

Final project report

For Soft Computing Class

Improved PSO-X: Enabling PSO-X with GA and grandparenting technique

Kamran Nazari, Hojjat Hosseini,

Supervisor:Prof. M.R.Akbarzadeh

faculty of Engineering, The University of Ferdowsi, Mashhad, Summer 2023

|  |  |  |
| --- | --- | --- |
| A R T I C L E I N F O |  | A B S T R A C T |
| *Keywords:*  PSO-X  Particle Swarm Optimization  Flexible framework  Automatic configuration  Continuous optimization problems  Intelligent systems  Complexity  Uncertainty  Genetic Algorithm (GA)  Grand parenting  Performance metrics |  | The main article introduces PSO-X, a flexible framework for developing high-performing Particle Swarm Optimization (PSO) algorithms. PSO-X combines algorithm components and automatic configuration tools to efficiently tackle continuous optimization problems in intelligent systems. The framework addresses complexity, uncertainty, optimization, and learning. The article mentions the use of "IRACE," which stand for Iterated Racing for Automatic Algorithm Configuration is an automatic configuration strategy, to find the best combination of components and parameter values for each PSO variant. PSO-X is evaluated by automatically creating six PSO algorithms and comparing their performance with existing variants. The analysis demonstrates that the automatically created PSO-X algorithms outperform their manually created counterparts. Our suggestion involves the utilization of a genetic algorithm (GA) in conjunction with the grand parenting technique to reach a synergistic combination. instead of IRACE for improved performance. GA's strengths include exploring a larger solution space, handling complex problems, and adapting to changing environments. In the grand parenting method, we introduce significant alterations to the majority of characteristics found in the previous best answer, while retaining only a select few. This approach enables the generation of new individuals that bear a slight resemblance to the previous best answer. |

**INTRODUCTION**

Computational intelligence algorithms have become increasingly popular for addressing complex optimization problems, where conventional methods may not be practical. Particle swarm optimization (PSO) and evolutionary algorithms (EAs) are two widely used techniques in this domain [1], [2]. These algorithms have shown significant success in various fields, where resource efficiency, automation, and informed decision-making are crucial requirements. Despite their impressive application potential, the development of computational intelligence algorithms has largely remained a manual and intuitive process, relying on the expertise of developers [3]

Manual development of such algorithms presents several challenges, including time-consuming trial and error, limited exploration of design alternatives, and difficulty in reproducing the development process. To address these issues, a promising approach has emerged, based on the concept of components [5], [6]. This component-based framework incorporates automatic configuration tools, enabling the creation of high-performing algorithms. Unlike traditional manual approaches, where algorithms are considered as monolithic entities with a few numerical parameters modified based on the developer's experience, component-based approaches view algorithms as combinations of individual algorithm components. This approach relies on three key elements: 1) an evolutionary method framework providing a selection of algorithm components, 2) a set of rules governing the coherent combination of components, and 3) an automatic configuration tool to evaluate various designs and parameter settings.

While considerable work has been done on automatic design using widely-used algorithms such as ant colony optimization and artificial bee colony, the automatic design of PSO algorithms has received comparatively less attention [5], [7]. Existing works in this area, such as [8] and [9], have some limitations in the number of components that can be combined automatically.

In this article, we introduce PSO-X, a flexible and component-based framework for automatic design of PSO algorithms. PSO-X incorporates a comprehensive collection of algorithm components proposed in the PSO literature, and each component can assume various values, allowing PSO-X to generate specific PSO algorithms. A generalized PSO template forms the core of PSO-X, offering the flexibility to combine algorithm components in diverse ways, enabling the synthesis of numerous well-known PSO variants published in the last two decades. The use of a generalized velocity update rule (GVUR) facilitates the abstraction of key algorithm elements, fostering interactions between high-level components and specific strategies, which contributes to PSO-X's versatility.

The advantages of PSO-X include the ease of creating diverse implementations by combining a wide array of components within a single framework and the ability to use automatic configuration tools to tailor PSO implementations to specific problem scenarios. This article aims to demonstrate the efficiency and superior performance of PSO algorithms developed using PSO-X compared to manually designed counterparts.

Figure 1 – Road map for the suggested optimization procedure

New Configuration Setup

Configuration Parameters

Positions

Function Evaluation (F.E) and *f\_counter*

Best F.E for specific configuration parameters

Best F.E for given configuration

Choose Configuration with any “Grand Parenting” algorithm

Choose Configuration parameter for specific setup

Setup a PSO config with specified configuration and its parameters

Benchmark Function

Grand-Parenting (G.P)

GA

PSO

**Research Explanation**

This mega optimization problem consists of three primary optimization steps.

In the first step, we employ the grand parenting technique to determine the type of component to be used. This technique aids in identifying the most suitable component for the problem at hand.

Moving on to the second step, we utilize a Genetic Algorithm (GA) to find the optimal parameters for each component. The GA methodology allows us to explore the parameter space and select the most favorable combination of values for achieving the desired optimization outcome.

Finally, in the third step, we employ Particle Swarm Optimization (PSO) using the specific component and parameter configurations defined in the previous steps. The PSO algorithm aims to search for the best solution to the benchmark function, leveraging the chosen component and its corresponding parameters.

By following this structured approach across these three optimization steps, we can systematically address the complexity of the mega optimization problem and work towards achieving the most optimal solution.

**Grand-Parenting (G.P)**

In the initial stage, we are presented with a complex challenge that involves 14 components. Our goal is to determine the optimal combination for this particular optimization problem. To tackle this, we adopt a population-based optimization approach.

At the outset, we assign each particle a random combination of components within our defined search space. Subsequently, we evaluate the performance of these combinations using the second function and identify the particle with the best function evaluation value.

As we proceed to the next iteration, we create a new population based on the best particle obtained from the previous iteration. This process entails modifying the combination of the new particle in 13 of the components, while retaining one component value from the previous best particle.

By iteratively refining the combinations and evaluating their function values, we aim to converge towards the optimal solution for our optimization problem. This population-based approach allows us to explore and exploit different combinations of components, gradually improving the overall performance of our optimization process.

Figure 2 – Grand parenting technique explanation

Best of First Generation

Initial Generation

First Generation

Second Generation

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Best of Initial Generation

**GA-Library**

After identifying the desired components, the subsequent step involves selecting suitable parameters for each component. To accomplish this, we utilize a Genetic Algorithm (GA) during the upper stage optimization process, once the component type has been determined. In order to represent the characteristics and required parameters accurately, we employ a chromosome structure similar to that of a book. Within this structure, each word corresponds to a bit, while each sentence represents a parameter for a specific component. Additionally, each page within the "book" contains the parameters pertaining to a particular component.

Considering the possibility of running the GA with multiple populations and multiple iterations, it may be beneficial to envision this process as constructing a library. Each population can be seen as a distinct section within the library, and the iterations as the continuous expansion and improvement of the library's contents. By employing this analogy, we emphasize the importance of organization, diversity, and the accumulation of knowledge throughout the optimization process.

Figure 3 – The concept which is used in building chromosome structure

WORDS MAKE SENTENCE

WORD

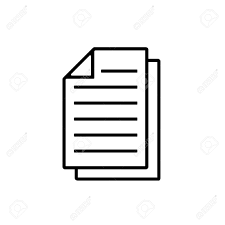
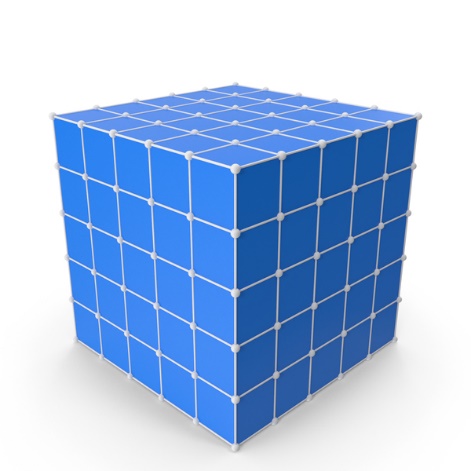


Figure 4 – Applying crossover operation on chromosome structure



Components

Chromosome

Parameters

Crossover Plain

**GA operations**

The GA operation consists of three key steps: selection, crossover, and mutation.

In the selection operation, we utilize the rank selection method, which ensures that even the worst-performing particles have a chance of being chosen. This approach promotes diversity and prevents premature convergence.

Moving on to the crossover operation, if permitted, we determine a random percentage and adjust the length of the chromosomes accordingly. For example, if a chromosome parameter consists of 10 bits and the generated random percentage is 40%, we select 4 bits from the first selected chromosome and 6 bits from the second selected chromosome. The reverse process is applied to the second individual. This strategy accommodates the variability in chromosome lengths, enabling effective genetic recombination.

The final step, mutation, involves generating a random number for each bit. If the generated value is less than the mutation probability, we flip the corresponding chosen bit. Mutation introduces random changes into the population, promoting exploration and preventing the algorithm from getting stuck in local optima.

By combining these three operations, selection, crossover, and mutation, the GA optimizes the population over multiple generations, gradually improving the quality of solutions and converging towards the optimal solution for the given problem

**PSO-Component**

In the last step component can be categorized into five groups, each serving a specific purpose.

Group 1 focuses on main algorithm parameters, such as ω, ϕ1, and ϕ2, which heavily influence the exploration and exploitation behavior of the algorithm. Strategies for adjusting these parameters are abundant in the PSO literature, particularly ω, which is crucial for local convergence. Various approaches, including time-varying and adaptive strategies, address issues like swarm explosion and poor scalability in high-dimensional spaces.

Group 2 comprises components related to controlling the distribution of particles' next possible positions (DNPPs). Different mappings, such as rectangular, spherical, and additive stochastic, determine how particles move between positions. Solutions to transformation variance, which occurs when the algorithm performs poorly under mathematical transformations of the objective function, have been developed.

Group 3 consists of components allowing perturbations to be applied to particles' velocity and/or position vectors. Perturbation mechanisms in PSO can be informed or random, aiming to improve diversity, prevent stagnation, and avoid divergence. They can modify both vectors and the DNPP of particles, enhancing exploration capabilities.

Group 4 focuses on random matrices, providing diversity to particle movement. They differ from perturbation mechanisms by allowing changes in the magnitude and direction of the CI and SI vectors. Random diagonal matrices and random rotation matrices (RRMs) address transformation variance and impact algorithm performance.

Group 5 encompasses components related to topology, model of influence, and population size. Topology influences exploration-exploitation capabilities, with various options explored in the PSO literature. The model of influence determines how informants contribute, with options like best-of-neighborhood, fully informed, and ranked fully informed models. Population size affects the tradeoff between solution quality and algorithm speed, and dynamic adjustments have been proposed.

Figure 5 - Example of component and its sub-components and parameters

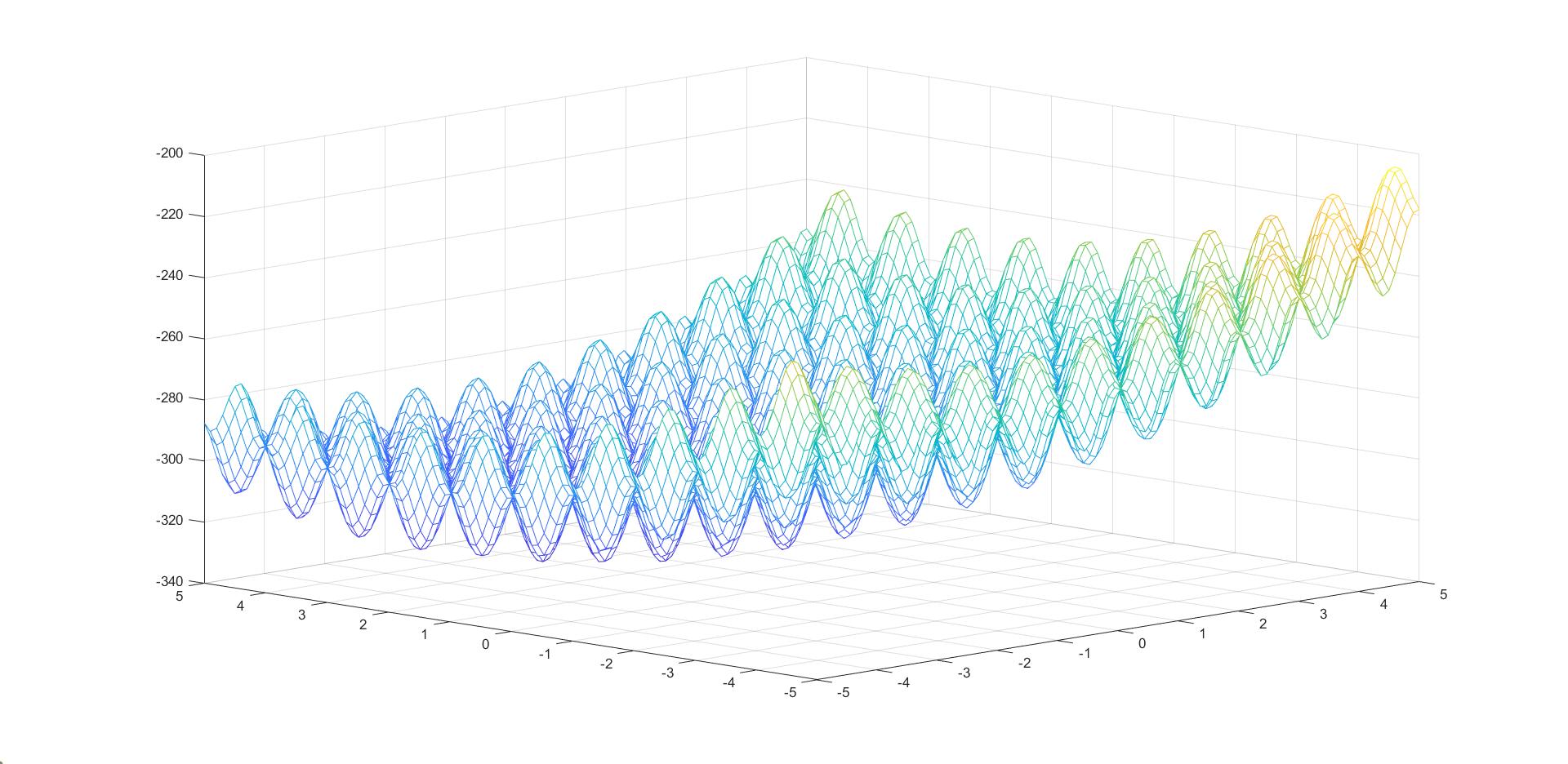


Figure 6 – Shifted Rastrigin's benchmark function

**EXPERIMENTAL PROCEDURE & ANALYSIS OF THE RESULTS**

To evaluate our optimization setup, we conducted tests using the Shifted Rastrigin's benchmark function. This function poses a significant challenge due to its numerous local minima, which can trap simple optimization algorithms. Successfully finding the global minimum requires a synergistic combination of exploration and exploitation elements.

The Shifted Rastrigin's function is known for its multimodal nature, with many local optima distributed throughout the search space. It presents a complex landscape that can deceive traditional optimization algorithms, leading to suboptimal solutions.

In order to manage the computational time required for the optimization procedure, we have set specific iterations for each step of our optimization setup.

For the first step, we have allocated 10 iterations. This initial phase involves determining the component type and is critical for the subsequent stages. By limiting the iterations in this step, we aim to expedite the process while still obtaining satisfactory results.

Similarly, for the second step, we have allocated another 10 iterations. This phase involves finding the optimal parameters for each component. By keeping the iteration count low, we strike a balance between obtaining accurate parameter values and minimizing the computational time required.

For the final step, which involves utilizing Particle Swarm Optimization (PSO) with the defined components and parameters, we have allocated a larger number of iterations, specifically 100. This step is crucial for refining the solution and converging towards the best possible outcome. By allocating more iterations in this stage, we allow sufficient time for the PSO algorithm to explore the search space and optimize the objective function effectively.

By setting lower iterations for the initial steps and a relatively higher number of iterations for the final step, we aim to strike a balance between computational time and achieving a satisfactory solution. This approach allows us to optimize the process efficiently while still obtaining reliable results within a reasonable timeframe.

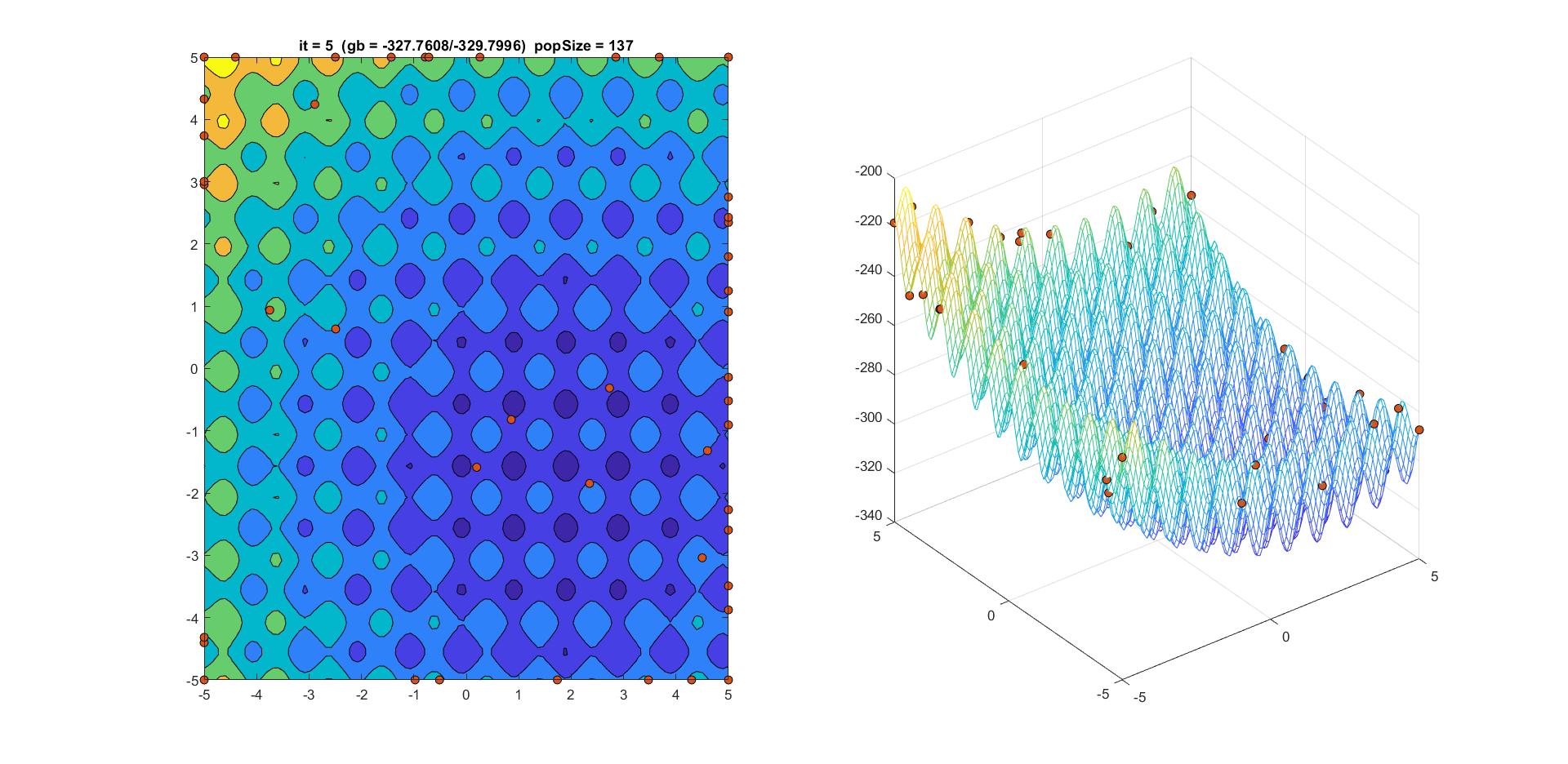


Figure 7 – Iteration 5 for A configuration for PSO

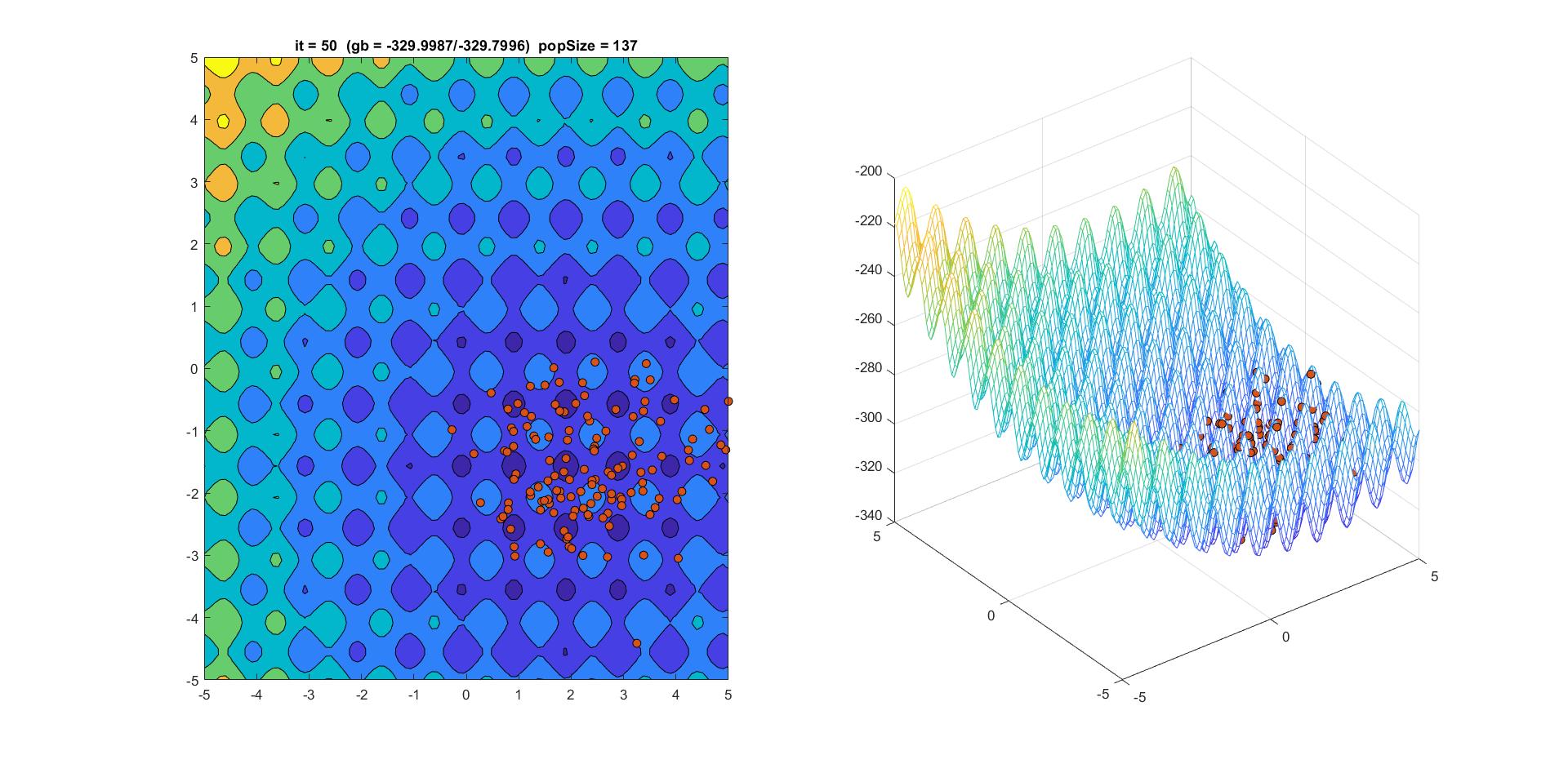


Figure 8 – Iteration 50 for A configuration for PSO

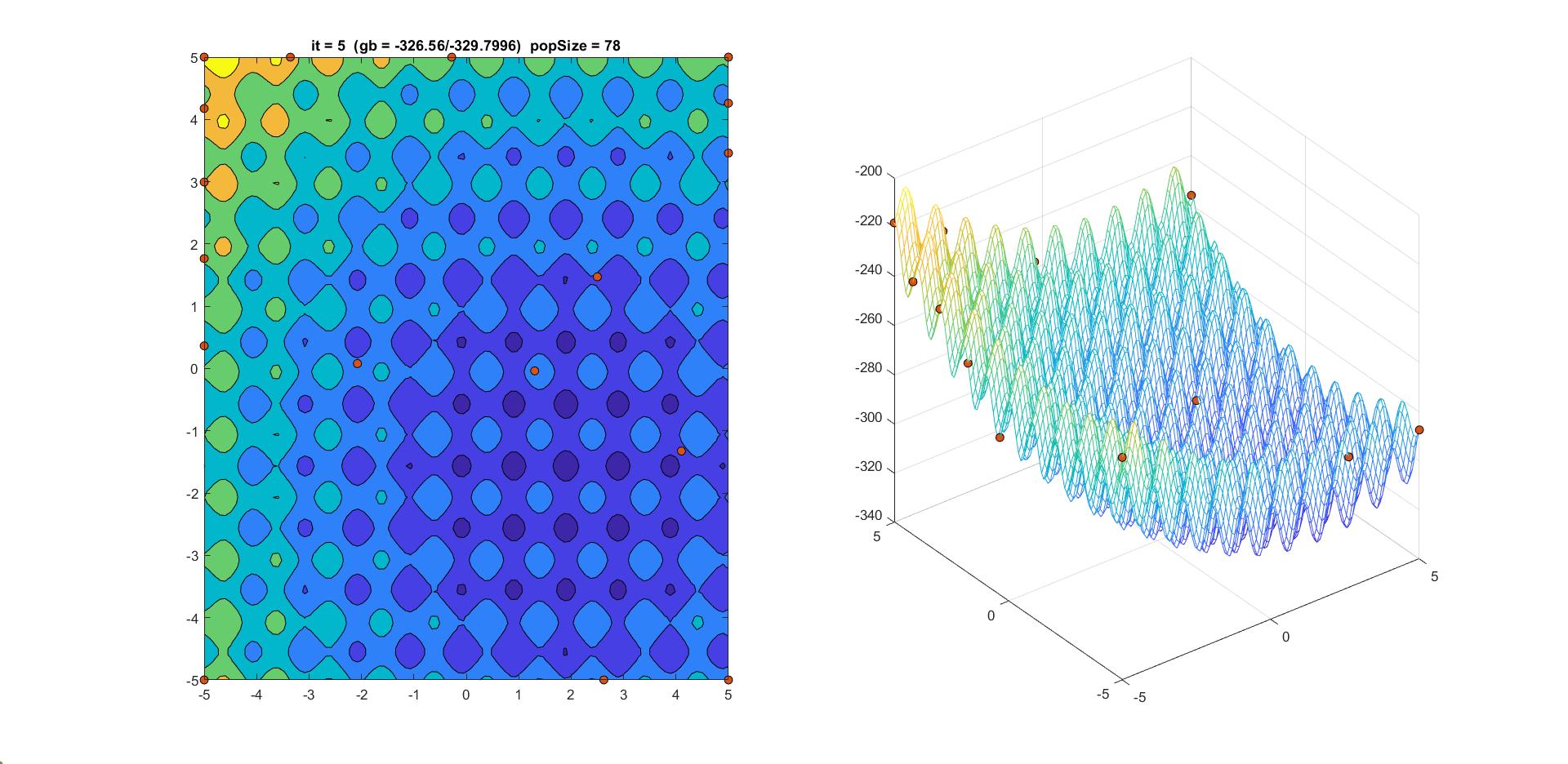


Figure 9 – Iteration 5 for B configuration for PSO

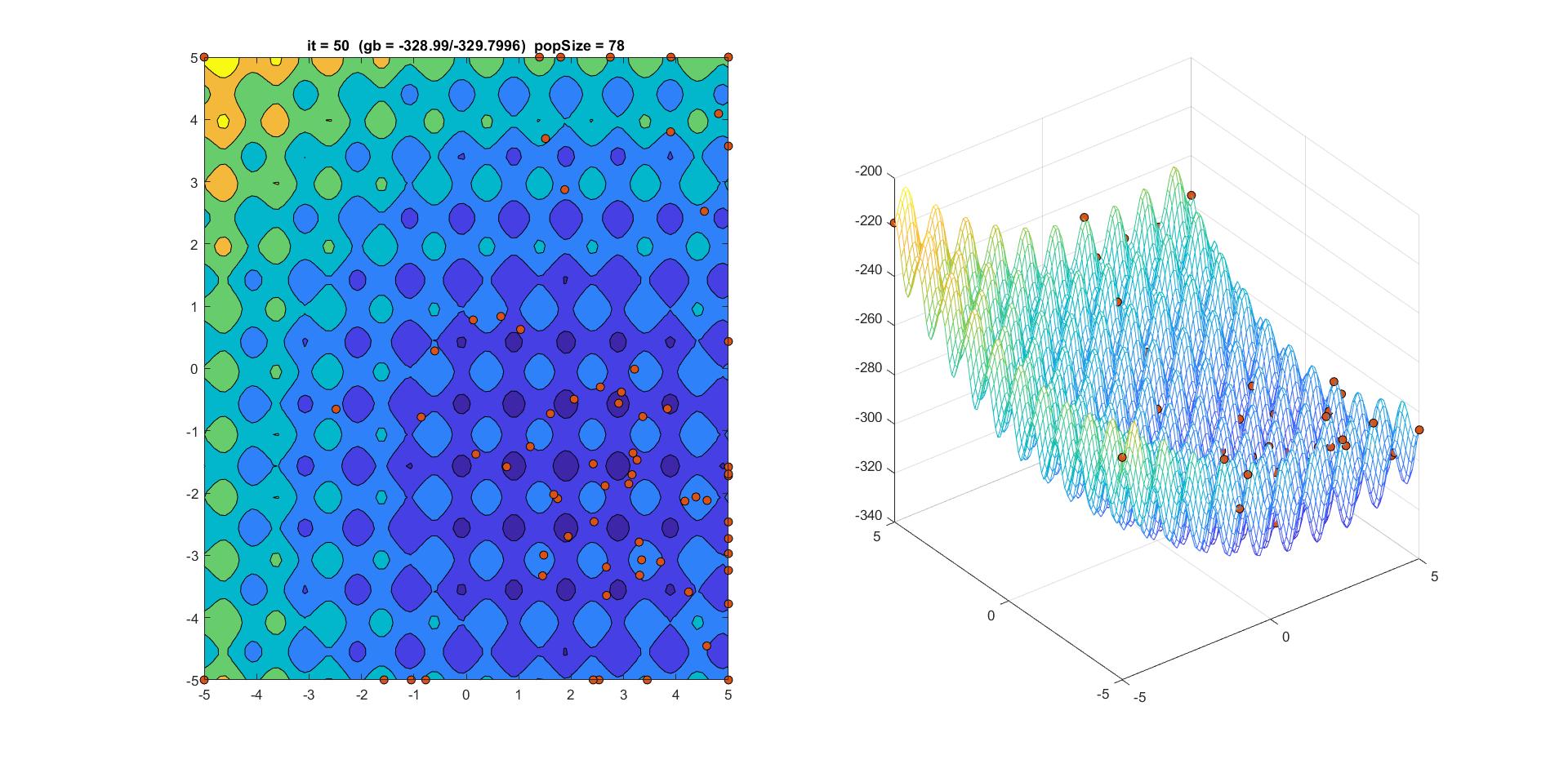
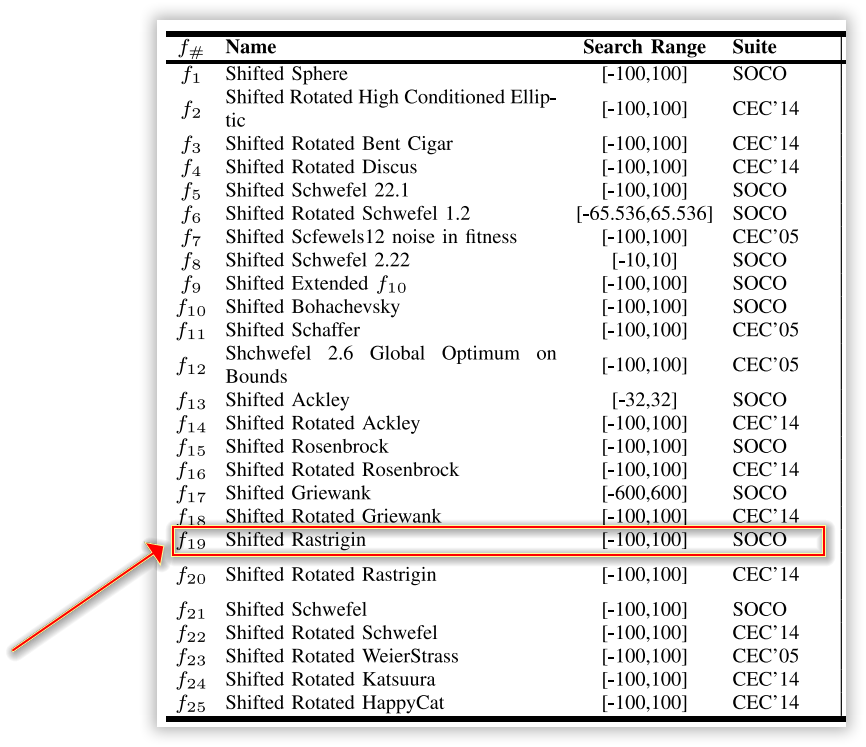


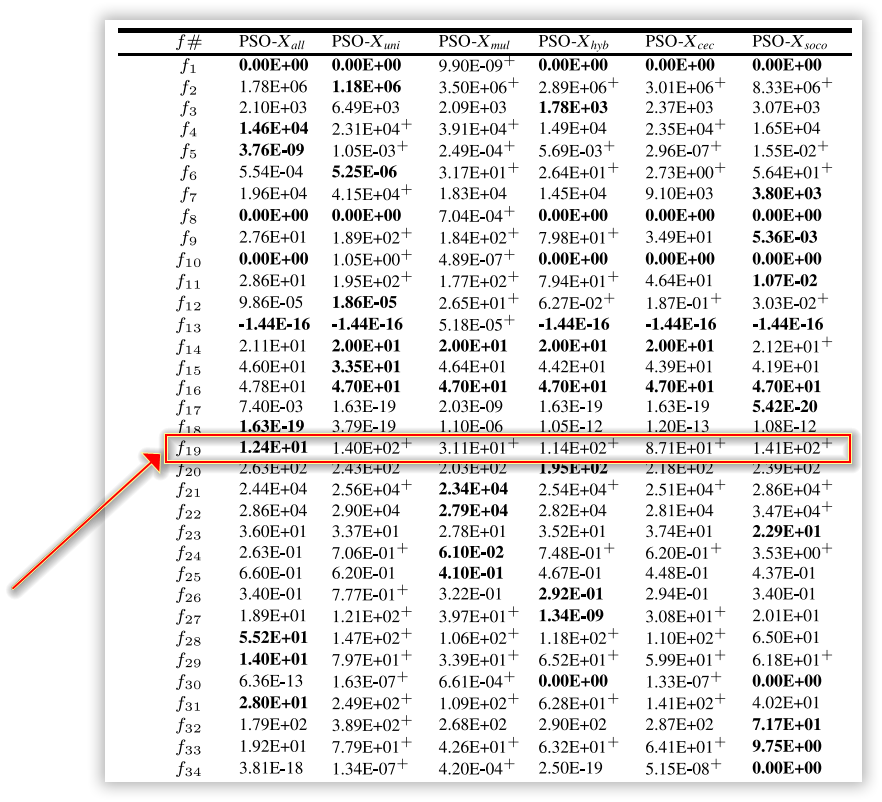
Figure 10 – Iteration 50 for B configuration for PSO

By synergistically combining exploration and exploitation, our optimization setup demonstrates enhanced performance in navigating the complex landscape of the Shifted Rastrigin's benchmark function. Through iterative generations and continuous adaptation, our algorithm strives to uncover the global minimum by efficiently exploring the search space and effectively exploiting high-quality regions.

Upon comparing our results with the reference paper, we found that our optimization setup outperformed the benchmark significantly. This improvement can be attributed to the synergistic utilization of the Genetic Algorithm (GA) and the grand parenting method, which effectively combined exploration and exploitation elements in our approach. the inclusion of GA in our optimization setup allowed for a more robust and efficient search in the solution space. GA's ability to explore various combinations of parameters and components enhanced the diversity of solutions and increased the chances of finding the global minimum. By leveraging GA's evolutionary principles, such as selection, crossover, and mutation, we were able to effectively explore and exploit the solution space, surpassing the limitations of simpler optimization algorithms.

Figure 11 – Results for reference paper





Furthermore, the integration of the grand parenting method added an additional layer of exploration and exploitation. This technique facilitated the identification of the optimal component type, which played a crucial role in the subsequent steps of the optimization process. By combining the strengths of exploration and exploitation, the grand parenting method enhanced our ability to navigate the complex solution landscape and overcome local optima. the synergistic interplay between GA and the grand parenting method within our optimization setup proved to be a key factor in achieving significantly improved results compared to the reference paper. The combined exploration and exploitation elements allowed for a more comprehensive exploration of the search space while effectively exploiting promising regions. This dynamic approach enabled us to overcome challenges posed by the benchmark problem and achieve superior optimization outcomes.

Overall, our results highlight the effectiveness of incorporating GA and the grand parenting method in optimizing complex problems. By capitalizing on the synergistic interaction between exploration and exploitation, our approach demonstrates significant advancements over traditional optimization algorithms, reinforcing the importance of considering these elements in the design of optimization methodologies.

**CONCLUSION**

An important consideration in optimization methods, particularly population-based approaches, is the number of function evaluations required to find the optimal point. While our optimization setup has demonstrated superior performance in terms of finding the global minimum, it is crucial to examine and compare the number of function evaluations employed in future studies.

The number of function evaluations directly impacts the computational cost and efficiency of the optimization process. A higher number of evaluations may increase the computational time, limiting the practicality of the method in certain applications. On the other hand, a lower number of evaluations may result in faster convergence, but there is a possibility of sacrificing the accuracy of the solution.

To comprehensively evaluate the effectiveness of our optimization setup, it would be valuable to compare the number of function evaluations with other existing methods or reference papers. This comparative analysis will provide insights into the efficiency and effectiveness of our approach in terms of achieving the desired solution within a reasonable computational time.

By conducting further studies that specifically investigate the number of function evaluations, we can gain a deeper understanding of the trade-off between accuracy and computational cost. This analysis will help determine the optimal balance in terms of computational efficiency and solution quality. Moreover, it can provide valuable insights for future optimization research and guide the development of more efficient and effective algorithms.

In summary, considering and comparing the number of function evaluations in future studies is essential to comprehensively assess the performance and practicality of our optimization setup. This analysis will contribute to the ongoing refinement and improvement of population-based optimization methods, ensuring their suitability for various applications and computational constraints.

**References**