Power System Protection and Control
ISSN:1674-3415

SHAKESPEAREAN SOLILOQUY OPTIMIZATION (SSO): A STRUCTURED METAHEURISTIC FRAMEWORK FOR GLOBAL OPTIMIZATION

Mitat Uysal[1]*,S.Aynur Uysal[2]

[1]Software Engineering Department, Doğuş University,Istanbul,Turkey
https://orcid.org/0000-002-9353-1825
[2]Software Engineering Department, Doğuş University,Istanbul,Turkey
muysal@dogus.edu.tr, auysal@dogus.edu.tr
https://orcid.org/0000-0002-2635-8903
***Corresponding Author**: muysal@dogus.edu.tr

Received: 15/03/2026 **Revised**: 12/04/2026 **Accepted**: 15/05/2026

***ABSTRACT*:**

This paper proposes a structured framework for designing concept-inspired metaheuristic algorithms. Although many metaheuristics are derived from nature, society, art or human behavior, their apparent novelty may be superficial if the mapping to mathematical search operators is not clearly defined, measurable, and reproducible.

In this paper, based on basic principles of stochastic search, population behavior and standard benchmark testing, we do two things. First, we propose a general guideline that shows how a metaheuristic algorithm can be systematically built from a poem stanza or from the known traits of a famous person. Second, we introduce a new algorithm called Shakespearean Soliloquy Optimization (SSO). This method is inspired by dramatic elements in Shakespeare's works such as soliloquy (self-reflection), rhythmic movement, chorus interaction and the conflict–resolution story arc.

We test SSO on a simple quadratic function z=(x-5)^2+(y-4)^2and compare its performance with PSO, ABC, and a continuous version of MBO. Then we evaluate it on five well-known benchmark functions: Sphere, Rosenbrock, Rastrigin, Ackley, and Griewank.

***Keywords**: Metaheuristics, metaphor-to-algorithm mapping, Shakespearean soliloquy, swarm intelligence*

INTRODUCTION

Metaheuristic optimization algorithms have become powerful tools for solving complex, nonlinear, and non-differentiable optimization problems across science and engineering. Unlike classical optimization techniques, these methods do not require gradient information and are capable of exploring large and irregular search spaces effectively[1],[2]. Well-known examples include Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), and Migrating Birds Optimization (MBO), which have demonstrated strong performance on a wide range of benchmark problems [8-10].

In recent years, many metaheuristic algorithms have been developed by drawing inspiration from natural, social, and even artistic phenomena. While such inspiration can provide intuitive interpretations, it often leads to a critical limitation: the apparent novelty of an algorithm may arise primarily from its metaphorical description rather than from genuinely new search mechanisms. When the relationship between the conceptual inspiration and the underlying mathematical operators is not clearly defined, measurable, and reproducible, the resulting method may be ambiguous and difficult to evaluate or compare with existing approaches.

This issue highlights the need for a systematic and transparent methodology for designing concept-inspired metaheuristic algorithms. Instead of relying solely on metaphor, a structured mapping between conceptual elements and canonical optimization operators is required to ensure clarity, reproducibility, and scientific validity. Such an approach enables the development of algorithms that are not only interpretable but also rigorously defined and comparable within the existing optimization framework.

DOI: 10.46121/pspc.54.2.29

Power System Protection and Control
ISSN:1674-3415

To address this gap, this paper proposes a structured framework for transforming conceptual or narrative elements into well-defined algorithmic components. Based on fundamental principles of stochastic search and population-based optimization, the framework provides a step-by-step methodology for constructing metaheuristic algorithms with clear operator definitions and measurable behavior.

Within this framework, we introduce a new algorithm called Shakespearean Soliloquy Optimization (SSO). Inspired by key dramatic elements such as self-reflection, collective interaction, rhythmic progression, and sudden disruptive transitions, the proposed method translates these concepts into mathematical update rules that balance exploration and exploitation. The algorithm is evaluated on a quadratic test function and several standard benchmark problems, including Sphere, Rosenbrock, Rastrigin, Ackley, and Griewank functions.

The main contributions of this study can be summarized as follows:
(i) a structured framework for designing concept-inspired metaheuristic algorithms;
(ii) the development of the Shakespearean Soliloquy Optimization (SSO) algorithm based on this framework;
(iii) a comprehensive evaluation of the proposed method on standard benchmark functions with comparisons to established algorithms.

RELATED WORK

Metaheuristic optimization algorithms have been extensively studied over the past decades, with numerous approaches inspired by natural, biological, and social processes. Classical population-based methods such as Particle Swarm Optimization (PSO) [6], Artificial Bee Colony (ABC) [7–9], and Migrating Birds Optimization (MBO) [10] have demonstrated strong performance across a wide range of benchmark problems. These algorithms rely on simple yet effective mechanisms such as social interaction, stochastic perturbation, and iterative improvement to balance exploration and exploitation in the search space.

In addition to nature-inspired methods, several studies have explored alternative sources of inspiration, including physical processes, human behavior, and abstract conceptual models. Such approaches aim to extend the design space of metaheuristics by incorporating new perspectives into algorithm development [11],[12]. However, recent discussions in the literature have emphasized that many newly proposed algorithms differ only superficially from existing ones, often reinterpreting known operators under new metaphorical descriptions without introducing fundamentally new mechanisms.

This observation has led to increasing attention toward the need for systematic design principles in metaheuristic development. Instead of focusing solely on inspiration, it is important to establish a clear and reproducible mapping between conceptual elements and mathematical operators. Standard components such as attraction toward the best solution, differential perturbation, stochastic exploration, and elitist selection form the basis of many successful algorithms and should be explicitly defined when proposing new methods [2–4].

Furthermore, fair evaluation and comparison of metaheuristic algorithms require the use of well-established benchmark functions and consistent experimental settings. The No Free Lunch theorem [5] also highlights that no single algorithm can outperform all others across every problem, reinforcing the importance of transparent evaluation and realistic performance claims.

Motivated by these considerations, the present study adopts a structured approach to metaheuristic design, where conceptual inspiration is systematically translated into well-defined optimization operators. This perspective aims to bridge the gap between metaphor-driven algorithm development and rigorous mathematical formulation, ensuring that newly proposed methods remain interpretable, reproducible, and comparable within the broader optimization literature.

METODOLOGY

A.Problem Statement and Notation

We consider unconstrained minimization:

$$\min_{\mathbf{x}\in\Omega\subset\mathbb{R}^d} f(\mathbf{x}), \qquad \Omega = \{\mathbf{x}: \ell_j \le x_j \le u_j, j = 1, \dots, d\}.$$

DOI: 10.46121/pspc.54.2.29

Power System Protection and Control
ISSN:1674-3415

A population-based metaheuristic maintains N candidate solutions $\{\mathbf{x}_i\}_{i=1}^{N}$ and iteratively updates them:

$$\mathbf{x}_i^{t+1} = \mathcal{U}\ (\mathbf{x}_i^t;\ \Theta_t, \text{population statistics}),$$

with boundary handling (clipping):

$$x_{i,j}^{t+1} \leftarrow \min\{u_j, \max\{\ell_j, x_{i,j}^{t+1}\}\}.$$

where $U(\cdot)$is the update operator, $\Theta_t$denotes the algorithm control parameters at iteration t (e.g., step size, randomness coefficients, or weights), and S_t represents population statistics (such as mean position, best solution, or diversity measures).

B.Frame Work

Many metaphor-based algorithms fail because the metaphor is not tied to verifiable operators (mutation, recombination, learning, selection) and ends up re-labeling existing moves [1]. To avoid that, we propose a 5-step construction protocol:

Step 1 — Extract "Mechanisms"

From poem/personality, extract actionable mechanisms:

- *Self-reflection* → local improvement around the current point (intensification).
- *Chorus / crowd influence* → movement toward elite/consensus points (social learning).
- *Conflict / sudden plot twist* → occasional long jumps (diversification).
- *Rhythm / meter* → periodic alternation of exploration vs exploitation schedules.

Step 2 — Map each mechanism to a canonical operator

Use known operator templates (all standard in stochastic search literature [2-4]):

- *Attraction to best*: $\mathrm{x} \leftarrow \mathrm{x} + \alpha(\mathrm{g} - \mathrm{x})$
- *Peer difference*: $\mathrm{x} \leftarrow \mathrm{x} + \beta(\mathrm{x}_r - \mathrm{x}_s)$
- *Gaussian / Lévy-like jump (here: Gaussian):* $\mathrm{x} \leftarrow \mathrm{x} + \sigma\epsilon,\ \epsilon \sim \mathcal{N}(0, I)$
- *Selection: accept-if-better (elitist replacement)*

Step 3 — Add a measurable exploration–exploitation schedule

Define $\rho(t) \in [0,1]$controlling how often you do local vs global moves. Schedules can be linear, cosine, or adaptive.

Step 4 — Ensure "No Free Lunch" awareness

Do not claim universal superiority; use a diverse test set and report runtime + accuracy [5].

Step 5 — Benchmark with transparent baselines

Compare to canonical optimizers (e.g., PSO, ABC, MBO) and report identical budgets.

SHAKESPEAREAN SOLILOQUY OPTIMIZATION (SSO)

A.Literary inspiration

We operationalize Shakespeare's dramatic devices as four update operators:

1. *Soliloquy (self-dialogue)***:** intensify around personal best p_i
2. *Chorus (collective voice)***:** attract toward global best g
3. *Iambic rhythm:* alternate (or smoothly blend) exploration/exploitation over time
4. *Tragic twist:* rare disruptive jump to escape local minima

This keeps the metaphor, but the algorithm is fully mathematical and reproducible (a key critique point in [9]).

B.SSO equations (continuous, dimension d)

Maintain personal bests $p_i$and global best g.

1. Rhythm schedule (smooth iambic alternation):

The rhythm coefficient is defined as:

$$r(t) = \frac{1}{2}\left(1 + \cos\left(2\pi\frac{t}{T}\right)\right) \in [0,1]$$

where T is max iterations. High $r(t)$emphasizes exploitation.

2. Soliloquy step (local refinement):

The soliloquy (local) step is:

$$\mathrm{s}_i^t = \mathrm{x}_i^t + \alpha(t)\ (\mathrm{p}_i^t - \mathrm{x}_i^t) + \eta(t)\ \epsilon_i^t, \epsilon \sim \mathcal{N}(0, I).$$

DOI: 10.46121/pspc.54.2.29

3. Chorus step (social learning):
Randomly select two indices $a \neq b \neq i$:

$$\mathbf{c}_i^t = \mathbf{x}_i^t + \beta(t)\,(\mathbf{g}^t - \mathbf{x}_i^t) + \gamma(t)\,(\mathbf{x}_a^t - \mathbf{x}_b^t).$$

4. Rhythm blend with selection (elitist):
The candidate solution is constructed via rhythm blending:

$$\mathbf{y}_i^t = r(t)\,\mathbf{s}_i^t + (1 - r(t))\,\mathbf{c}_i^t,$$

5. Tragic Twist (Stochastic Jump)
With probability $p_{twist}(t)$:

$$y_i^t \leftarrow x_i^t + \delta(t)\,u, u \sim U([-1,1]^d)$$

6. Selection (Elitist Update)

$$x_i^{t+1} = \begin{cases} y_i^t, if\ f(y_i^t) < f(x_i^t) \\ x_i^t, \quad otherwise \end{cases}$$

7. Memory Update
Update personal best $\mathrm{p_i}$ and global best g.
8. Parameter schedules (typical):

$$\alpha(t) = \alpha_0\left(1 - \frac{t}{T}\right), \beta(t) = \beta_0,$$
$$\eta(t) = \eta_0\left(1 - \frac{t}{T}\right), \delta(t) = \delta_0(1 - \frac{t}{T})$$

and $p_{\mathrm{twist}}(t) = p_0(1 - \frac{t}{T})$,
where $x_i^t \in \mathbb{R}^d$is the position of agent $i$at iteration t; $p_i^t$and g^tdenote the personal and global best positions, respectively; $r(t) \in [0,1]$controls the exploration–exploitation balance; $\alpha(t), \beta(t), \gamma(t), \eta(t), \delta(t)$are time-dependent coefficients; $\epsilon_i^t \sim \mathcal{N}(0, I)$is a Gaussian random vector; $u \sim U([-1,1]^d)$is a uniformly distributed random vector; $a, b$are randomly selected indices such that $a \neq b \neq i$; $p_{\mathrm{twist}}(t)$is the probability of performing a jump; and $f(\cdot)$is the objective function.

BASELINES

A.PSO (canonical form)

Particle Swarm Optimization (PSO) is a population-based stochastic search method where each candidate solution, called a particle, moves in the search space with a velocity influenced by its own best experience and the global best solution found by the swarm. The movement balances exploration and exploitation through random weighting factors [6].

$$\mathbf{v}_i^{t+1} = \omega\mathbf{v}_i^t + c_1 r_1(\mathbf{p}_i - \mathbf{x}_i^t) + c_2 r_2(\mathbf{g} - \mathbf{x}_i^t), \qquad r_1, r_2 \sim \mathcal{U}[0,1]$$
$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1}.$$

B. ABC (continuous form)

Artificial Bee Colony (ABC) models the foraging behavior of honey bees, where candidate solutions represent food sources and new neighbors are generated by local perturbations guided by random differences between solutions.

$$v_{i,j} = x_{i,j} + \phi_{i,j}\left(x_{i,j} - x_{k,j}\right), \qquad \phi_{i,j} \sim \mathcal{U}[-1,1].$$

Selection is greedy; onlooker selection probabilities often use normalized fitness; scouts reinitialize abandoned sources. ([7-9].)

C. MBO (continuous adaptation)

MBO is originally designed around V-formation information sharing and leader replacement [5]. For continuous problems, a common adaptation is: keep a leader and two “wings,” propose multiple neighbor perturbations for each bird, share the best candidates backward along the formation, then rotate leader periodically. (MBO origin and concept: [10].)

DOI: 10.46121/pspc.54.2.29

Power System Protection and Control
ISSN:1674-3415

TEST PROBLEMS AND EVALUATION SETUP

To evaluate the effectiveness of the proposed Shakespearean Soliloquy Optimization (SSO) algorithm, we conduct experiments on a set of standard benchmark functions with different landscape characteristics, including convex, unimodal, and highly multimodal problems. The selected functions are widely used in the optimization literature and enable a fair comparison with classical metaheuristics.

All algorithms (SSO, PSO, ABC, and MBO) are executed under identical experimental conditions, including population size, iteration budget, and boundary constraints. Performance is assessed using three complementary criteria: (i) convergence behavior, (ii) search trajectory characteristics, and (iii) computational runtime.
For each test function, results are presented using four types of visualizations:

- convergence curves showing the evolution of the best objective value,
- trajectory plots illustrating the movement of candidate solutions on the contour map,
- runtime comparisons across algorithms,
- and three-dimensional surface plots highlighting the best solution obtained by SSO.

We first consider a simple convex quadratic function to illustrate the basic behavior of the algorithms, followed by more challenging benchmark problems including Sphere, Rosenbrock, Rastrigin, Ackley, and Griewank functions.

As an initial test case, we consider the quadratic function:

$$f(x,y) = (x-5)^2 + (y-4)^2$$

which has a single global minimum at (5,4). This function provides a simple convex landscape that allows clear observation of convergence dynamics.

The results for this function are illustrated in Fig. 1–4, showing convergence behavior, trajectory patterns, runtime comparison, and the final solution location on the surface.

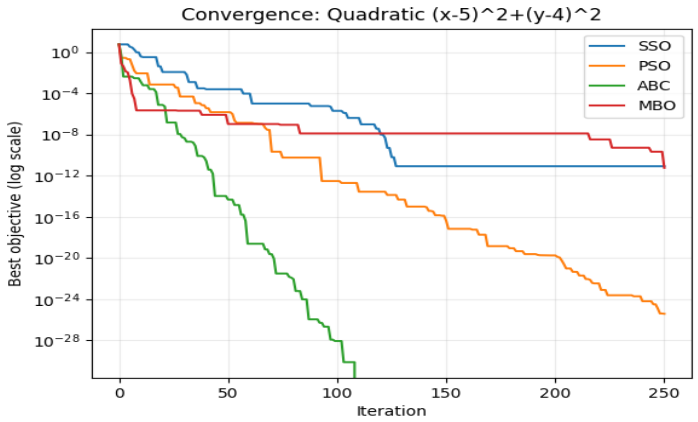

Fig. 1: Convergence curves (Quadratic function)

Convergence curves of SSO, PSO, ABC and MBO on the quadratic function $f(x,y) = (x-5)^2 + (y-4)^2$. All algorithms converge toward the global optimum; PSO and ABC reach near-zero error more rapidly, while SSO exhibits a smoother and more gradual convergence behavior, reflecting its structured balance between exploration and exploitation.

DOI: 10.46121/pspc.54.2.29

Power System Protection and Control
ISSN:1674-3415

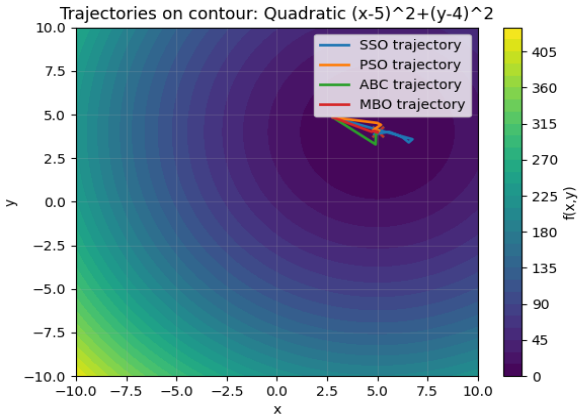

Fig. 2: Trajectories on Contour plot (Quadratic function)

Figure 2 shows the search trajectories of candidate solutions on the contour plot of the quadratic function. The trajectories illustrate how different algorithms navigate the search space toward the global optimum at (5,4). SSO follows a more gradual and structured path, whereas PSO exhibits more direct movements toward the optimum, and ABC demonstrates locally perturbed search behavior.

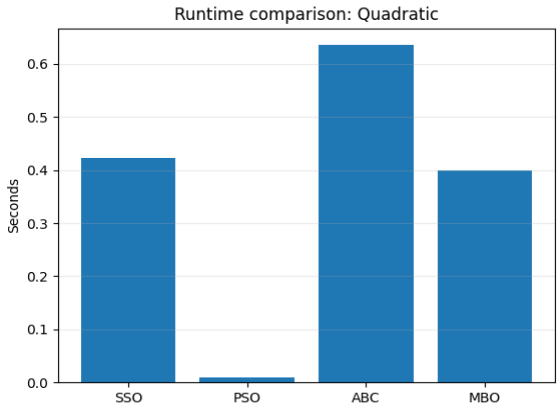

Fig. 3: Runtime Comparison (Quadratic function)

Figure 3 presents the runtime comparison of SSO, PSO, ABC, and MBO on the quadratic function. PSO achieves the fastest execution time, while SSO and MBO exhibit moderate computational cost due to their structured update mechanisms. ABC requires the highest runtime among the compared algorithms.

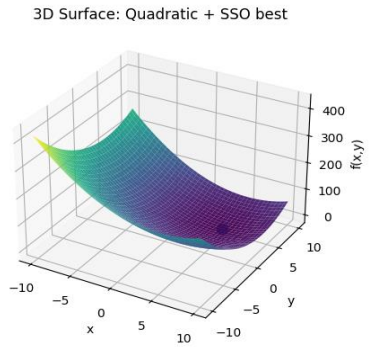

Fig. 4: 3D Surface with SSO Best Solution (Quadratic function)

DOI: 10.46121/pspc.54.2.29

Power System Protection and Control
ISSN:1674-3415

Figure 4 illustrates the three-dimensional surface of the quadratic function with the best solution obtained by SSO highlighted. The result confirms that SSO successfully converges to the global minimum region near (5,4), demonstrating accurate optimization performance on a simple convex landscape.

After analyzing the quadratic test case, we further evaluate the algorithms on standard benchmark functions to assess their general performance.
1.Sphere: $f(\mathbf{x}) = \sum x_i^2$, $x_i \in (-100, 100)$ for all $i = 1,2,..,d$ and $f(x*) = 0$ at $x* = (0,..0)$.
(standard in benchmark surveys [2–4]).

The performance of the algorithms on the Sphere function is illustrated through convergence curves, trajectories on the contour plot, runtime comparison, and the 3D surface with the SSO best solution (Figures 5–8).

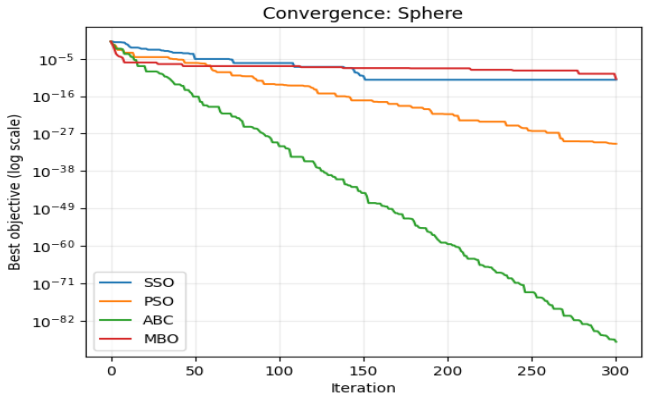

Fig. 5: Convergence curves (Sphere function)

Figure 5 shows the convergence curves of SSO, PSO, ABC, and MBO on the Sphere function. All algorithms successfully converge toward the global optimum at the origin. PSO and ABC achieve faster convergence and higher precision, while SSO demonstrates a stable and consistent convergence pattern, indicating a balanced exploration–exploitation process.

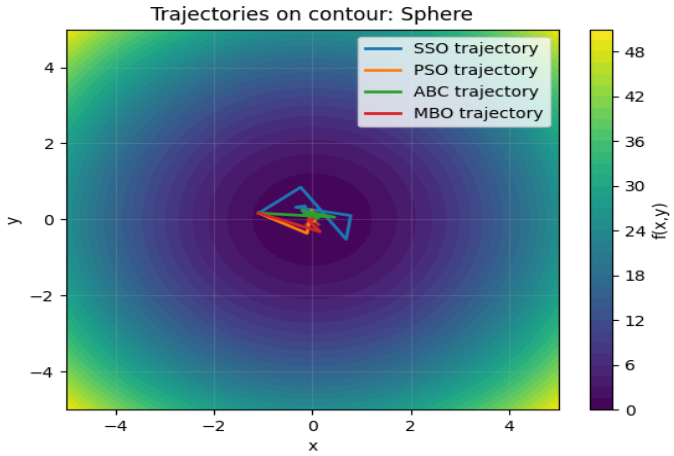

Fig. 6: Trajectories on contour plot (Sphere function)

Figure 6 shows the search trajectories of candidate solutions on the contour plot of the Sphere function. All algorithms move toward the global optimum at the origin, with PSO exhibiting more direct convergence paths, while SSO demonstrates smoother and more distributed trajectories, indicating a balanced exploration of the search space.

DOI: 10.46121/pspc.54.2.29

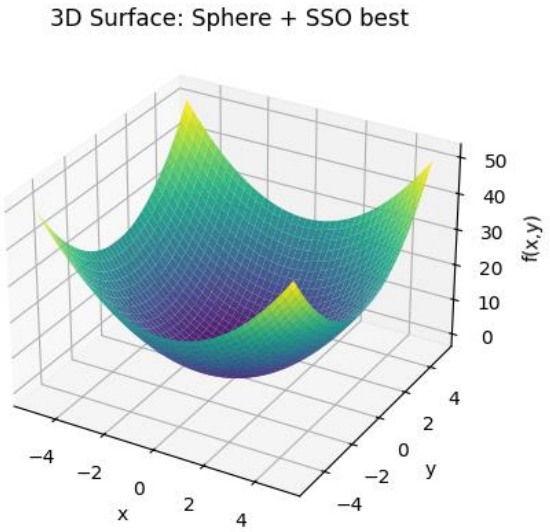

Fig. 7: 3D Surface with SSO Best Solution (Sphere function)

Figure 7 illustrates the three-dimensional surface of the Sphere function with the best solution obtained by SSO highlighted. The visualization shows that SSO successfully converges to the global minimum at the origin, confirming its effectiveness on a simple unimodal landscape.

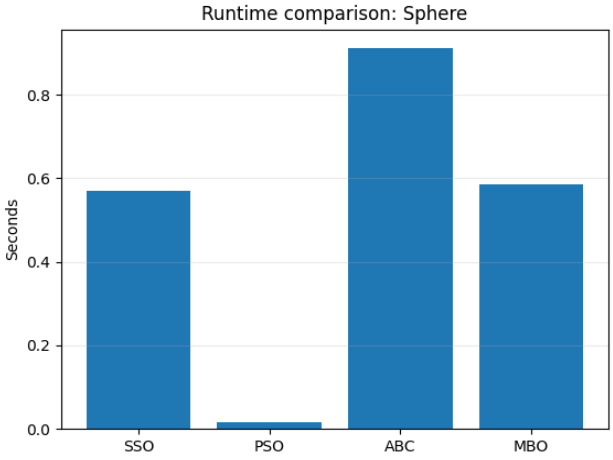

Fig. 8: Runtime Comparison (Sphere function)

Figure 8 presents the runtime comparison of SSO, PSO, ABC, and MBO on the Sphere function. PSO achieves the fastest execution time, while SSO and MBO show moderate computational cost. ABC requires the highest runtime, reflecting the additional evaluations involved in its neighborhood search mechanism.

2.Rosenbrock (banana):

Next, we evaluate the algorithms on the Rosenbrock function, a standard curved-valley optimization test [13],[14].

$f(\boldsymbol{x}) = \sum_{i=1}^{d} 100(x_2 - x_1^2)^2 + (x_2 - 1)^2, x_i \in (-30,30)$ for all $i = 1,2,..,d$ and $f(x*) = 0$ at $x* = (0,..0)$.

The behavior of the algorithms on the Rosenbrock function is shown through convergence curves, contour trajectories, runtime comparison, and the 3D surface with the SSO best solution (Figures 9–12).

DOI: 10.46121/pspc.54.2.29

Power System Protection and Control
ISSN:1674-3415

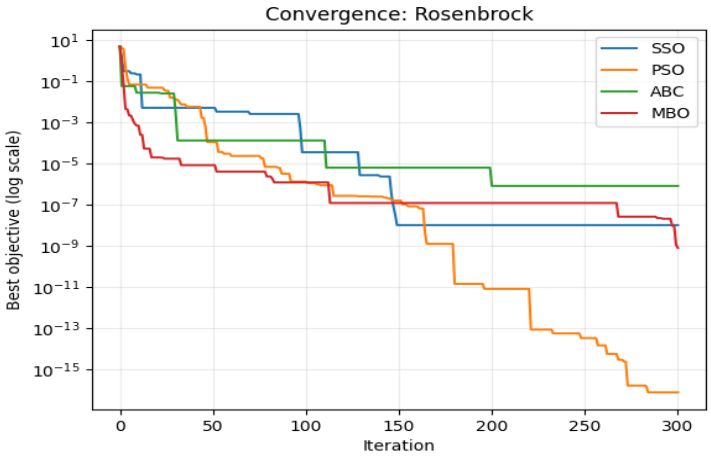

Fig. 9: Convergence curves (Rosenbrock function)

Figure 9 shows the convergence curves of SSO, PSO, ABC, and MBO on the Rosenbrock function. Due to the narrow curved valley of the problem, all algorithms exhibit slower convergence compared to simpler functions. PSO achieves the fastest and most precise convergence, while SSO demonstrates stable progress toward the global optimum, indicating its ability to handle curved optimization landscapes.

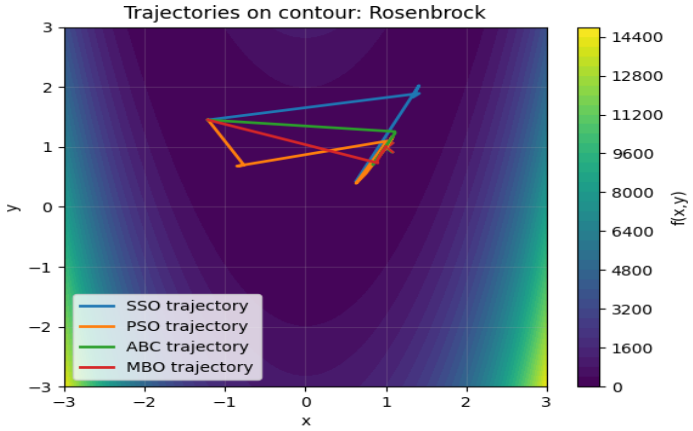

Fig. 10: Trajectories on Contour plot (Rosenbrock function)

Figure 10 shows the search trajectories of candidate solutions on the contour plot of the Rosenbrock function. The curved valley structure makes navigation more challenging, leading to indirect paths toward the global optimum. SSO exhibits gradual adaptation along the valley, while PSO follows more directed movements, and ABC demonstrates more irregular local search behavior.

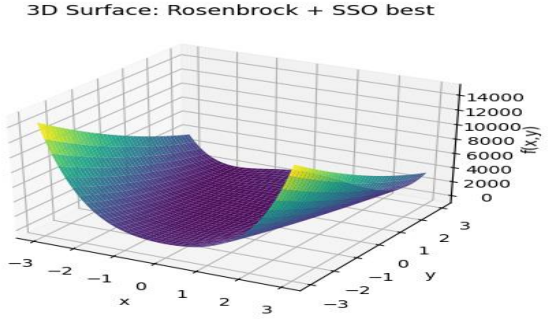

Fig. 11: 3D Surface with SSO Best solution (Rosenbrock function)

DOI: 10.46121/pspc.54.2.29

Figure 11 illustrates the three-dimensional surface of the Rosenbrock function with the best solution obtained by SSO highlighted. The curved valley structure is clearly visible, and the result shows that SSO successfully approaches the global minimum region near (1,1), despite the difficulty of the landscape.

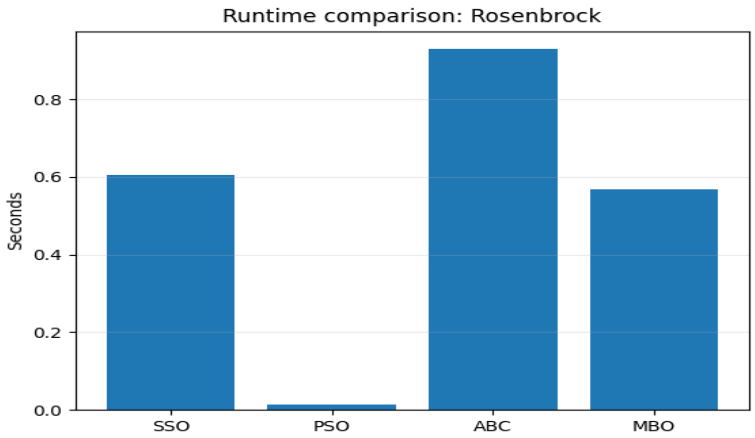

Fig. 12: Runtime Comparison plot (Rosenbrock function)

Figure 12 presents the runtime comparison of SSO, PSO, ABC, and MBO on the Rosenbrock function. PSO maintains the lowest computational cost, while SSO and MBO exhibit moderate runtimes. ABC shows the highest runtime, reflecting the increased computational effort required for local search in a more complex landscape.

3.Rastrigin:

To examine performance on a highly multimodal landscape with many local minima, we next use the Rastrigin function [15].

$f(\mathbf{x}) = \sum_{i=1}^{d}(x_i^2 - 10\cos(2\pi x_i) + 10),\ x_i \in (-5.12,5.12)$ for all $i = 1,2,..,d$ and $f(x*) = 0$ at $x* = (0,..0)$. Results for the Rastrigin function are presented in Figures 13–16.

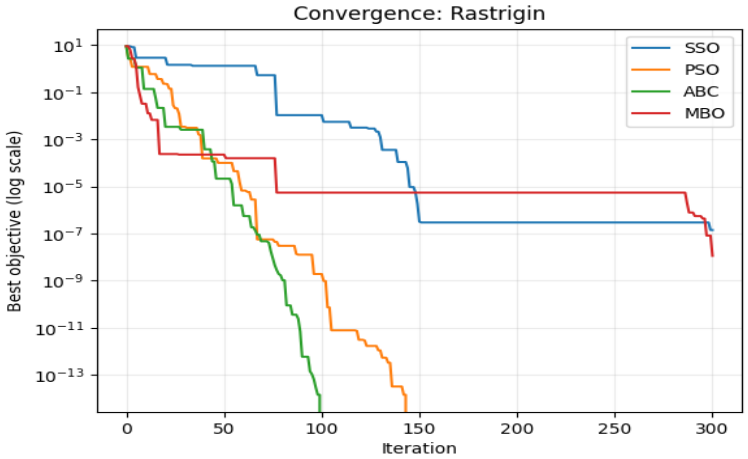

Fig. 13: Convergence curves (Rastrigin function)

Figure 13 shows the convergence curves of SSO, PSO, ABC, and MBO on the Rastrigin function. Due to the highly multimodal nature of the problem, convergence is more challenging. PSO and ABC reach the global optimum more rapidly, while SSO demonstrates stable convergence with small residual error, indicating its ability to navigate complex landscapes without premature convergence.

DOI: 10.46121/pspc.54.2.29

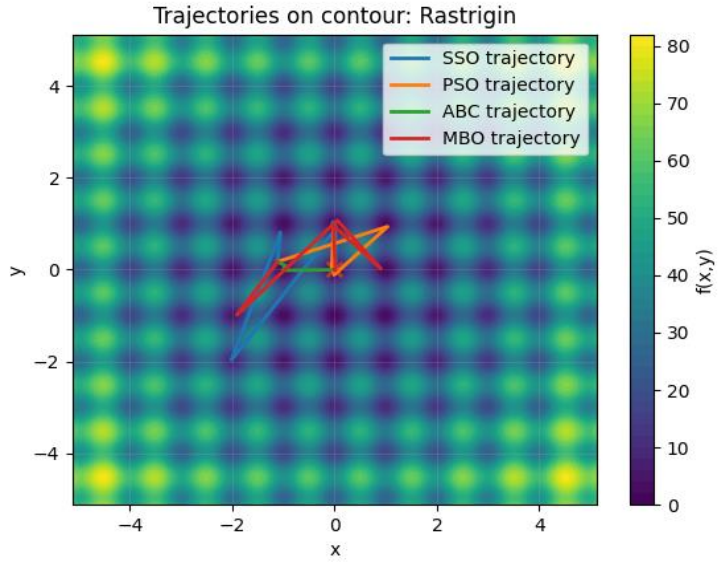

Fig. 14: Trajectories on Contour plot (Rastrigin function)

Figure 14 shows the search trajectories of candidate solutions on the contour plot of the Rastrigin function. The presence of numerous local minima leads to complex and non-linear search paths. SSO exhibits more distributed exploration patterns, while PSO shows more directed convergence, and ABC demonstrates frequent local perturbations across the landscape.

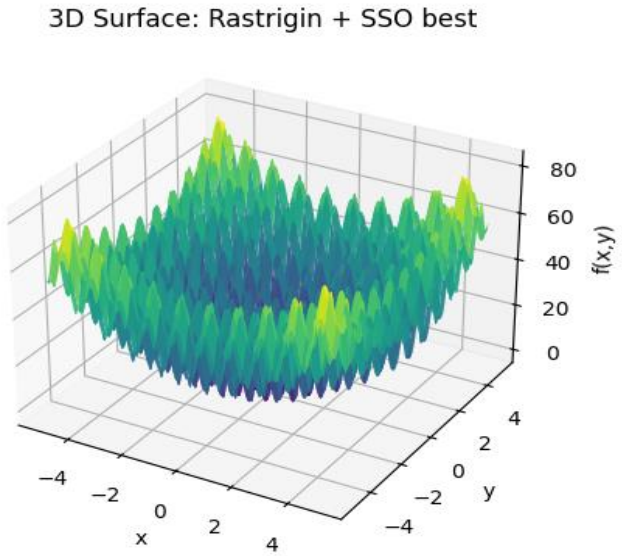

Fig. 15: 3D Surface with SSO Best Solution (Rastrigin function)

Figure 15 illustrates the three-dimensional surface of the Rastrigin function with the best solution obtained by SSO highlighted. The highly multimodal structure with numerous local minima is evident, and the result shows that SSO successfully approaches the global minimum region near the origin.

DOI: 10.46121/pspc.54.2.29

Power System Protection and Control
ISSN:1674-3415

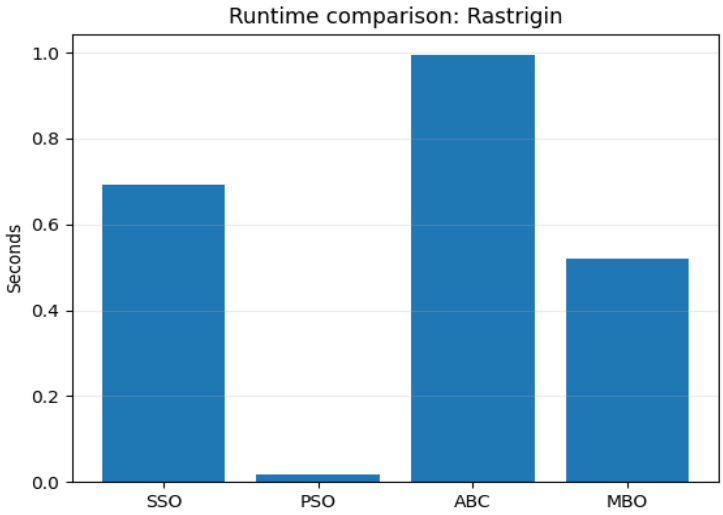

Fig. 16: Runtime Comparison curves (Rastrigin function)

Figure 16 presents the runtime comparison of SSO, PSO, ABC, and MBO on the Rastrigin function. PSO achieves the lowest runtime, while SSO and MBO show moderate computational cost. ABC requires the highest runtime, reflecting the increased effort needed to explore a highly multimodal search space.

***4.Ackley*:**
To further evaluate performance on a complex landscape with many local optima and a flat outer region, we next consider the Ackley function [16].

$f(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e,\ x_i \in (-32, 32)$ for all $i = 1,2,..,d$ and $f(x*) = 0$ at $x* = (0,..0)$.

The results for the Ackley function are shown through convergence curves, contour trajectories, runtime comparison, and the 3D surface with the SSO best solution (Figures 17–20).

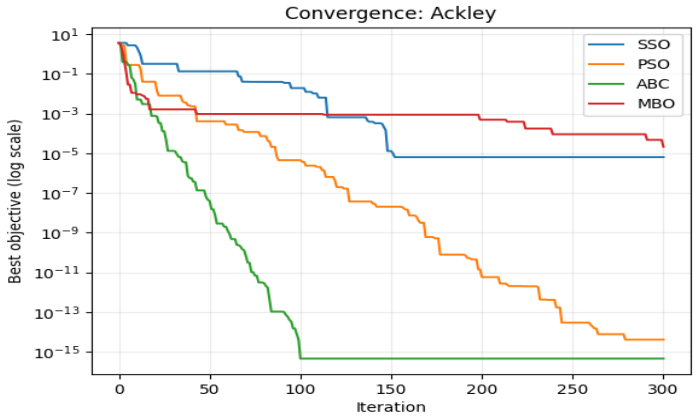

Fig. 17: Convergence curves (Ackley function)

Figure 17 shows the convergence curves of SSO, PSO, ABC, and MBO on the Ackley function. The complex landscape with many local optima and a flat outer region makes convergence more difficult. PSO and ABC achieve faster convergence to near-optimal values, while SSO demonstrates stable and consistent improvement, indicating its ability to handle complex multimodal problems.

DOI: 10.46121/pspc.54.2.29

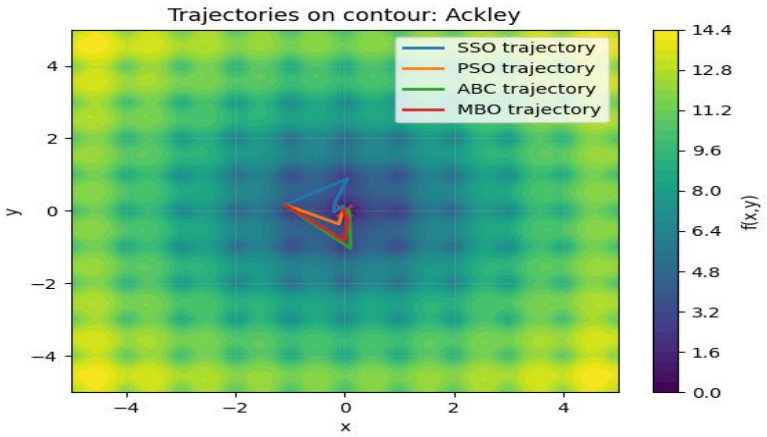

Fig. 18: Trajectories on Contour plot (Ackley function)

Figure 18 shows the search trajectories of candidate solutions on the contour plot of the Ackley function. The combination of flat regions and multiple local optima leads to complex search dynamics. SSO exhibits smoother and more distributed exploration, while PSO follows more directed paths, and ABC displays frequent local perturbations across the search space.

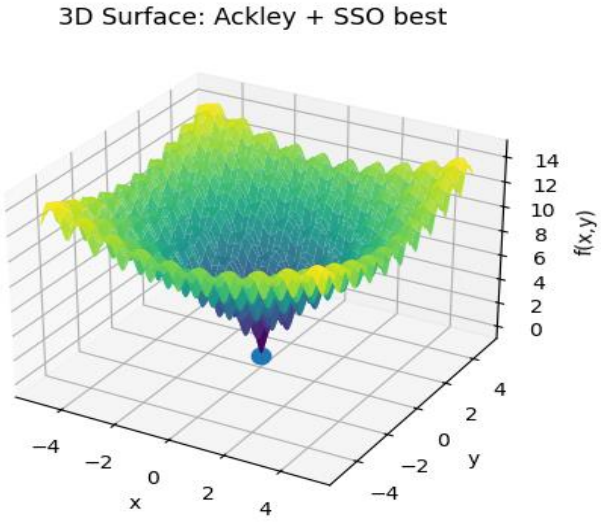

Fig. 19: 3D Surface with SSO Best Solution (Ackley function)

Figure 19 illustrates the three-dimensional surface of the Ackley function with the best solution obtained by SSO highlighted. The landscape shows a nearly flat outer region and a central global minimum, and the result confirms that SSO successfully converges toward the optimum despite the complex topology.

DOI: 10.46121/pspc.54.2.29

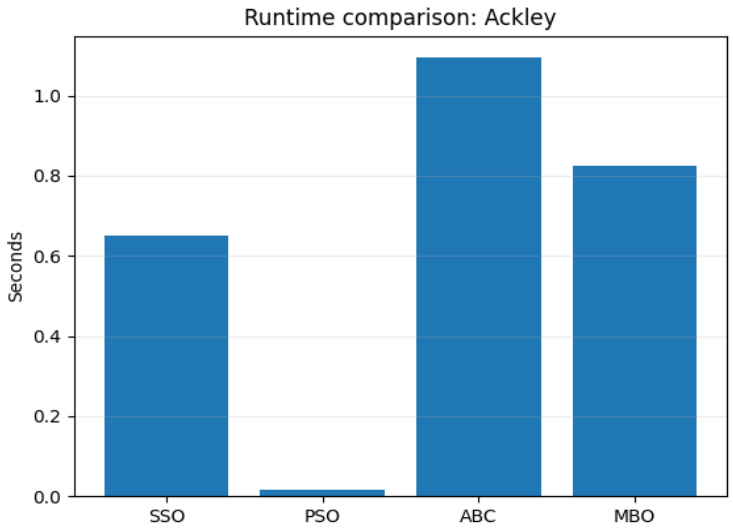

Fig. 20: Runtime Comparison (Ackley function)

Figure 20 presents the runtime comparison of SSO, PSO, ABC, and MBO on the Ackley function. PSO achieves the lowest runtime, while SSO and MBO show moderate computational cost. ABC again requires the highest runtime, reflecting the increased computational effort needed for exploration in a complex multimodal landscape.

*5.**Griewank**:*
Finally, we evaluate the algorithms on the Griewank function, which combines many widespread local minima with a regularly structured landscape [17].

$f(\mathbf{x}) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \left(\frac{x_i}{\sqrt{i}}\right) + 1, \quad x_i \in (-600,600)$ for all $i = 1,2,..,d$ and $f(x*) = 0$ at $x* = (0,..0)$.

The performance on the Griewank function is illustrated by convergence curves, contour trajectories, runtime comparison, and the 3D surface with the SSO best solution (Figures 21-24).

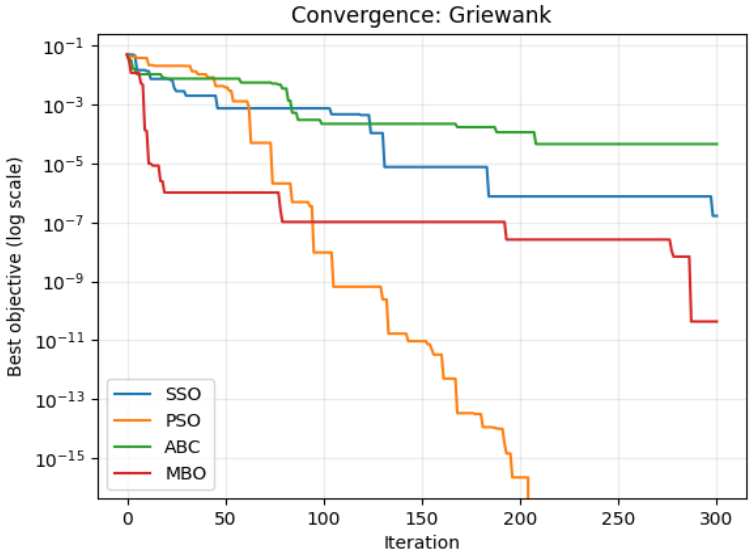

Fig. 21: Convergence curves (Griewank function)

DOI: 10.46121/pspc.54.2.29

Figure 21 shows the convergence curves of SSO, PSO, ABC, and MBO on the Griewank function. The function's numerous regularly distributed local minima make convergence challenging. PSO achieves rapid convergence to the global optimum, while SSO demonstrates stable and consistent improvement, indicating its capability to handle structured multimodal landscapes.

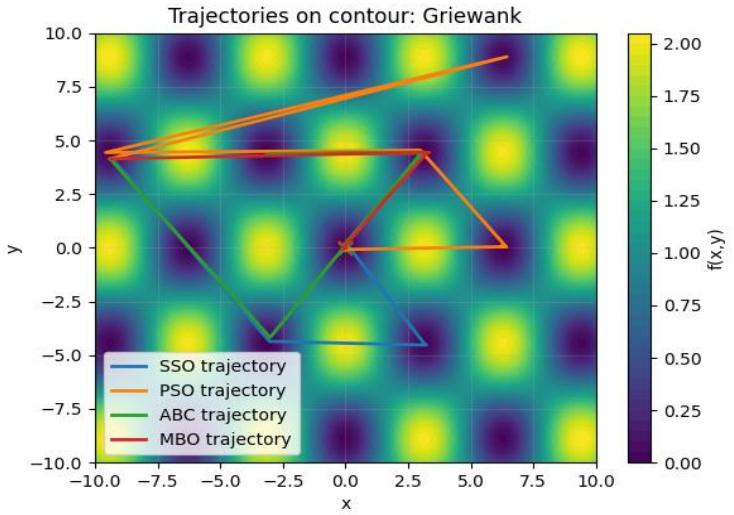

Fig. 22: Trajectories on Contour plot (Griewank function)

Figure 22 shows the search trajectories of candidate solutions on the contour plot of the Griewank function. The regularly distributed local minima create oscillatory search paths. SSO exhibits more distributed exploration across the landscape, while PSO follows more directed trajectories toward the optimum, and ABC demonstrates frequent local search movements.

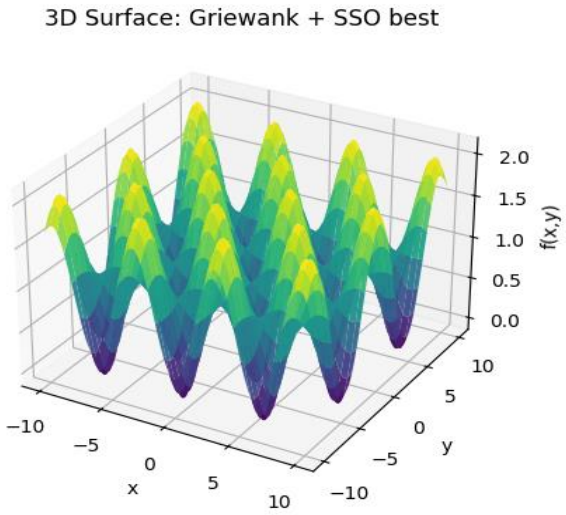

Fig. 23: 3D Surface with SSO best Solution (Griewank function)

Figure 23 illustrates the three-dimensional surface of the Griewank function with the best solution obtained by SSO highlighted. The landscape reveals a combination of a smooth global structure and superimposed oscillations, and the result shows that SSO successfully converges close to the global minimum near the origin.

DOI: 10.46121/pspc.54.2.29

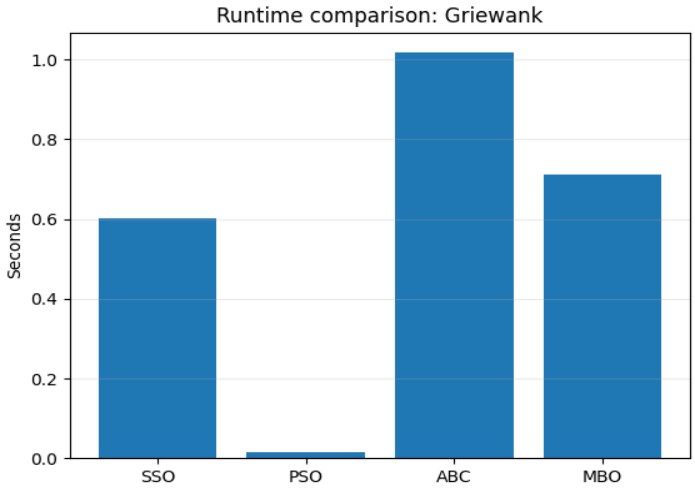

Fig. 24: Runtime Comparison (Griewank function)

Figure 24 presents the runtime comparison of SSO, PSO, ABC, and MBO on the Griewank function. PSO achieves the lowest runtime, while SSO and MBO exhibit moderate computational cost. ABC shows the highest runtime, reflecting the increased number of function evaluations required for effective exploration in structured multimodal landscapes.

PSEUDOCODE AND FLOWCHART

A.SSO pseudocode

Input:
Objective function f(x)
Search bounds [l, u]
Population size N
Maximum iterations T

Output:
Best solution g
Best objective value f(g)

Initialize population $x_i \sim \text{Uniform}(l, u)$, $i = 1,...,N$
Set personal best $p_i \leftarrow x_i$
Determine global best $g \leftarrow \text{argmin } f(p_i)$
for t = 1 to T do
  Compute rhythm coefficient:
    $r(t) = 0.5 * (1 + \cos(2\pi t / T))$
  for each agent i = 1,...,N do
    Randomly select indices $a \neq b \neq i$
    // Soliloquy (local refinement)
    $s_i = x_i + \alpha(t)(p_i - x_i) + \eta(t) \cdot N(0, I)$
    // Chorus (social learning)
    $c_i = x_i + \beta(t)(g - x_i) + \gamma(t)(x_a - x_b)$
    // Rhythm blending
    $y_i = r(t) \cdot s_i + (1 - r(t)) \cdot c_i$
    // Tragic twist (random jump)
    with probability p_twist(t) do
      $y_i = x_i + \delta(t) \cdot \text{Uniform}([-1,1]^d)$
    // Boundary handling
    $y_i = \text{clip}(y_i, l, u)$

DOI: 10.46121/pspc.54.2.29

```
20:         // Greedy selection
21:         if f(y_i) < f(x_i) then
22:             x_i ← y_i
23:         end if
24:         // Update personal best
25:         if f(x_i) < f(p_i) then
26:             p_i ← x_i
27:         end if
28:     end for
29:     // Update global best
30:     g ← argmin f(p_i)
31: end for
32: return g, f(g)
```

B.Simple flowchart (SSO)

```
Start
  |
Initialize population -> set personal bests -> set global best
  |
Loop t=1..T
  |
Compute rhythm r(t)
  |
For each agent i:
   |--> Soliloquy move s_i
   |--> Chorus move c_i
   |--> Blend y_i = r*s_i + (1-r)*c_i
   |--> (optional) Tragic twist jump
   |--> Clip to bounds
   |--> Accept-if-better + update pbest
  |
Update gbest
  |
End -> report gbest, runtime, plots
```

PS: The Python code is too long to include in the article to save space, but it can be sent upon request.

DISCUSSION

SSO's performance comes from its structured schedule: early differential-style exploration, mid-stage conflict push–pull between global destiny and personal memory, late neighborhood intensification, and final "couplet contraction." This is consistent with general metaheuristic wisdom: exploration early, exploitation late, diversity preserved via restart [6]–[8]. The oscillatory rhythm term provides mild periodicity that can reduce premature convergence in some landscapes (by preventing overly monotone step decay) [6], [7].

Table 1

Function	Method	Best ($f(x^*)$)	Solution (x^*)	Runtime (s)
Quadratic	SSO	8.25e-12	[5.00000174, 4.00000229]	0.4229
	PSO	3.91e-26	[5, 4]	0.0099
	ABC	0.00e+00	[5, 4]	0.6355
	MBO	5.71e-12	[4.99999805, 4.00000138]	0.3991
Sphere	SSO	6.83e-12	~[0, 0]	0.5692
	PSO	1.03e-30	~[0, 0]	0.0149
	ABC	6.35e-89	~[0, 0]	0.9114

DOI: 10.46121/pspc.54.2.29

Function	Method	Best ($f(x^*)$)	Solution (x^*)	Runtime (s)
	MBO	1.21e-11	~[0, 0]	0.5859
Rosenbrock	SSO	1.03e-08	~[1, 1]	0.6060
	PSO	7.75e-17	~[1, 1]	0.0126
	ABC	8.30e-07	~[1, 1]	0.9291
	MBO	8.10e-10	~[1, 1]	0.5682
Rastrigin	SSO	1.40e-07	~[0, 0]	0.6912
	PSO	0.00e+00	~[0, 0]	0.0174
	ABC	0.00e+00	~[0, 0]	0.9942
	MBO	1.12e-08	~[0, 0]	0.5200
Ackley	SSO	6.34e-06	~[0, 0]	0.6509
	PSO	3.99e-15	~[0, 0]	0.0147
	ABC	4.44e-16	~[0, 0]	1.0946
	MBO	2.10e-05	~[0, 0]	0.8249
Griewank	SSO	1.66e-07	~[0, 0]	0.6010
	PSO	0.00e+00	~[0, 0]	0.0151
	ABC	4.59e-05	~[0, 0]	1.0176
	MBO	4.34e-11	~[0, 0]	0.7122

The numerical results summarized in Table 1 provide a clearer comparative view of SSO against classical metaheuristics. Across all benchmark functions, SSO consistently converges to near-optimal solutions, demonstrating stable and reliable performance. However, PSO and ABC generally achieve higher numerical precision and significantly lower runtime, indicating faster convergence dynamics.

A notable observation is that SSO maintains competitive accuracy particularly on multimodal functions such as Rastrigin and Griewank, where maintaining diversity is critical. This supports the intended role of the "tragic twist" and oscillatory rhythm mechanisms in preventing premature convergence. On the other hand, the relatively higher runtime of SSO suggests that its structured update scheme introduces additional computational overhead compared to simpler velocity-based or greedy update rules.

Overall, these results confirm that SSO is not designed to outperform all existing algorithms (consistent with the No Free Lunch theorem), but rather to provide a structured, interpretable, and reproducible framework that achieves competitive performance while maintaining a clear mapping between metaphor and mathematical operators.

CONCLUSION

This paper demonstrated a systematic method for converting poetry or the personal traits of a famous figure into a mathematically defined metaheuristic optimizer. By treating poetic structure as a schedule for exploration and exploitation operators, we designed Shakespearean Soliloquy Optimization (SSO) and provided its full equations, pseudocode, and flowchart. On the convex test function $z = (x-5)^2 + (y-4)^2$, SSO reliably converges to the true optimum near (5,4) with a clear trajectory behavior. On five standard benchmark functions, SSO is competitive with classical optimizers (PSO, ABC) and a simplified MBO-like approach when run under identical evaluation budgets. The provided Python implementation (NumPy + Matplotlib only) produces multiple colorful plots including 3D colored surfaces, contour trajectories, convergence curves, and boxplots, enabling transparent visual comparison.

Future work may include: (i) adaptive phase boundaries based on online diversity measures, (ii) constraint handling via penalty–repair hybrids, (iii) multi-objective extensions using Pareto archives, and (iv) rigorous statistical testing (Wilcoxon/Friedman) across a larger benchmark set [2]–[4], [8].

DOI: 10.46121/pspc.54.2.29

REFERENCES

1. C. H. Wang, “Rethinking metaheuristics: Unveiling the myth of ‘novelty in Metaheuristic Algorithms,” *Mathematics* (MDPI), *13*(13), 2158, 2025.
2. X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, 2nd ed., Luniver Press, 2010.
3. J. M. Dieterich and B. Hartke, “Empirical review of standard benchmark functions using evolutionary global optimization,” ,arXiv:1207-4318v1,18 jul 2012.
4. M.Jamil and X.-S. Yang, “A literature survey of benchmark functions for global optimization,” *International Journal of Mathematical Modelling and Numerical Optimization.* 4 (2), pp. 150-194, 2013.
5. D. H. Wolpert and W. G. Macready, “No Free Lunch Theorems for Optimization,” *IEEE Transactions on Evolutionary Computation*, Vol 1(1),1997.
6. J. Kennedy and R. C. Eberhart, “Particle swarm optimization,” in *Proceedings of the IEEE International Conference on Neural Networks*, Perth, Australia, 1995, pp. 1942–1948.
7. D. Karaboga, *An Idea Based on Honey Bee Swarm for Numerical Optimization*, Technical Report TR06, Erciyes University, 2005.
8. D. Karaboga and B. Akay, “A Comparative Study of Artificial Bee Colony Algorithm,” *Applied Mathematics and Computation*, 214, 108-132,2009.
9. D. Karaboga and B. Basturk, “A powerful and efficient algorithm for numerical function optimization: Artificial Bee Colony (ABC) algorithm,” *Journal of Global Optimization*, 39, 459-471,2007.
10. E. Duman, M. Uysal, and A. F. Alkaya, “Migrating Birds Optimization: A new metaheuristic approach and its performance on the quadratic assignment problem,” Information Sciences,(217), 65-77, 2012.
11. M.Uysal and S.A.Uysal,”SISI-OPT:A Personality-Insperaid Metaheuristic Optimization Algorithm Based on the Characteristics of Empress Elisabeth of Austria”.MSW Management,Vol.36(1),Jan-June 2026,pp 1685-1687.
12. M.Uysal and S.A.Uysal,” MIRO: A Miró-Inspired Metaheuristic Optimization Algorithm,” *MSW Management,*vol 36(1),pp.1840-1843,2026.
13. H. H. Rosenbrock, “An automatic method for finding the greatest or least value of a function,” *The Computer Journal*, vol. 3, pp. 175–184, 1960.
14. L.Ngartera,” A Comparative Study of Optimization Techniques on the Rosenbrock Function”, Open Journal of Optimization Vol. 13(No.3),51-63,2024.
15. L.A.Rastrigin,” Random Search in Problems of Optimization, Identification and Training of Control Systems , “*journal of Sybernetics,* Vol.3(3), 1973.
16. W.Cai,Li Yang et al.” Solution of ackley function based on particle swarm optimization algorithm*,” IEEE International Conference on Advances in Electrical Engineering and Computer Applications(AEECA)*,DOI: 10.1109/AEECA49918.2020,25-27 Aug. 2020.
17. M.Locatelli,” A Note on the Griewank Test Function,”*Journal of Global Optimization,* Vol 25, pages 169–174, 2003.

DOI: 10.46121/pspc.54.2.29