational Intelligence in Sustainable Computing and Optimization*, Elsevier, 2025, pp. 259–280.

[8] H. Dahrouj *et al.*, "An overview of machine learning-based techniques for solving optimization problems in communications and signal processing," *IEEE Access*, vol. 9, pp. 74908–74938, 2021.

[9] E. H. Houssein, M. K. Saeed, G. Hu, and M. M. Al-Sayed, "Metaheuristics for Solving Global and Engineering Optimization Problems: Review, Applications, Open Issues and Challenges," *Arch. Comput. Methods Eng.*, pp. 1–35, 2024.

[10] B. Akay, D. Karaboga, and R. Akay, "A comprehensive survey on optimizing deep learning models by metaheuristics," *Artif. Intell. Rev.*, vol. 55, no. 2, pp. 829–894, 2022.

[11] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, and A. Cosar, "A survey on new generation metaheuristic algorithms," *Comput. Ind. Eng.*, vol. 137, p. 106040, 2019.

[12] S. Amaran, N. V Sahinidis, B. Sharda, and S. J. Bury, "Simulation optimization: a review of algorithms and applications," *Ann. Oper. Res.*, vol. 240, pp. 351–380, 2016.

[13] E.-G. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[14] E. Osaba *et al.*, "A tutorial on the design, experimentation and application of metaheuristic algorithms to real-world optimization

problems," *Swarm Evol. Comput.*, vol. 64, p. 100888, 2021.

[15] A. Rai, R. Patnayakuni, and N. Seth, "Firm performance impacts of digitally enabled supply chain integration capabilities," *MIS Q.*, pp. 225–246, 2006.

[16] H. Tang, X. Liu, C. Xiong, Z. Wang, and M. A. M. Ez Eldin, "Proof of nondeterministic polynomial-time complete problem for soil slope-stability evaluation," *Int. J. Geomech.*, vol. 16, no. 5, p. C4015004, 2016.

[17] B. Toaza and D. Esztergár-Kiss, "A review of metaheuristic algorithms for solving TSP-based scheduling optimization problems," *Appl. Soft Comput.*, p. 110908, 2023.

[18] Y. Salami, "Dialectic optimization algorithm (DOA): a novel metaheuristic inspired by dialectical philosophy," *J. Supercomput.*, vol. 81, no. 15, 2025, doi: 10.1007/s11227-025-07879-3.

[19] M. Baghel, S. Agrawal, and S. Silakari, "Survey of metaheuristic algorithms for combinatorial optimization," *Int. J. Comput. Appl.*, vol. 58, no. 19, 2012.

[20] G.-G. Wang, "Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems," *Memetic Comput.*, vol. 10, no. 2, pp. 151–164, 2018.

[21] V. Kesavan, R. Kamalakannan, R. Sudhakarapandian, and P. Sivakumar, "Heuristic and meta-heuristic algorithms for solving medium and large scale sized cellular manufacturing system NP-hard problems: A comprehensive review," *Mater. today Proc.*, vol. 21, pp. 66–72, 2020.

[22] A. Hassanat, K. Almohammadi, E. Alkafaween, E. Abunawas, A. Hammouri, and V. B. S. Prasath, "Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach," *Information*, vol. 10, no. 12, p. 390, 2019.

[23] D. T. Pham and M. Castellani, "The bees algorithm: modelling foraging behaviour to solve continuous optimization problems," *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.*, vol. 223, no. 12, pp. 2919–2938, 2009.

[24] D. Henderson, S. H. Jacobson, and A. W. Johnson, "The theory and practice of simulated annealing," *Handb. metaheuristics*, pp. 287–319, 2003.

[25] F. Pourpanah, R. Wang, C. P. Lim, X.-Z. Wang, and D. Yazdani, "A review of artificial fish swarm algorithms: Recent advances and applications," *Artif. Intell. Rev.*, vol. 56, no. 3, pp. 1867–1903, 2023.

[26] E. Atashpaz-Gargari and C. Lucas, "Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition," in *2007 IEEE congress on evolutionary computation*, Ieee, 2007, pp. 4661–4667.