A Novel Global Search Strategy Using Hyper-Heuristic Algorithms: A Comparative Analysis

Tse-Lin Li
Department of Computer Science
Fu Jen Catholic University
New Taipei, Taiwan
511172176@m365.fju.edu.tw

Abstract—Global optimization is a critical area in computational intelligence, dealing with the optimization of complex systems that are often non-linear, multi-modal, and high-dimensional. Traditional optimization algorithms sometimes fall short when handling such complexity due to premature convergence and inefficiency in large search spaces. This paper presents advanced global optimization strategies by integrating hyper-heuristic algorithms, specifically focusing on adaptive methods like Evolutionary Bald Eagle Search (EBES) and a multi-method hyper-heuristic framework. These strategies enhance the capability of solving complex optimization problems by dynamically selecting and adapting low-level heuristics. Comprehensive experiments on benchmark functions and real-world problems demonstrate significant improvements in convergence speed, solution accuracy, and robustness over traditional methods.

Index Terms—Global Optimization, Hyper-heuristics, Evolutionary Algorithms, Metaheuristics, Adaptive Search Strategies.

I. Introduction

Global optimization plays a pivotal role in solving complex problems across various domains, including engineering design, machine learning, financial modeling, and operational research [1]. These problems often involve objective functions that are non-linear, multi-modal, and high-dimensional, making them challenging to optimize due to the presence of numerous local optima [2]. Traditional optimization algorithms, such as Genetic Algorithms (GA) [3], Particle Swarm Optimization (PSO) [4], and Differential Evolution (DE) [5], have been widely employed because of their simplicity and general applicability. However, these methods often suffer from limitations like premature convergence to local optima and inefficiency in navigating high-dimensional search spaces [6], which restrict their effectiveness in more intricate problems.

Hyper-heuristics have emerged as a promising alternative, offering a high-level framework that adaptively selects or generates low-level heuristics to overcome these limitations [7]. Unlike traditional metaheuristics tailored to specific problem types, hyper-heuristics aim to provide a generalized solution approach applicable across various optimization scenarios. They operate on a search space of heuristics, allowing for greater flexibility and adaptability [8].

Recent advancements in hyper-heuristics, including Evolutionary Bald Eagle Search (EBES) and multi-method frameworks [9], have demonstrated significant improvements in balancing exploration and exploitation, enhancing global search capabilities. These methods integrate multiple heuristics or evolutionary strategies to adaptively adjust parameters and operators during the optimization process, leading to better performance in complex optimization landscapes.

This paper explores the efficacy of hyper-heuristic algorithms for global optimization, emphasizing their adaptability and efficiency in circumventing local optima. By integrating multiple heuristics, the proposed strategies aim to improve performance across diverse problem settings. We evaluate these methods using standard optimization test functions and real-world optimization problems, providing a comparative analysis of their strengths and limitations.

The rest of the paper is organized as follows: Section II provides background and related work on hyper-heuristics and evolutionary algorithms. Section III presents the proposed methodologies. Section IV details the experimental setup. Section V discusses the results and provides an in-depth analysis. Finally, Section VI concludes the paper and suggests future research directions.

II. Background and Related Work

A. Hyper-Heuristics

Hyper-heuristics represent a high-level search strategy that operates on a space of heuristics rather than directly on the solution space [7]. This abstraction allows hyper-heuristics to be more versatile and applicable to a wider range of problems. They are primarily categorized into two approaches:

1) Heuristic Selection-Based Hyper-Heuristics: These methods select the most appropriate low-level heuristic from a predefined set at each iteration based on certain criteria or performance metrics [10]. The selection can be deterministic or probabilistic and may involve strategies

such as roulette wheel selection, tournament selection, or adaptive selection mechanisms [11]. Learning mechanisms like reinforcement learning [12] or genetic programming [13] can also be employed to improve the selection process over time.

For instance, reinforcement learning-based hyper-heuristics adjust the selection probabilities of heuristics based on their past performance, allowing the algorithm to learn which heuristics are more effective in different situations [21].

2) Heuristic Generation-Based Hyper-Heuristics: Unlike selection-based methods, generation-based hyper-heuristics dynamically create new heuristics during the search process [14]. This can involve combining existing heuristics, modifying their parameters, or designing entirely new heuristic operators tailored to the problem at hand. Genetic programming is often used in this context to evolve new heuristics better suited for the specific characteristics of the problem.

These hyper-heuristics have been successfully applied in various domains, including scheduling [15], timetabling [16], and resource allocation [17]. Their ability to generalize across different problem types with minimal tuning makes them an attractive option for complex optimization tasks.

B. Evolutionary Algorithms for Global Optimization

Evolutionary algorithms, such as GA and PSO, have been extensively utilized for global optimization due to their ability to explore large search spaces and robustness in handling non-linear and multi-modal functions [18]. Despite their strengths, these algorithms are susceptible to premature convergence, where the population loses diversity and gets trapped in local optima [19].

To mitigate these issues, hybrid approaches like Evolutionary Bald Eagle Search (EBES) and Evolutionary PSO (EPSO) [20] have been developed. EBES integrates the Bald Eagle Search algorithm with evolutionary principles, allowing for real-time adjustment of search parameters through a genetic algorithm. This hybridization enhances the algorithm's ability to maintain diversity and adaptively balance exploration and exploitation, leading to improved convergence rates and solution quality.

C. Multi-Method Hyper-Heuristics

Multi-method hyper-heuristics take a hybrid approach by integrating multiple metaheuristic algorithms into a single optimization framework [9]. By leveraging the strengths of different heuristics—such as the exploratory capabilities of PSO, the adaptive mechanisms of GA, and the robust mutation strategies of DE—these frameworks can dynamically select the most suitable heuristic based on the current state of the search process.

This dynamic selection is often guided by online learning mechanisms that assess the performance of each heuristic in real-time, ensuring that the optimization process adapts to the problem's evolving landscape [21]. For example, a multi-armed bandit model can be used to balance the exploration of different heuristics with the exploitation of the best-performing ones, leading to more efficient search processes.

As a result, multi-method hyper-heuristics exhibit enhanced robustness, lower variance in solution quality, and superior performance across diverse and complex optimization problems compared to single-method algorithms.

III. Proposed Methodology

A. Evolutionary Bald Eagle Search (EBES)

EBES is a novel evolutionary hyper-heuristic inspired by the hunting behavior of bald eagles, which involves distinct phases of selection, local search, and swooping. The algorithm integrates the Bald Eagle Search (BES) with evolutionary principles to enhance its global search capabilities.

1) Algorithm Structure: The EBES algorithm operates in three main phases:

1) Selection Phase: Identifies promising regions in the search space by evaluating the fitness of current solutions. The best individuals are selected based on a fitness threshold or ranking mechanism. This phase ensures that only high-quality solutions are considered for further exploration.
2) Local Search Phase: Fine-tunes solutions within the selected regions to explore their immediate neighborhoods. This phase employs local search operators, such as mutation and crossover, to exploit the search space around high-quality solutions effectively. It aims to refine the solutions and move them closer to the global optimum.
3) Swooping Phase: Introduces exploratory moves to investigate new areas of the search space, preventing the algorithm from getting trapped in local optima. Random perturbations or global search operators are applied to diversify the population, enhancing exploration.

A genetic algorithm operates at a higher level to dynamically adjust the parameters of the BES in real-time. This includes adapting mutation rates, crossover probabilities, and other hyperparameters based on performance feedback from each iteration [6]. By maintaining a diverse population of BES agents, EBES ensures that multiple search strategies are employed concurrently, enhancing the algorithm's ability to navigate complex landscapes effectively.

2) Algorithm Implementation: The implementation of EBES involves initializing a population of candidate solutions and iteratively applying the three phases. The adaptation mechanisms monitor metrics such as diversity, convergence rate, and fitness improvement to adjust parameters dynamically. The algorithm ensures a balance between exploration and exploitation by continuously adapting its search strategies based on real-time feedback.

Algorithm 1 Evolutionary Bald Eagle Search (EBES)

```
1: Initialize population $P$ with $N$ individuals
2: Set initial parameters for mutation rate $\mu$ and crossover probability $c$
3: for each generation do
4:     Selection Phase:
5:     Evaluate fitness of each individual in $P$
6:     Select top $k$ individuals to form subset $S$
7:     Local Search Phase:
8:     Apply crossover and mutation operators on $S$ to create new offspring
9:     Evaluate fitness of new offspring
10:    Swooping Phase:
11:    Introduce random perturbations to a subset of $P$ to diversify the population
12:    Parameter Adaptation:
13:    Adjust $\mu$ and $c$ based on convergence metrics
14:    Update population $P$ with new solutions
15: end for
16: Return best solution found
```

B. Multi-Method Hyper-Heuristic Framework

The multi-method hyper-heuristic framework amalgamates several low-level heuristics—specifically variants of PSO, DE, and GA—into a cohesive optimization process [9]. The framework operates by evaluating the performance of each heuristic at different stages of the search and selecting the most appropriate one based on predefined criteria.

1) Key Components:

1) Heuristic Pool: Contains a diverse set of low-level heuristics, including PSO, DE, and GA variants. Each heuristic is parameterized differently to cover a wide range of search behaviors. For example, different PSO variants with varying inertia weights and acceleration coefficients are included to enhance exploration and exploitation.
2) Performance Evaluation: Continuously monitors the effectiveness of each heuristic using metrics such as convergence speed, solution quality, and diversity maintenance. Historical performance data is stored for decision-making. The framework employs statistical measures to assess the heuristics' performance over recent iterations.
3) Selection Mechanism: Utilizes an online learning algorithm, such as a multi-armed bandit model [21], to determine the best heuristic to apply at each iteration. The selection mechanism balances exploration of different heuristics and exploitation of the best-performing ones. This dynamic selection helps in adapting to different phases of the optimization process.

2) Framework Implementation: At each iteration, the framework selects a heuristic from the pool based on the selection mechanism and applies it to the current population. The performance of the heuristic is evaluated, and the selection probabilities are updated accordingly. This dynamic adaptation allows the framework to respond to changes in the problem landscape and maintain high performance.

The implementation involves maintaining a history of heuristic performances and using probabilistic models to select heuristics that are likely to perform well in the current context. This approach ensures that the framework can adaptively shift between exploration and exploitation as needed.

IV. Experimental Setup

To evaluate the effectiveness of the proposed hyper-heuristic strategies, we conducted experiments using a suite of benchmark optimization problems from the CEC 2017 competition and real-world optimization problems, such as engineering design optimization [22] and portfolio optimization [23]. These problems are characterized by their complexity, featuring multiple local optima and high-dimensional search spaces, making them ideal for testing global optimization algorithms.

A. Benchmark Functions

The benchmark suite includes 10 well-known functions, as summarized in Table I. These functions present various challenges, including non-convexity, multi-modality, separability, and high dimensionality.

TABLE I: Description of Benchmark Functions

Function	Dimensions	Characteristics
F1 (Sphere)	10	Unimodal, Convex
F2 (Rosenbrock)	30	Non-convex, Multi-modal
F3 (Rastrigin)	50	Highly Multi-modal
F4 (Ackley)	100	Multi-modal, Non-separable
F5 (Griewank)	10	Multi-modal, Periodic
F6 (Schwefel)	30	Complex, Many Local Optima
F7 (Levy)	50	Fractal-like Landscape
F8 (Alpine)	100	Non-convex, Rugged
F9 (Schwefel 2.22)	10	Non-differentiable
F10 (Expanded Schwefel)	100	Deceptive, Complex

Each function poses unique challenges, such as the deceptive valleys in the Rosenbrock function or the numerous local minima in the Rastrigin function. Testing on these functions ensures a comprehensive evaluation of the algorithms' capabilities.

B. Real-World Optimization Problems

To assess practical applicability, we included real-world optimization problems:

- Engineering Design Optimization: Optimization of a welded beam design to minimize cost while satisfying stress and deflection constraints [24]. This problem involves nonlinear constraints and requires balancing multiple objectives.

- Portfolio Optimization: Maximizing return while minimizing risk in financial portfolios, subject to investment constraints [23]. This problem is combinatorial in nature and involves handling real-world data and constraints.

These problems were selected because they represent common challenges in engineering and finance, providing a testbed for evaluating the algorithms in practical scenarios.

C. Experimental Environment

- Hardware: Intel Core i7 Processor, 3.60 GHz, 16 GB RAM
- Software: MATLAB R2021a
- Parameter Settings:
  - Population Size: 100
  - Number of Generations: 500
  - Mutation Rate (μ): Adapted dynamically for EBES, initial value set to 0.1
  - Crossover Probability (c): 0.8 for GA components
  - Selection Pressure: Implemented using tournament selection with size 5
  - Inertia Weight (for PSO components): Linearly decreasing from 0.9 to 0.4
  - Acceleration Coefficients (PSO): $c_1 = c_2 = 2.0$

These settings were chosen based on standard practices in the literature and preliminary experiments to ensure a fair comparison among the algorithms.

D. Evaluation Metrics

1) Average Relative Error (ARE): Measures the deviation of the obtained solution from the known global optimum or best-known solution, averaged over all runs.
2) Convergence Speed: Assessed by the number of generations required to reach a predefined solution accuracy (e.g., within 1% of the global optimum).
3) Computational Efficiency: Evaluated based on the total computational time taken to achieve convergence.
4) Robustness: Measured by the variance and standard deviation in solution quality across multiple runs, indicating consistency.
5) Success Rate: The percentage of runs where the algorithm converged to a solution within a predefined threshold of the global optimum.

Each algorithm—EBES, EPSO, BES, and PSO—was executed 30 independent runs for each benchmark and real-world problem to ensure statistical significance of the results.

V. Results and Discussion

A. Comparative Analysis of EBES and EPSO

Table II presents the detailed performance metrics for EBES, EPSO, BES, and PSO across the 10 benchmark functions and real-world problems.

1) Discussion:

- ARE Analysis: EBES achieved the lowest Average Relative Error of 0.73%, indicating its high accuracy in finding the global optimum across various test functions. This demonstrates the algorithm's effectiveness in diverse optimization landscapes.
- Standard Deviation: A lower standard deviation for EBES (0.05) indicates consistent performance across multiple runs, highlighting its robustness. The small variability suggests that EBES reliably produces high-quality solutions.
- Convergence Speed: EBES converged in 150 generations on average, significantly faster than EPSO and other algorithms, demonstrating its efficiency. Faster convergence is crucial in time-sensitive applications.
- Variance: The low variance in EBES suggests that the algorithm consistently finds high-quality solutions with minimal fluctuation between runs. This consistency is important for practical applications where reliability is essential.
- Computational Time: EBES required less computational time compared to BES and PSO, despite its complex adaptive mechanisms, due to faster convergence. Efficient use of computational resources makes EBES suitable for large-scale problems.
- Success Rate: EBES had a success rate of 96.7% in reaching the global optimum within a predefined threshold, outperforming other algorithms. A high success rate indicates the algorithm's reliability in finding optimal solutions.

These results indicate that EBES effectively balances exploration and exploitation, leading to superior performance in global optimization tasks. Its adaptive mechanisms allow it to adjust search strategies dynamically, improving both efficiency and effectiveness.

B. Effectiveness of Multi-Method Hyper-Heuristic

The multi-method hyper-heuristic framework was evaluated against single-method algorithms to assess its robustness and adaptability. Table III shows the detailed performance metrics.

1) Discussion:

- Solution Accuracy: The multi-method framework improved ARE by approximately 11% over EBES, indicating better optimization performance. This suggests that integrating multiple heuristics can enhance solution quality.
- Variance Reduction: A 36% reduction in variance suggests more consistent performance, enhancing reliability. Lower variance means that the algorithm is less sensitive to initial conditions.
- Convergence Speed: The framework converged 10 generations faster than EBES, showing enhanced efficiency in reaching optimal solutions. Faster convergence can reduce computational costs in practical applications.

TABLE II: Detailed Performance Comparison Across Algorithms

Algorithm	ARE (%)	Std. Dev.	Convergence Speed (Gen.)	Variance	Time (s)	Best Solution Found	Success Rate (%)
EBES	0.73	0.05	150	2.5×10^{-3}	120.5	Optimal	96.7
EPSO	2.59	0.15	250	2.25×10^{-2}	150.3	Near-optimal	83.3
BES	4.73	0.30	350	9.0×10^{-2}	200.7	Sub-optimal	70.0
PSO	28.75	1.50	500	2.25	250.9	Poor	50.0

TABLE III: Multi-Method Hyper-Heuristic Detailed Performance

Metric	Multi-Method	EBES	Improvement	Time (s)	Success Rate (%)
Solution Accuracy (ARE %)	0.65	0.73	10.96%	130.0	98.3
Variance in Solution Quality	1.6×10^{-3}	2.5×10^{-3}	36.00%	-	-
Convergence Speed (Generations)	140	150	6.67%	-	-
Computational Efficiency (Time/s)	130.0	120.5	-7.85%	130.0	-

- Success Rate: An increased success rate of 98.3% demonstrates the framework's robustness across different problem instances. High success rates are critical for applications where failure to find an optimal solution is costly.
- Computational Time: Although there is a slight increase in computational time compared to EBES, the performance gains justify the overhead. The trade-off between computational time and solution quality is acceptable in many real-world scenarios.

The multi-method hyper-heuristic framework leverages the strengths of various heuristics, adapting dynamically to the problem landscape, which contributes to its superior performance. By selecting the most appropriate heuristic at each iteration, the framework can navigate complex optimization landscapes more effectively.

C. Statistical Analysis

A paired t-test was conducted to evaluate the significance of the performance improvements. The results showed that both EBES and the multi-method hyper-heuristic framework provided statistically significant enhancements over traditional algorithms, with p-values less than 0.01 for most benchmark functions and real-world problems. This statistical significance confirms that the observed improvements are not due to random chance.

D. Computational Efficiency

While EBES and the multi-method framework demonstrated superior performance, they required additional computational resources due to the complexity of their adaptive mechanisms. However, the increase in computational time is offset by the substantial improvements in solution quality, convergence speed, and robustness. For many practical applications, especially those where solution quality is paramount, this trade-off is acceptable.

Moreover, the multi-method framework's ability to dynamically adapt its heuristic selection can lead to overall time savings in large-scale or complex problems by reducing the number of generations needed for convergence.

E. Limitations and Future Work

1) Limitations:

- Computational Overhead: The adaptive mechanisms introduce computational overhead, which may be significant for very large-scale problems or real-time applications. Optimizing the algorithms for computational efficiency remains a challenge.
- Parameter Sensitivity: Although the algorithms adapt parameters dynamically, initial parameter settings can influence performance during the early stages of the search. Sensitivity analysis is needed to understand the impact of initial settings.
- Scalability: The current implementations may face scalability issues when applied to problems with extremely high dimensionality or in distributed computing environments. Further research is needed to enhance scalability.

2) Future Work:

- Integration with Machine Learning: Incorporating machine learning techniques to predict and select the most promising heuristics more effectively, potentially using reinforcement learning or neural networks. This could enhance the adaptability of the hyper-heuristic framework.
- Expanded Heuristic Pool: Including additional heuristics, such as Artificial Bee Colony (ABC) [25], Firefly Algorithm [26], and Ant Colony Optimization (ACO) [27], to enhance diversity and adaptability. A larger heuristic pool may improve performance on a wider range of problems.
- Real-Time Applications: Applying the methods to real-time optimization problems in robotics, autonomous systems, and dynamic environments where the optimization landscape changes over time. This

would test the algorithms' adaptability in changing conditions [28].
- Parallelization: Implementing the algorithms in parallel computing frameworks or using GPU acceleration to improve computational efficiency and scalability. Parallelization could mitigate computational overhead issues [29].
- Benchmarking on Diverse Problems: Testing the algorithms on a broader range of real-world problems, including those from bioinformatics [30], logistics , and energy management, to assess generalizability. This would provide insights into the algorithms' applicability across different domains.

VI. Conclusion

This paper presented a comprehensive analysis of hyper-heuristic strategies for global optimization, introducing the Evolutionary Bald Eagle Search (EBES) and a multi-method hyper-heuristic framework. Both approaches exhibited significant performance improvements over traditional optimization techniques, particularly in terms of convergence speed, solution accuracy, and robustness. The adaptive mechanisms inherent in hyper-heuristics enable these algorithms to effectively navigate complex, high-dimensional search spaces and avoid local optima traps.

The experimental results demonstrate that EBES and the multi-method framework are effective tools for solving complex optimization problems. Their ability to adaptively adjust search strategies based on real-time performance metrics allows them to outperform traditional methods.

Future research will explore the integration of advanced machine learning algorithms to further enhance the adaptability and predictive capabilities of hyper-heuristic methods. Additionally, applying these strategies to a broader range of real-world optimization problems will help in assessing their scalability and practical utility across various industries.

References

[1] C. A. Floudas and P. M. Pardalos, Encyclopedia of Optimization. Springer Science & Business Media, 2008.

[2] J. D. Pintér, Global Optimization in Action: Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications. Springer Science & Business Media, 1996.

[3] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989.

[4] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948.

[5] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," Journal of Global Optimization, vol. 11, no. 4, pp. 341–359, 1997.

[6] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," IEEE Transactions on Evolutionary Computation, vol. 3, no. 2, pp. 124–141, 1999.

[7] E. K. Burke, M. Gendreau, M. Hyde, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: a survey of the state of the art," Journal of the Operational Research Society, vol. 64, no. 12, pp. 1695–1724, 2013.

[8] K. Chakhlevitch and P. M. Cowling, "Hyperheuristics: recent developments," in Eds., Springer Studies in Computational Intelligence. Vol. 136, pp. 3-29.

[9] J. Grobler, A. P. Engelbrecht, and M. Clarke, "Alternative hyper-heuristic strategies for multi-method global optimization," in 2010 IEEE Congress on Evolutionary Computation, 2010, pp. 1–8, doi: 10.1109/CEC.2010.5585980.

[10] E. K. Burke, M. Hyde, G. Kendall, and J. Woodward, "A classification of hyper-heuristic approaches," in Handbook of Metaheuristics. Springer, 2010, pp. 449–468.

[11] E. K. Burke, "A survey of hyper-heuristics," Computer Science Technical Report, NOTTCS-TR-SUB-0906241418-2747 , 2009.

[12] A. Nareyek, "Choosing search heuristics by non-stationary reinforcement learning," in Metaheuristics: Computer Decision-Making. Springer, 2003, pp. 523–544.

[13] E. K. Burke, M. Hyde, G. Kendall, and J. Woodward, "Hyflex: A flexible framework for the design and analysis of hyper-heuristics," in Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA 09), 2009, pp. 790-797.

[14] E. K. Burke, M. Hyde, G. Kendall, and J. Woodward, "The scalability of evolved on line bin packing heuristics," in Computational Intelligence, IEEE, 2007, pp. 2530-2537.

[15] B. Bilgin, E. Özcan, and E. K. Burke, "An experimental study on hyper-heuristics and exam timetabling," in Practice and Theory of Automated Timetabling VI. Springer, 2007, pp. 394–412.

[16] S. Luke, Essentials of Metaheuristics. Lulu, 2011.

[17] D. Pisinger and S. Ropke, "A general heuristic for vehicle routing problems," Computers & Operations Research, vol. 34, no. 8, pp. 2403–2435, 2007.

[18] A. E. Eiben and J. E. Smith, Introduction to Evolutionary Computing. Springer, 2003.

[19] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies—a comprehensive introduction," Natural Computing, vol. 1, no. 1, pp. 3–52, 2002.

[20] V. Miranda and N. Fonseca, "EPSO—evolutionary particle swarm optimization, a new algorithm with applications in power systems," in IEEE/PES Transmission and Distribution Conference and Exhibition, vol. 2, 2002, pp. 745–750.

[21] A. Fialho, L. M. Costa, M. Schoenauer, and M. Sebag, "Analyzing bandit-based adaptive operator selection mechanisms," Annals of Mathematics and Artificial Intelligence, vol. 60, no. 1, pp. 25–64, 2010.

[22] X.-S. Yang, Engineering Optimization: An Introduction with Metaheuristic Applications. John Wiley & Sons, 2010.

[23] T.-J. Chang, N. Meade, J. Beasley, and Y. M. Sharaiha, "Heuristics for cardinality constrained portfolio optimisation," Computers & Operations Research, vol. 27, no. 13, pp. 1271–1302, 2000.

[24] C. A. C. Coello, "Use of a self-adaptive penalty approach for engineering optimization problems," Computers in Industry, vol. 41, no. 2, pp. 113–127, 2000.

[25] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," Journal of Global Optimization, vol. 39, no. 3, pp. 459–471, 2007.

[26] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," International Journal of Bio-Inspired Computation, vol. 2, no. 2, pp. 78–84, 2010.

[27] M. Dorigo and T. Stützle, Ant Colony Optimization. MIT Press, 2004.

[28] F. Stulp and O. Sigaud, "Policy improvement methods: Between black-box optimization and episodic reinforcement learning," arXiv preprint arXiv:1206.4621, 2012.

[29] M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo, "Distributed algorithms for environment partitioning in mobile robotic networks," IEEE Transactions on Automatic Control, vol. 56, no. 8, pp. 1834–1848, 2011.

[30] C. Blum and X. Li, "Swarm intelligence in optimization," in Swarm Intelligence. Springer, 2011, pp. 43–85.