

Received November 12, 2020, accepted November 18, 2020, date of publication December 3, 2020, date of current version December 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3042196

An Adaptive Fuzzy Symbiotic Organisms Search Algorithm and Its Applications

NURUL ASYIKIN ZAINAL , **SAIFUL AZAD** , (Member, IEEE),
AND KAMAL Z. ZAMLI , (Member, IEEE)

Faculty of Computing, College of Computing and Applied Sciences, Universiti Malaysia Pahang, Pekan 26600, Malaysia

Corresponding author: Kamal Z. Zamli (kamalz@ump.edu.my)

This work was supported by the MTUN from the Ministry of Higher Education Malaysia, The Development of T-Way Test Generation Tool for Combinatorial Testing, under Grant UIC19102.


ABSTRACT This paper discusses the development of a Symbiotic Organisms Search Algorithm (SOS) variant, called Adaptive Fuzzy SOS (FSOS). Like SOS, FSOS exploits three types of symbiosis operators namely mutualism, commensalism, and parasitism in order to undertake the search process. Unlike SOS, FSOS is able to adaptively select a single or any combination of mutualism, commensalism, and parasitism update operator(s) as the search progresses based on the current search status controlled by their individual probabilities via the fuzzy decision-making. To validate its performance, we have evaluated FSOS to solve 23 benchmark functions and take a t -way test generation as our case study. Experimental results demonstrate that FSOS exhibits competitive performance against its predecessor (SOS) and other competing metaheuristic algorithms.

INDEX TERMS Search-based software engineering, symbiotic organisms search, computational intelligence.

I. INTRODUCTION

An optimization problem refers to the problem of finding the best solution from a set of candidate solutions. Generally, an optimization algorithm exploits a mathematical function, called the objective function, which is often an extremum (either maximum or minimum) function for finding an optimal solution while satisfying a set of given constraints [1]–[4].

Generally, optimization problems can be solved using either exact or heuristic methods. Here, an exact method often offers the best optimal solution for an optimization problem. However, for many practical optimization problems, they are non-scalable; especially when the search spaces are large. For those problems, often heuristic methods are preferred owing to the resource and timing constraints since the effort grows exponentially with the increasing problem size. Unlike an exact method, a heuristic method can find a good enough solution in polynomial time but does not guarantee optimality. However, they lack generality and often problem-specific

The associate editor coordinating the review of this manuscript and approving it for publication was Shun-Feng Su .

because of the fact that they exploit the properties of a given problem while discovering an optimum solution.

To resolve this issue, many higher-level heuristics, called metaheuristics, have been proposed in the literature during the past 30 years. Apart from functioning at a high-level abstraction than that of heuristic methods, metaheuristics judiciously provide a mechanism for balancing between the exploration (roaming around all over the potential search space) and the exploitation (making use of the known local best) in order to tackle a wider range of optimization problems.

Typically, the balance between exploration and exploitation is achieved through adjusting control parameters. For instance, Particle Swarm Optimization (PSO) [5] algorithm exploits three parameters, namely inertia weight, social, and cognitive in order to decide whether to explore or to exploit. Similarly, the Genetic Algorithm (GA) [6] employs mutation and crossover rates to decide its course of searching for action. However, calibrating these parameters manually is time consuming and a delicate process since there is no one size fits for all approaches.

Recently, researchers have come out with a family of parameter-free metaheuristics such as Teaching Learning

Based Optimization (TLBO) [7], Jaya Algorithm [8], and Symbiotic Organism Search (SOS) [9] which rule out this tuning need. Although helpful, parameter-free metaheuristics often rely on a pre-set sequence for exploration and exploitation. More precisely, parameter-free metaheuristics iterate the exploration and exploitation update operator in a deterministic sequence of explore-exploit-explore-exploit until the maximum iteration is reached. Consider an instance where the search process is near-convergence. In this case, repeatable adoption of local search operator is preferable to obtain an optimal solution. Due to pre-set sequence, the adoption of the global search operator, instead, would move the current solution away from the optimal solution. As such, converge to optimal solution can be potentially hindered.

Addressing these issues, we propose a new adaptive technique in this paper, which decides whether to perform exploration or exploitation within the context of a recently developed parameter-free metaheuristic algorithm, called Symbiotic Optimization Search (SOS). In the current form, SOS splits its searching operations into three phases in accordance to three update operators (namely mutualism, commensalism, and parasitism) and executes them in sequence. However, having such a deterministic sequence can be counter-productive as the search process is not sensitive to its current search requirements and hence, could not be prescribed in advance. The proposed adaptive Fuzzy based SOS variant, called Fuzzy Symbiotic Optimization Search (FSOS) integrates the Mamdani Fuzzy technique to overcome this issue. The newly proposed variant is able to select a single or any combination of update operators among mutualism, commensalism, and parasitism adaptively based on the setting of their individual probabilities via a fuzzy decision-making process. Our contributions in this paper can be summarized as follows:

- A new SOS variant called FSOS, which integrates the Mamdani fuzzy inference system to permit adaptive controls between the exploration and exploitation operations by juggling among three update operators of SOS, namely mutualism, commensalism, and parasitism.
- An extensive experimental evaluation is performed for FSOS involving 23 benchmark functions as well as taking a *t*-way test suite generation as a case study and comparing with other analogous metaheuristics.

The rest of the paper is organized as follows. Section II investigates the existing techniques in brief. For a better understanding of the proposed technique, in Section III, the SOS and its variants are discussed elaborately. The proposed technique is explained in Section IV with adequate details. Section V and VI describe our experimental setup and analysis of the acquired results, respectively. Finally, this paper ends with a concluding remark in Section VII.

II. RELATED WORKS

The metaheuristic algorithms can be broadly classified into three different classes based on their sources of inspiration, namely nature-based, physics-based, and human-based.

As the name suggests, nature-based metaheuristics are inspired by several activities of natural living things, including animals and birds, and update their operators and manipulate their solutions by mimicking the collective information sharing strategies of these living things. For instance, the Firefly Algorithm (FFA) [10] mimics the strategy of how a firefly shares its location information by flashing the bioluminescence to attract other fireflies. Again, the location sharing strategy of an alpha wolf in the Grey Wolf Optimizer (GWO) [11] imitates the hunting techniques of a grey wolf. Furthermore, Particle Swarm Optimization (PSO) [5] imitates the path of flocking birds. Other examples include Mayfly Optimization Algorithm (MFA) [12], Flower Pollination Algorithm (FPA) [13], Symbiotic Organism Search (SOS) [9], Laying Chicken Algorithm (LCA) [14], Chicken Swarm Optimization (CSO) [15], Whale Optimization Algorithm (WOA) [16], and Artificial Bee Colony (ABC) [17].

Physics-based metaheuristics are conceptualized following the real-world physical phenomena. For instance, a number of metaheuristics in this class are designed based on observation on the state of physics, including annealing state in metal work (Simulated Annealing [18]), fluid flow state (Flow Regime Algorithm [19]), rain droplets state (Rain Optimization Algorithm [20]), and water-dropping state (Intelligent Water Drops [21]). Again, another set of metaheuristics are designed based on the law of physics including Central Force Optimization [22] (Gravitational law) and Charged System Search [23] (Coulomb's law, Gauss's law). Furthermore, metaheuristics like Quantum Genetic Algorithm [24], Black Hole Algorithm [25], and Quantum Inspired Social Evolution [26] are designed based on quantum physics.

Human-based metaheuristics are inspired by the interaction of human beings and their surrounding biological activities. The most profound algorithm designed following this phenomenon includes Genetic Algorithms (GA) [6], which is motivated by Darwin's Theory of human evolution. Among the other algorithms in this class, most of them endeavor to emulate human intelligence of problem solving for discovering better solutions. A few such algorithms are: Harmony Search [27] (music composition), Brain Storm Optimization [28] (brainstorming sessions), Teacher-Learning Based Optimization [7] (teaching methods), Interactive Autodidactic school (learning methods) [29], Student Psychology Based Optimization (student psychology) [30], and Search Group Algorithm [31] (human search methods).

Despite having different sources of inspiration, almost all metaheuristics take advantages of the exploration and exploitation capabilities for performing the search process, which is briefly highlighted previously in Section I. Balancing these capabilities is crucial as extensive exploration consumes resources; conversely, extensive exploitation potentially risks the search process (traps in local optima) and hinder convergence. By judiciously adjusting their control parameters, metaheuristics can balance their biases towards exploration (global search) or exploitation (local search) activities. However, tuning these control parameters is often

problem dependent and can sometimes be cumbersome as improper tuning may lead to poor performance. Addressing these issues, several metaheuristics are proposed recently that advocate parameter-free approach, including Interactive Autodidactic School (IAS) [29], Teaching-Learning Based Optimization (TLBO) [7], and Symbiotic Organism Search (SOS) [9].

Taking this new approach into consideration, existing metaheuristics can also be classified as either parameterized or parameter-free metaheuristics. From this perspective, a metaheuristic can be nature-based but also be a parameter-free algorithm (e.g. SOS). In the same manner, a metaheuristic can also be human-based but also be a parameter-free algorithm (e.g. TLBO). Generally, parameter-free metaheuristics often employ a preset sequence of the search process, e.g., explore-exploit-explore-exploit until the iteration threshold is reached. However, heedlessly following such a deterministic sequence can be counterproductive given the fact that exploration and exploitation are dynamic in nature. Again, any such deterministic sequence can lead to a stagnation problem, which in turn, increases the probability of trapping the search process into local optima. Therefore, to overcome these issues, in this current work, an adaptive Fuzzy based SOS variant or FSOS is proposed. Exploiting the Mamdani fuzzy inference system, FSOS is able to select a single or any combination of mutualism, commensalism, and parasitism update operator(s) adaptively based on the fuzzy decision-making process.

III. ORIGINAL SOS AND ITS VARIANTS

The following section highlights the main features of SOS and its proposed variants.

A. ORIGINAL SOS

Inspired by the interrelationship of the organisms in nature, Cheng and Prayogo developed a population based parameter-free metaheuristic algorithm, called SOS [9]. The main feature of SOS is the fact that it integrates three key search phases, which are inspired from the three basic types of symbiosis from mutualism, commensalism, and parasitism as mentioned earlier in Section I.

The mutualism phase mimics the interactions of two organisms that receive the mutual benefits of living together and thus, increasing their chances of survival in the ecosystem via complementing each other. Therefore, candidate solutions are calculated by measuring the differences between the best solution and the average of two organisms, called *Mutual_Vector*, which is calculated as in Eq. 1:

$$Mutual_Vector = \frac{X_i + X_j}{2} \quad (1)$$

where, X_i and X_j are two organisms in interactions. The intuition behind such an average is that when interactions between two organisms are far, they will produce unique solutions and thereby, enabling the SOS to explore new search spaces. Afterwards, both organisms are updated

simultaneously with two random values, namely BF_1 and BF_2 (taking the value of either 1 or 2) as in Eq. 2 and Eq. 3:

$$X_{i_{new}} = X_i + rand(0, 1) * (X_{best} - Mutual_Vector * BF_1) \quad (2)$$

where $X_{i_{new}}$ be the new value for X_i and X_{best} be the current best solution.

$$X_{j_{new}} = X_j + rand(0, 1) * (X_{best} - Mutual_Vector * BF_2) \quad (3)$$

where $X_{j_{new}}$ be the new value for X_j and X_{best} be the current best solution.

In the commensalism phase, among the two interacting organisms, only one receives the benefits of interaction while the other remains unaffected. To be specific, assuming X_j be the passive receiver and X_i be the active receiver, X_i receives the benefits from X_j and is enhanced by Eq. 4:

$$X_{i_{new}} = X_i + rand(-1, 1) * (X_{best} - X_j) \quad (4)$$

where $X_{i_{new}}$ be the new value for X_i and X_{best} be the current best solution, which is utilized as a reference point so that the new candidate solution will exploit around that region only.

In the parasitism phase, only one organism survives while the other is killed — either the parasite itself or the host. Here, parasitism operator duplicates one random organism and modify its characteristics randomly to act as parasite. This is denoted as $X_{Parasite_Vector}$, as in Eq. 5:

$$X_{Parasite_Vector} = rand(lb, ub) \quad (5)$$

where, lb be the lower bound, and ub be the upper bound. Now, when parasite is compared with X_j and the latter is found to have better fitness, then, X_j is assumed to be immune from the parasite. Hence, the parasite is discarded or killed. Conversely, if the parasite has better fitness, then the host, X_j is discarded or killed and the former undertakes its position.

These three phases are repeated sequentially in every iteration until the stopping condition is met as demonstrated in Algorithm 1. For a greater insight, a flow chart of SOS is also presented in Figure 1.

Algorithm 1 The Original SOS Algorithm

Input : the population $X = X_1, X_2, \dots, X_D$,

Output : X_{best} and the updated population $X = X_1, X_2, \dots, X_D$,

1. Population Initialization
 2. **while** stopping criteria not met **do**
 3. Mutualism Phase
 4. Commensalism Phase
 5. Parasitism Phase
 6. Get best result X_{best}
 7. **end while**
-

B. REVIEW OF SOS VARIANTS

Owing to its simplicity, SOS has been adopted in many optimization problems, including optimizing steel rigid frame design [32], minimizing real power losses [33], and optimizing the scheduling problems [34]–[36]. Again, to cater

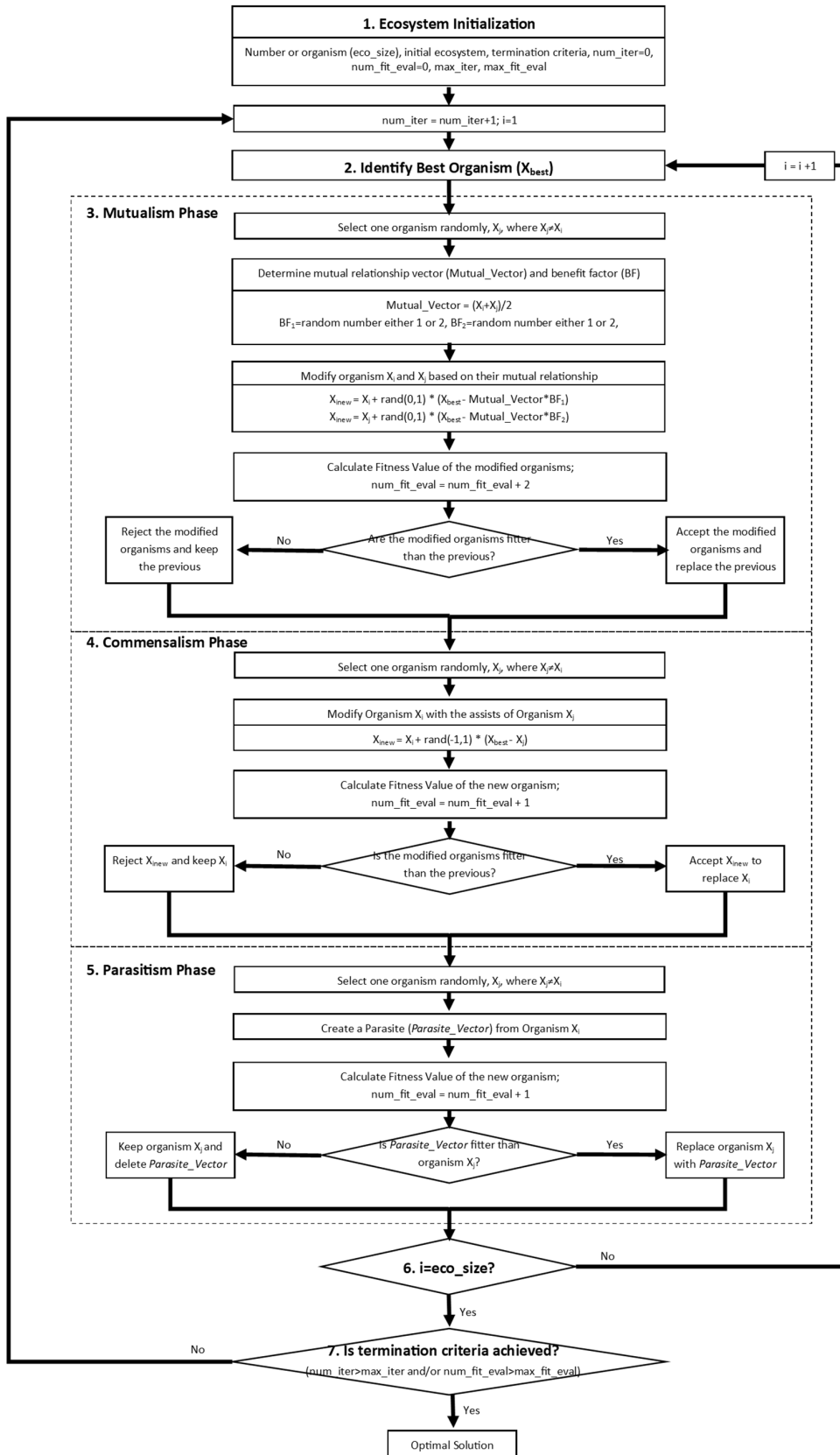


FIGURE 1. Flow chart of the SOS algorithm.

to the needs of its application, many variants of SOS have been proposed in the literature. According to Ezugwu and Prayogo [37], these variants can be classified as discrete SOS, adaptive SOS, modified SOS, and multi-objective SOS. As the name suggests, discrete SOS refers to the variants that solve discrete optimization problems. For instance, the Discrete SOS algorithm (DSOS) [36] is such an algorithm, which is developed by the SOS author himself to solve scheduling problems by reducing fluctuations in terms of resource usages. Here, a discrete solution is achieved by transforming the continuous solution spaces of SOS into discrete solution spaces. Other variants are in [33], authors introduce a fix function for rounding the solutions into the nearest integers approaching zero and in [38], a hybrid operator is introduced to solve the traveling salesman problem.

Recently, adaptive SOS emerged as the need for controlling exploration and exploitation activities dynamically during the runtime. To-date, much recent work on adaptive SOS has revolved around manipulating the benefit factors. In [39], [40], Tejani et al exploit one of the mutualism benefit factor (BF1) with an adaptive variant (ABF1), that uses the fitness ratio of the current fitness over the best fitness. In other work, the same author in [41], [42] also explores the possibility of exploiting both the mutualism benefit factors with ABF1 and ABF2 using the same ratio. Analogous to this work, our proposed work employs the concept of adaptive factors; however, our adaptive factors are not built-in parameters like ABF1 and ABF2. Due to these built-in parameters, the applicability of the algorithm can be restricted to only certain optimization problems. On the contrary, FSOS can be applied to almost any optimization problem due to the fact that it is the FIS — not any built-in parameters— that performs the selection of the operators.

Modified SOS refers to those variants that modify its parameters and/or combine with other metaheuristics. For instance, a piecewise linear chaotic map based on Chaotic Local Search (PWLCM-CLS) is embedded with SOS to make the convergence faster [43]. Here, the effectiveness of the chaotic maps depends on the search space topology of a particular optimization problem at hand. In another work [44], SOS has been modified by restricting the random values of the commensalism phase in the range of [0.4,0.9]. On a positive note, the restriction has improved the convergence speed of SOS. However, on a negative note, such restriction potentially increases the possibilities of trapping in local as the range for exploration is capped. In another work in [45], the parasitism phase of SOS has also been modified to re-initialize partial population based on the given percentage of the maximum fitness evaluation in order to enhance its local search performance. There are another class of variants where SOS is combined with other algorithms, including Simulated Annealing [34], [46] and Quasi-Oppositional Based Learning [47]. The combination of two or more metaheuristics results in a better performance when integrated algorithms complement each other and thus, heighten their strength; otherwise, may lead to poor performance. Again,

these combinations may make SOS parameterized metaheuristics, which require extensive calibrating of the control parameters.

Finally, to solve multi-objective problems, many variants of multi-objective SOS also have emerged, including Opposition Multiple Objective Symbiotic Organisms Search (OMOSOS) [48], Chaotic Symbiotic Organisms Search (CMSOS) [35], and Multi-Objective modified adaptive symbiotic organisms search (MOMASOS) [49]. These variants provide solutions for the problems with multiple criteria decisions making and require a number of trade-offs between two or more conflicting objectives.

IV. FUZZY SOS

The proposed FSOS is based on the Mamdani fuzzy interference system as demonstrated in Figure 2 with three normalized performance inputs. More precisely, the performances of mutualism, commensalism, and parasitism phases are assessed after each iteration to measure the productiveness of each phase. Later, these measurements are utilized in deriving these probabilities, namely p_1 , p_2 , and p_3 , which are afterwards utilized in selecting mutualism, commensalism, and parasitism, respectively.

A. SEARCH PERFORMANCE MEASUREMENT MODEL

Three criteria are used for the performance measurement after completing every iteration based on Normalized Performance Evaluation (NPE), Fail Success Rate (FSR), and Overall Success Rate (OSR).

The Normalized Performance Evaluation (NPE) is the normalized measure of the delta difference between the current fitness value against the best solution (as proposed by Shalabi, *et al.* [50]). NPE can be calculated as in Eq. 6:

$$NPE = \frac{(F[xi] - Min_a)}{(Max_a - Min_a)} \times (NewMax_a - NewMin_a) + NewMin_a \times 100 \quad (6)$$

where $F[xi]$ is the current value, Min_a is the minimum value in the data set, Max_a is the maximum value in the dataset, $NewMin_a$ is the new minimum value in the data set, and $NewMax_a$ is the new maximum value in the dataset. Here, a low NPE value indicates that the organisms are placed closely and started to converge towards the best solution or fall into local optima. A high NPE indicates that the search is still in the exploring mode.

Fail Success Rate (FSR) represents the percentage of the ratio of failure and success. FSR can be calculated as in Eq. 7:

$$FSR = \frac{FailCount}{SuccessCount} \times 100 \quad (7)$$

where *FailCount* is the number of non-improved objective values in a particular iteration and *SuccessCount* is the number of improved objective value in a particular iteration. As can be observed from Eq. 7, a low FSR value indicates the percentage of high success while a large FSR value indicates that the solutions are not improving by the involved phase.

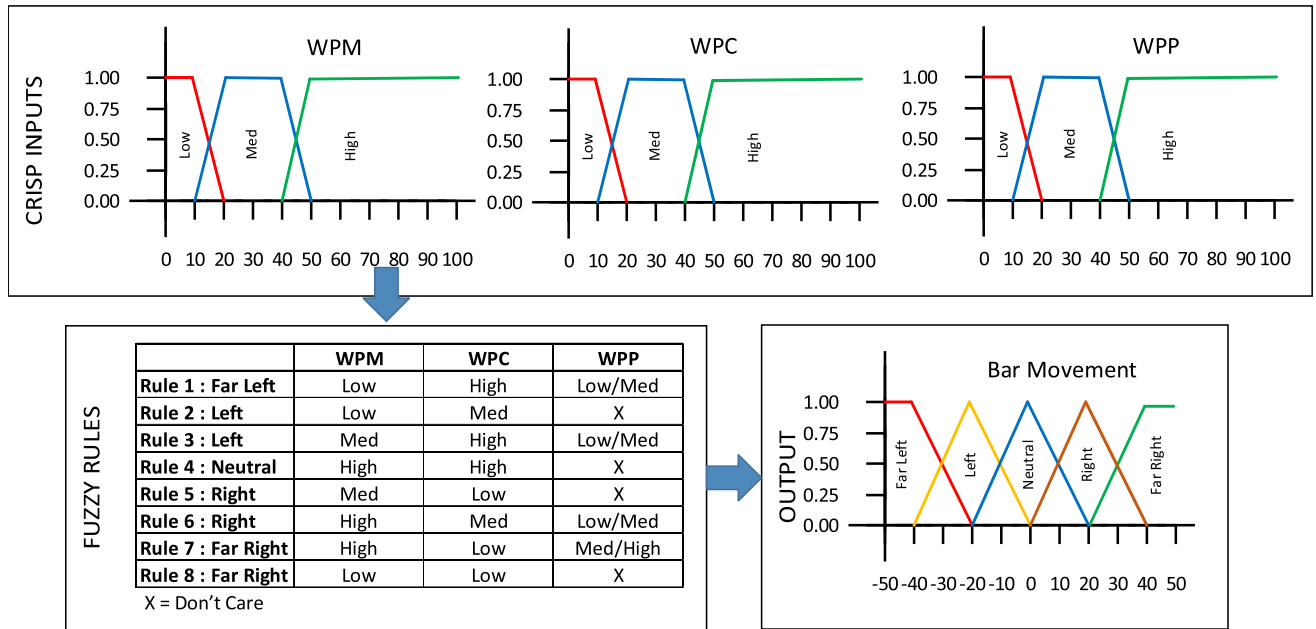


FIGURE 2. Fuzzy Input-Output and Membership Rules.

Overall Success Rate (OSR) complements the FSR by measuring the percentage of the ratio of the number of improved objective values against the population size, which can be calculated as in Eq. 8:

$$OSR = \frac{SuccessCount}{EcoSize} \times 100 \quad (8)$$

where *SuccessCount* holds the same definition as in Eq. 7 and *EcoSize* is the population size. According to Eq. 8, a high OSR value indicates good performance and a low OSR value indicates otherwise.

Since the aforementioned performance measures (NPE, OSR, and FSR) are intertwined and interrelated; therefore, we develop three equations to find the weighted performance of them for each phase, called Weighted Performance Mutualism (WPM), Weighted Performance Commensalism (WPC), and Weighted Performance Parasitism (WPP). These weighted performances can be calculated as in Eq. 9 till Eq. 11.

$$WPM(Mutualism) = \alpha NPE + \beta OSR + \gamma(100 - FSR) \quad (9)$$

$$WPC(Commensalism) = \alpha NPE + \beta OSR + \gamma(100 - FSR) \quad (10)$$

$$WPP(Parasitism) = \alpha NPE + \beta OSR + \gamma(100 - FSR) \quad (11)$$

where, α , β , and γ are adaptively calculated during each iteration as in Eq.12 till Eq. 14:

$$\alpha = (1 - \gamma)/2 \quad (12)$$

$$\beta = (1 - \gamma)/2 \quad (13)$$

$$\gamma = \frac{Number\ of\ Fitness\ Evaluation}{Maximum\ Fitness\ Evaluation} \quad (14)$$

Here, the maximum value of γ is capped at 0.5 to avoid any large imbalanced contribution to the weighted performance measures.

B. FUZZY INFERENCE SYSTEM (FIS)

As mentioned earlier, the proposed FSOS is based on the Mamdani FIS with 3 crisp inputs, namely WPM, WPC, and WPP and one output - called Bar Movement. Fig. 3 highlights the overall fuzzy inference system for FSOS.

As fuzzy rules can be expressed as linguistic constraints that are easy to understand and maintain, Mamdani inference is often preferred over Sugeno [51]. Furthermore, we opt for triangular/trapezoidal based membership functions over Gaussian type membership owing to its superior performance [52]. Based on three linguistic terms of low, medium, and high, all weighted performance measures (WPM, WPP, and WPC) share the identical membership functions defined as: low (0-10), partial low, and medium (10-20), absolute medium (20-40), partial medium, and high (40-50) and absolute high (above 50) respectively.

Meanwhile, the following 8 fuzzy rules are used in the fuzzification process:

Rule 1: When the performance of the commensalism phase is high while the mutualism phase experiences low performance and the parasitism phase experiences moderate/low performance, the search is converging, and therefore, requires more exploitations.

Rule 2: When the performance of the commensalism phase is medium while the mutualism receives low performance regardless of the performance of the parasitism phase, the search needs a few exploitations.

Rule 3: When the performance of the commensalism phase is high while the mutualism phase is performing

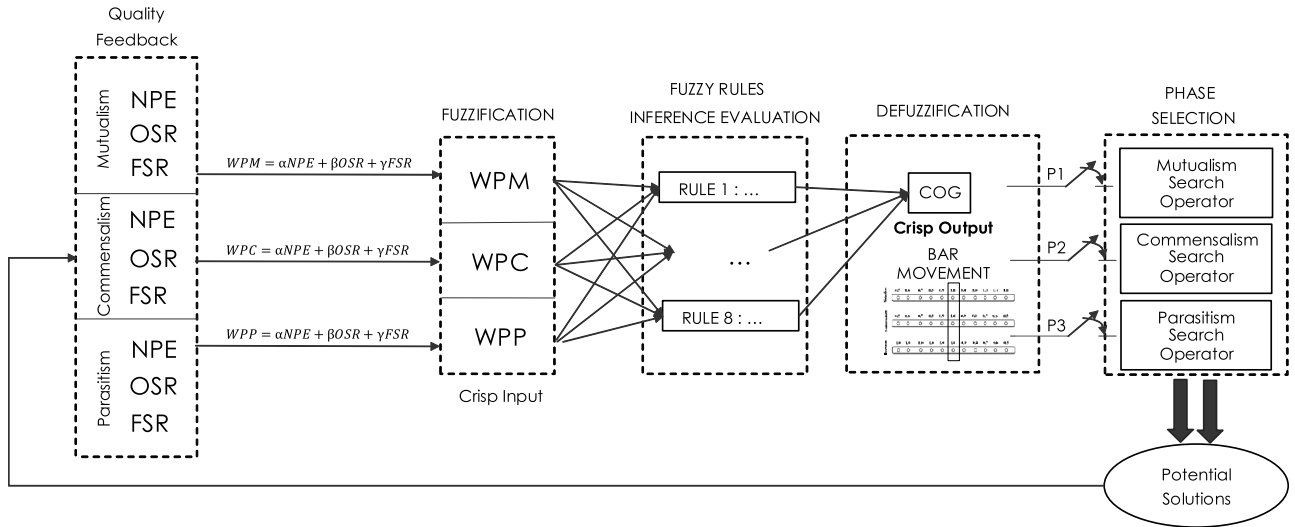


FIGURE 3. Fuzzy Inference System for FSOS.

moderately and the parasitism phase is performing moderate/low, the search almost nearing a possible solution, and therefore, requires a moderate number of exploitations with a few explorations to avoid being trapped into local optima.

Rule 4: When the performances of all the phases are high, it can be assumed that all phases are still performing, and therefore, the search needs to continue exploration and exploitation activities.

Rule 5: When the performance of the mutualism phase is medium while the commensalism phase receives low performance, the search still not nearing any possible solution, and therefore, requires more explorations with a few exploitations.

Rule 6: When the performance of the mutualism phase is high while the commensalism phase is performing moderately and the parasitism phase is performing moderately/highly, the possibility of discovering an optimum solution increases, and therefore, requires a moderate amount of explorations with a few exploitations to inquire the possible solutions.

Rule 7: When the performance of the mutualism phase is high while the parasitism phase experiences low performance and the commensalism phase experiences moderate/low performance, the search is still not nearing any possible solution, and therefore, requires more explorations.

Rule 8: When the performances of all the phases are low and no noticeable phase performance is observed, the search needs more explorations as it might be trapped into local optima.

A single output, called bar movement value is defined for defuzzification. Triangular membership functions are chosen as the Bar Movement defuzzification with five overlapping (and equal-width) linguistic terms. Concerning overlapping, we start the overlapping at their center points

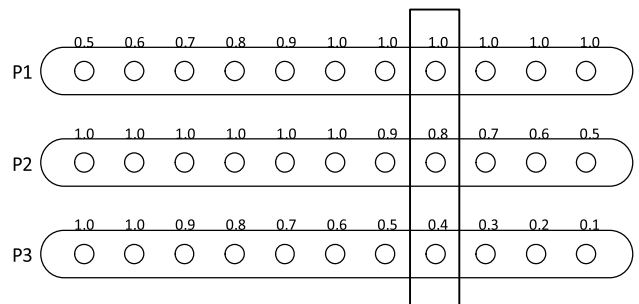


FIGURE 4. Bar Movement after Defuzzification Process.

using middle intensification (MIDI) and middle weakening (MIDW), where the performance of the fuzzy system is at best as proposed by Mizumoto [53]. Given the aforementioned design choices, the selection of the output will be based on the center of gravity in the range of -50 to 50 .

Taking inspiration from the work of Cheng and Prayogo [54], a bar controller is employed for self-adjusting the probability of executing mutualism, commensalism, and parasitism in the next iteration phase. For instance, if the bar movement result is $+20$, then the bar will move 0.20 steps to the right from the center. This can be translated as the probability of executing the mutualism phase in the next iteration is 100% ($P1 = 1.0$), commensalism phase is 80% ($P2 = 0.8$) and parasitism phase is 40% ($P3 = 0.4$), respectively. Fig. 4 demonstrates a bar movement after the defuzzification process.

C. FSOS IMPLEMENTATION

The complete pseudo-code of the FSOS algorithm is presented in Algorithm 2. In this algorithm, line 3 defines the FIS input-output membership rules after defining the required data structures in line 1 and 2. Afterward, a random population is initialized and their fitness values are calculated to discover the initial best solution.

Line 5 marks the start of the search process to find the best solution, X_{best} . Three random numbers are generated at the beginning of every iteration, namely $Rn1$, $Rn2$, and $Rn3$ accordance with mutualism, commensalism, and parasitism phases, respectively within the range of [0,1].

For enforcing the execution and evaluation of the mutualism, commensalism, and parasitism phases, in the first iteration, the values of P1, P2, and P3 are set to 1. From the subsequent iteration, this restriction is lifted and individual phases are selected based on their respective random numbers, which are mentioned earlier. For instance, if $Rn1 < P1$, the mutualism phase is selected (see line 7); whereas, the commensalism phase is selected if $Rn2 < P2$ (see line 12), and the parasitism phase is selected if $Rn3 < P3$ (see line 17).

Algorithm 2 The General Adaptive SOS Algorithm Based on Fuzzy Inference System

Input : the population $X = X_1, X_2, \dots, X_n$

Output : X_{best} and the final population $X_f = X_1^f, X_2^f, \dots, X_n^f$

1. Define population size, max fitness evaluation
 2. Define of member functions for the linguistic variables
 3. Define fuzzy rules
 4. Initialize populations and calculation X_{best}
 5. **while true do**
 6. Generate 3 random number, $Rn1$, $Rn2$ and $Rn3$
 7. **if** $Rn1 < P1$ **then**
 8. **for** $i = 1$ **to** *population size* **do**
 9. Mutualism Phase
 10. **end**
 11. **end**
 12. **if** $Rn2 < P2$ **then**
 13. **for** $i = 1$ **to** *population size* **do**
 14. Commensalism Phase
 15. **end**
 16. **end**
 17. **if** $Rn3 < P3$ **then**
 18. **for** $i = 1$ **to** *population size* **do**
 19. Parasitism Phase
 20. **end**
 21. **end if**
 22. Calculate NPE, FSR, and OSR for each phase
 23. Calculate α , β , and γ
 24. Calculate WPM, WPC, and WPP
 25. Fuzzify based on WPM, WPC, and WPP
 26. Defuzzify and use bar movement for the next P1, P2, and P3
 27. Get best result X_{best}
 28. **if** *fitness evaluation* \geq *maximum fitness evaluation* **then**
 29. **exit from loop**
 30. **end**
 31. **end**
 32. **return** X_{best} and X_f
-

After executing these phases, their performances are measured in terms of NPE, FSR, and OSR according to Eq. 6, 7, and 8 respectively as mentioned in line 22. Again, their weighted performances are calculated in terms of WPM, WPC, and WPP using Eq. 9, 10, and 11, respectively (see line 24). These weighted measures are fed into the FIS to produce the bar movement output to acquire the probabilities of P1, P2, and P3 for the next iteration (see lines 23-26). In line 27, X_{best} is updated for every iteration. This process continues until either of the following stopping conditions is met: the maximum number of evaluations have been completed or the max fitness evaluation is reached (see lines 28-29).

V. FSOS EVALUATION

For evaluating the performance of the FSOS in a comprehensive manner, two separate sets of experiments are performed for: i) benchmarking its performance against other analogous metaheuristics and ii) identifying its performance in solving combinatorial test suite generation problems, which are detailed in Section A and B, respectively. As far as our computing platform for conducting all these experiments is concerned, we have adopted a PC running Windows 10 with the specifications as depicted in Table 1. All the results reported in this paper are acquired after 30 independent runs. The source code of our implementation is available upon sending a request to the corresponding author.

A. BENCHMARKING WITH OTHER METAHEURISTICS

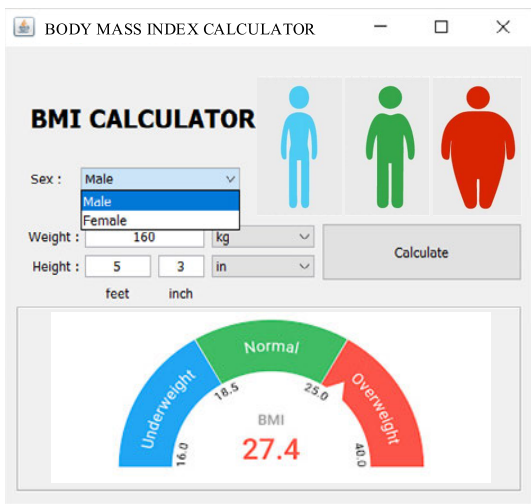
The first set of experiments takes 23 standard benchmark functions into consideration, where 7 among them are unimodal functions, 6 are multimodal, and the rest are fixed-dimension multimodal functions. The performance of the proposed algorithm, FSOS is benchmarked against its predecessor, SOS and other parameter-free metaheuristics, including Jaya [8], TLBO [7], Sine Cosine Algorithm (SCA) [55], and Chaotic Fruit fly Optimization Algorithm (CFOA) [56]. For ensuring a fair comparison during the experimental evaluations, the following actions are taken into considerations: only parameter-free metaheuristics are chosen since FSOS falls into that class, the population size is fixed to 30 for all of the experiments, unless otherwise explicitly mentioned, and the maximum fitness function evaluation is set to 50,000 for all the compared algorithms. The result of less than 10^{-250} will be considered as zero.

B. COMBINATORIAL TEST SUITE GENERATION

The second set of experiments deals with a case study involving a combinatorial test suite generation problem. Considering NP-complete, the combinatorial test suite generation is an optimization problem with the aim of generating the minimum t -way interactions, where t represents the interaction strength. The rationale behind t -way combinatorial testing is that not every parameter contributes to every fault, and many faults can be exposed by interactions involving only a few parameters [57].

TABLE 1. Hardware and Software Requirements.

Hardware	Description
Processor	CPU 2.9 GHz Intel Core I5
Memory	16 GB 1867 MHz DDR3 RAM
Hard Disk Space	512 of flash HDD
Software	Description
Operating System	Windows 10
Language	Java 1.8.0



	Parameters		
	P1 - Sex	P2 - Height	P3 - Weight
Values	0 - Male	0 - Inch	0 - Pound
	1 - Female	1 - Centimetre	1 - Kilogram

Test No.	P1	P2	P3
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

Test No.	P1	P2	P3
1	1	1	1
2	1	0	0
3	0	0	1
4	0	1	0

Exhaustive Testing

2-Way Test Suite

FIGURE 5. BMI Calculator System.

Considering a hypothetical BMI calculator with 3 parameters, namely sex, height, and weight and with 2 values each, a test suite generation problem is designed in this paper as depicted in Fig. 5. If, for a system with k parameters with v values each, the number of all possible combinations is v^k ; in this manner, there are 8 or 2^3 possible exhaustive combinations in our example. By using the interaction strength ($t = 2$), the test cases needed for the BMI calculator can be reduced to 4 tests only.

Mathematically, the t -way test suite generation problem is often associated with covering array (CA). Assuming aforementioned notations for a uniform value, a t -way test suite generation problem can be written as, $CA(N; t, v^k)$ where N is the test cases with t interaction strength for k number of parameters with v values. When $t = 2, k = 4,$ and $v = 3,$ the aforementioned expression can be rewritten as: $CA(N; 2, 3^4)$, which represents a 2-way test generation for a system of 4 parameters with 3 values each. Similarly, $CA(N; 3, 4^6)$ refers to a 3-way test suite generation of 6 parameters with 4 values each. Conversely, when the values of v are non-uniform, we called the test suite as a Mixed covering Array (MCA), where an expression, $MCA(N; 2, 5^1 3^8 2^2)$ denotes the 2-way test suite generation

for 1 parameter with 5 values, 8 parameters with 3 values, and 2 parameters with 2 values.

In our experiments, 16 different configurations combining both CA and MCA are taken into account similar to that of the earlier work in [58]: $CA(N; 2, 3^4), CA(N; 2, 3^{13}), CA(N; 2, 10^{10}), CA(N; 2, 15^{10}), CA(N; 2, 5^{10}), CA(N, 2, 8^{10}), CA(N; 3, 3^6), CA(N; 3, 4^6), CA(N; 3, 5^6), CA(N; 3, 6^6), CA(N; 3, 5^7), CA(N; 3, 10^6), MCA(N; 2, 5^1 3^8 2^2), MCA(N; 2, 7^1 6^1 5^1 4^6 3^8 2^3), MCA(N; 3, 5^2 4^2 3^2), MCA(N; 3, 10^1 6^2 4^3 3^1).$

Analogous to the first set of experiments, FSOS is also compared against its predecessor, SOS and other parameter-free metaheuristics (see Section V-A). However, unlike the first set of experiments, the population size for all the considered algorithms is set to 10 with maximum of 5000 fitness function evaluations.

C. STATISTICAL ANALYSIS

The performance of FSOS will be analyzed using Wilcoxon's signed ranked test [59] against other algorithms. The absolute values of the difference between the mean scores of the two algorithms d_i , for i th out of n solutions, will be ranked. In the case of ties, the average ranking of the result will be applied [60]. Referring to Eq. 15, R+ will be the sum of ranks for the outperformed solutions of the FSOS with other

TABLE 2. Description of Unimodal Benchmark Functions.

		Dim	Range	f_{min}
Sphere	$F_1(x) = \sum_{i=1}^n x_i^2$	30	[-100,100]	0
Schwefel 2.22	$F_2(x) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i $	30	[-10,10]	0
Schwefel 1.2	$F_3(x) = \sum_{i=1}^n \left(\sum_{j=1}^i x_j \right)^2$	30	[-100,100]	0
Schwefel 2.21	$F_4(x) = \max_i \{ x_i , 1 \leq i \leq n\}$	30	[-100,100]	0
Rosenbrock	$F_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30,30]	0
Step	$F_6(x) = \sum_{i=1}^n ([x_i + 0.5])^2$	30	[-100,100]	0
Quartic with Noise	$F_7(x) = \sum_{i=1}^n ix_i^4 + random[0,1]$	30	[-100,100]	0

tested algorithms respectively, whereas, the underperformed solutions are denoted by R- as in Eq. 16. When the difference of $d_i = 0$, the rank will be divided equally between the sums. If an odd number of them is present, one is ignored:

$$R^+ = \sum_{d_i > 0} rank(d_i) + \frac{1}{2} \sum_{d_i = 0} rank(d_i) \quad (15)$$

$$R^- = \sum_{d_i > 0} rank(d_i) + \frac{1}{2} \sum_{d_i = 0} rank(d_i) \quad (16)$$

When $T = \min(R+, R-)$ is less or equal to the value of the distribution of Wilcoxon for n degrees of freedom [61], the null hypothesis of equality of means is rejected. This means that the proposed algorithm outperforms the other one.

VI. RESULTS AND DISCUSSIONS

In accordance with the experimental setup, the results are also discussed in two different sections as below:

A. BENCHMARKING TEST RESULTS

The acquired results of the FSOS for 23 benchmark functions are presented in Table 5 (for unimodal functions), Table 6 (for multimodal functions), and Table 7 (for fixed-dimension multimodal functions) alongside other compared parameter-free metaheuristic algorithms. The last three rows in the tables summarize the other results presented there, which are: no. of global minimum found, average ranking, and overall ranking. Here, the first measure is counted by comparing a mean value of a function with the value mentioned in the ‘‘Min’’ (representing the global minimum solution) column for that respective functions for all the compared algorithms. The count is only increased when a match is found. Herein, the mean is taken into account over the best value for eliminating

the possibility of discovering an optimum solution by chance. Again, average rankings are calculated by taking the ranks of the functions into consideration where the ranks are determined by sorting the means. Similarly, the overall rankings of the compared metaheuristics are determined by sorting the average rankings.

1) UNIMODAL FUNCTIONS

Referring to the experimental results of the unimodal functions in Table 5, FSOS outperforms other contenders and secures the 1st position followed by SOS, TLBO, SCA, Jaya, and CFOA. The reason that SOS performs better than most of the other compared algorithms (except FSOS) owing to the improved balance between the exploitation and exploration activities. As pointed out earlier in Section III that the mutualism and commensalism phases in SOS enable the search procedure to discover diverse solutions in the search space, and thereby, improving the exploration ability of the algorithm. On the other hand, the parasitic phase enables the search procedure to exploit the current best and thus, improving the exploitation ability of the algorithm. A close competitor of SOS in terms of ranking is TLBO. Similar to SOS, TLBO presets its search operator sequence from teacher phase for exploration (based on global best) and student phase for exploitation (based on peer values). Unlike SOS, TLBO does not provide a fine grain local search operator resembling the mutation-based SOS parasitism operator. For this reason, in some cases, TLBO performs poorly for some benchmark functions.

With only one search operator, Jaya performs poorly. The same observation is also true for SCA and CFOA. Therefore, all these three contenders along with TLBO score 0 as far

TABLE 3. Description of Multimodal Benchmark Functions.

	Function	Dim	Range	f_{min}
Schwefel's 2.26	$F_8(x) = \sum_{i=1}^n -x_i \sin(\sqrt{ x_i })$	30	[-500,500]	-12569.5
Rastrigin	$F_9(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$	30	[-5.12,5.12]	0
Ackley	$F_{10}(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i)\right) + 20 + \exp(1)$	30	[-32,32]	0
Griewank	$F_{11}(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[-600,600]	0
Generalized Penalized 01	$F_{12}(x) = \frac{\pi}{n} \left\{ 10 \sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_n - 1)^2 \right\}$ $+ \sum_{i=1}^n u(x_i, a, k, m)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$ $a=5, k=100, m=4$	30	[-50,50]	0
Generalized Penalized 02	$F_{13}(x) = 0.1 \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)] \right\}$ $+ \sum_{i=1}^n u(x_i, a, k, m)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$ $a=5, k=100, m=4$	30	[-50,50]	0

as discovering global minimum (see Table 5) for all the unimodal functions. On the other hand, FSOS and SOS discover 2 global minimum solutions for F1 and F3 respectively. Side by side, a comparison of FSOS and SOS demonstrates the superiority of the former in terms of discovering optimum solutions for more functions. FSOS obtain the first rank for almost all individual functions except F6; whereas, SOS ranked 1 for only two functions, namely F1 and F3. Hence, the average ranking of FSOS is 1 and it is considerably ahead of SOS.

In Table 5, the standard deviation is also calculated from the discovered X_{best} to indicate the consistency of the discovered solutions. For 7 unimodal functions, FSOS attains the lowest standard deviation in 4 cases (F1, F2, F3, and F4), whereas, SOS attains the same in 4 cases (F1, F2, F3, and F7), TLBO in 3 cases (F1, F3, and F5), and Jaya in 1 case (F6). These results prove the consistency of the FSOS in finding the solutions. Again, in 4, 3, and 2 cases where identical solutions are for FSOS, SOS, and TLBO, respectively and therefore, receive 0.000E+00 standard deviation (see Table 5). For TLBO, none of these cases are the global minimums; whereas, for FSOS and SOS, in 2 cases both attain the global minimums, namely F1 and F3.

2) MULTIMODAL FUNCTIONS

The acquired results of multimodal functions are presented in Table 6. It is noteworthy to mention that unlike unimodal functions, multimodal functions have multiple optimum solutions. However, in our experiments, all the compared metaheuristics endeavor to discover the global optimum solution. Nevertheless, if there are multiple global optimum solutions due to having equal fitness values, discovering one of those solutions would be considered as mission accomplished.

Similar result trends as unimodal functions (see Section V1-A1) are observed for multimodal functions. However, the Jaya algorithm improves and secures the 4th position leaving behind SCA. The rest of the ranks remain identical as unimodal functions. However, the number of global minima found has increased to 3 for FSOS and SOS. On the contrary, the other contenders still fail to discover any global optimum solution from 6 functions. Although, SOS manages to outperform FSOS in the case of F13 or Generalized Penalized 02 function; on the contrary, FSOS outperforms SOS in the case of F12 or Generalized Penalized 01 function. In other cases, both attain identical mean values; and hence, ranked identically as 1st.

TABLE 4. Description of Fixed-Dimension Multimodal Benchmark Functions.

	Function	Dim	Range	f_{min}
Shekel's Foxholes	$F_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^2 (x_i - a_{ij})^6} \right)^{-1}$ $a = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \dots & 32 & 32 & 32 \end{pmatrix}$	2	[-65,65]	1
Kowalik	$F_{15}(x) = \sum_{i=0}^{10} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$ $a = (0.1957 \ 0.1947 \ 0.1735 \ 0.1600 \ 0.0844 \ 0.0627 \ 0.0456 \ 0.0342 \ 0.0323 \ 0.0235 \ 0.0246)$ $b = (0.25 \ 0.5 \ 1 \ 2 \ 4 \ 6 \ 8 \ 10 \ 12 \ 14 \ 16)$	4	[-5,5]	3.075E-4
Six Hump Camelback	$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[-5,5]	-1.0316
Branin	$F_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos x_1 + 10$	2	[-5,5]	0.39789
Goldstein-Price	$F_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	[-2,2]	3
Hartmann3D	$F_{19}(x) = - \sum_{i=1}^4 c_i \exp \left(- \sum_{j=1}^3 a_{ij} (x_j - p_{ij})^2 \right)$	3	[0,1]	-3.86278
Hartmann6D	$F_{20}(x) = - \sum_{i=1}^4 c_i \exp \left(- \sum_{j=1}^6 a_{ij} (x_j - p_{ij})^2 \right)$	6	[0,1]	-3.32237
Shakel05	$F_{21}(x) = - \sum_{i=1}^5 [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.1527
Shakel07	$F_{22}(x) = - \sum_{i=1}^7 [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.3999
Shakel10	$F_{23}(x) = - \sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.5319

In the case of standard deviations in Table 6, FSOS and SOS both attain minimum standard deviation, 0.00E+00 in 3 cases (for F9, F10, and F11) and TLBO attains in 2 cases (for F10 and F11). However, none of these two cases are the global optimum solutions; whereas, 2 among 3 cases are the global optimum solutions for FSOS and SOS indicating their performance consistencies.

3) FIXED-DIMENSION MULTIMODAL FUNCTIONS

As the name suggests, unlike multimodal functions, fixed-dimension multimodal functions are considerably less complex due to having a fixed number of dimensions. Therefore, almost all compared algorithms — except CFOA — attain competing results, and their ranks also changed extensively, unlike the last two sets of functions. The highest performance improvement is exhibited by the Jaya algorithm as presented in Table 7. Although, for unimodal and multimodal functions, Jaya fails to discover the global optimum solution even once;

however, for the current set of functions, it discovers 4 global optimum solutions for F16, F17, F18, and F19. Similarly, TLBO discovers the global optimum solutions for 3 cases and SCA discovers for 2 cases, where both of them failed before in case of unimodal and multimodal functions. Fixed dimensions of the searching spaces are facilitating these metaheuristics in finding the global optimum solutions.

On the contrary, the performance of SOS declines the most and slips to rank 5, only ahead of CFOA. Again, slightly losing for a few functions, specifically for 3 functions (F20, F21, and F23), on a positive note, FSOS still manages to get the first overall ranking followed closely by Jaya, TLBO, and SOS. Observing the average ranking in Table 7, it can be stated that all top 5 metaheuristics exhibit competing results and their differences are marginal. However, thanks to the FIS, which facilitates FSOS in attaining the highest performance by selecting the phases following a fuzzy decision making process.

TABLE 5. Results of Unimodal Benchmark Functions.

No		Min	Jaya	CFOA	SCA	TLBO	SOS	FSOS
F1	Best	0.000E+00	4.204E-11	1.750E+01	2.967E-11	5.893E-184	0.000E+00	0.000E+00
	Mean		1.919E-09	2.965E+01	1.326E-09	6.439E-180	0.000E+00	0.000E+00
	StdDev		3.112E-09	8.287E+00	2.547E-09	0.000E+00	0.000E+00	0.000E+00
	Rank		5	6	4	3	1	1
F2	Best	0.000E+00	3.185E-08	3.114E+01	2.732E-08	4.993E-98	2.183E-192	5.319E-228
	Mean		1.661E-07	4.560E+01	1.690E-07	1.612E-96	1.384E-189	2.051E-222
	StdDev		1.135E-07	6.777E+00	1.381E-07	1.891E-96	0.000E+00	0.000E+00
	Rank		4	6	5	3	2	1
F3	Best	0.000E+00	6.981E-09	6.443E+04	4.500E-08	1.693E-180	0.000E+00	0.000E+00
	Mean		7.333E-07	1.524E+05	4.438E-07	4.704E-177	0.000E+00	0.000E+00
	StdDev		1.895E-06	4.756E+04	4.792E-07	0.000E+00	0.000E+00	0.000E+00
	Rank		5	6	4	3	1	1
F4	Best	0.000E+00	6.441E+01	3.695E+01	7.532E+01	2.058E-78	1.275E-155	5.430E-179
	Mean		6.421E+01	4.557E+01	6.154E+01	1.188E-77	7.761E-154	4.321E-176
	StdDev		8.272E+00	6.941E+00	1.080E+01	1.663E-77	3.672E-153	0.000E+00
	Rank		6	4	5	3	2	1
F5	Best	0.000E+00	2.965E+01	1.952E+06	3.358E+01	2.674E+01	2.474E+01	2.455E+01
	Mean		5.916E+02	9.579E+06	7.733E+02	2.747E+01	2.595E+01	2.550E+01
	StdDev		5.638E+02	5.974E+06	1.116E+03	3.440E-01	7.017E-01	4.443E-01
	Rank		4	6	5	3	2	1
F6	Best	0.000E+00	2.884E-08	5.665E+03	1.839E-08	1.563E-04	8.482E-08	6.971E-11
	Mean		5.550E-07	1.154E+04	1.187E-06	4.569E-02	2.452E-03	8.624E-06
	StdDev		9.837E-07	3.840E+03	2.806E-06	8.573E-02	4.936E-03	2.944E-05
	Rank		1	6	2	5	4	3
F7	Best	0.000E+00	3.276E-01	3.911E+07	2.501E-01	5.710E-04	4.214E-05	3.252E-05
	Mean		9.051E+00	1.660E+08	5.604E+00	1.301E-03	2.464E-04	2.316E-04
	StdDev		2.123E+01	1.131E+08	1.353E+01	4.400E-04	1.082E-04	1.151E-04
	Rank		5	6	4	3	2	1
No of Global minimum found			0	0	0	0	2	2
Average Ranking			4.286	5.714	4.143	3.286	2.000	1.286
Overall Ranking			5	6	4	3	2	1

4) CONVERGENCE CHARACTERISTICS

In terms of convergence characteristics for the unimodal functions as shown in Figure 6, an intense competition can be observed between the SOS and FSOS. On the other hand, TLBO manages to converge fairly along with Jaya. The other metaheuristics, namely SCA and CFOA converge last by taking nearly 1000 iterations. Although, seen to start converging first; however, CFOA get stuck in local optimum and is unable to recover. Among FSOS and SOS, that latter is seen to have a slightly faster convergence as compared to the former for F1 and F4 functions. Again, they demonstrate neck to neck convergence rates for F2, F3, and F6 functions. On the contrary, FSOS has better convergence for F5 and F7 functions. Even though, there is no notable difference in terms of convergence rate between SOS and FSOS; however, since FSOS continues exploiting even after converging that makes FSOS attaining more precise results than SOS (see Table 5).

Analogous to the convergence characteristics of unimodal functions, CFOA is also seen to start converging faster than other algorithms for multimodal functions as depicted in Figure 7. However, CFOA then get stuck in local optima, hence, exhibiting poor performance (see Table 6). A competitive convergence characteristic is observed between the SOS

and FSOS where the latter dominates the former for F9, F10, and F11 functions; and, the former leads for F8, F11, and F12 functions.

In the convergence curves of fixed-dimension multimodal functions, which are presented in Figure 8, all the compared algorithms manage to attain competitive results by demonstrating their unique abilities and characteristics, which make none of the algorithms to stand out the most. Although, CFOA manages to converge the earliest for F15 and F16 functions; however, according to the results presented in Table 6, it fails to attain the sharpest results. Among the other metaheuristics, SOS has a better convergence rate for F21 and F22 functions; whereas, TLBO performs better for F20 function while Jaya performs better for F19 and F21 functions. In comparison to the other metaheuristics, FSOS is seen to have a better convergence rates for 3 functions, which is the highest among all, for F14, F18, and F23.

B. COMBINATORIAL TEST SUITE GENERATION TEST RESULTS

The acquired results for covering array and mixed covering array are presented in Table 8 and Table 9, respectively. In these tables, the last three rows summarize the other results,

TABLE 6. Results of Multimodal Benchmark Functions.

No		Min	Jaya	CFOA	SCA	TLBO	SOS	FSOS
F8	Best	-1.257E+04	-6.216E+03	-4.157E+03	-6.349E+03	-9.278E+03	-1.257E+04	-1.257E+04
	Mean		-5.098E+03	-2.754E+03	-5.240E+03	-7.522E+03	-1.257E+04	-1.257E+04
	StdDev		4.755E+02	6.501E+02	4.577E+02	8.139E+02	7.786E-01	1.308E+00
	Rank		4	6	5	3	1	1
F9	Best	0.000E+00	2.002E+02	1.751E+02	1.890E+02	3.241E+01	0.000E+00	0.000E+00
	Mean		2.313E+02	2.705E+02	2.375E+02	5.211E+01	0.000E+00	0.000E+00
	StdDev		1.713E+01	3.035E+01	2.290E+01	1.606E+01	0.000E+00	0.000E+00
	Rank		4	6	5	3	1	1
F10	Best	0.000E+00	5.443E-05	1.335E+01	1.924E-04	4.441E-16	4.441E-16	4.441E-16
	Mean		6.545E+00	1.541E+01	7.976E+00	4.441E-16	4.441E-16	4.441E-16
	StdDev		9.243E+00	1.048E+00	9.607E+00	0.000E+00	0.000E+00	0.000E+00
	Rank		4	6	5	1	1	1
F11	Best	0.000E+00	1.456E-10	3.144E+01	2.963E-10	1.644E-182	0.000E+00	0.000E+00
	Mean		4.184E-09	1.044E+02	4.118E-09	3.936E-179	0.000E+00	0.000E+00
	StdDev		7.160E-09	2.865E+01	5.966E-09	0.000E+00	0.000E+00	0.000E+00
	Rank		5	6	4	3	1	1
F12	Best	0.000E+00	4.230E+00	3.487E+04	6.441E+00	1.470E-05	3.296E-10	4.236E-13
	Mean		2.164E+01	4.790E+06	1.776E+01	1.222E-01	7.573E-06	3.537E-07
	StdDev		2.072E+01	4.972E+06	6.148E+00	2.010E-01	1.412E-05	1.281E-06
	Rank		5	6	4	3	2	1
F13	Best	0.000E+00	1.626E+00	4.627E+06	9.504E-01	1.207E-02	1.056E-06	2.826E-11
	Mean		2.240E+01	1.866E+07	4.659E+01	5.123E-01	4.969E-03	9.182E-03
	StdDev		1.145E+01	1.151E+07	1.410E+02	3.784E-01	1.081E-02	1.481E-02
	Rank		4	6	5	3	1	2
No of Global minimum found			0	0	0	0	3	3
Average Ranking			4.333	6.000	4.667	2.667	1.167	1.167
Overall Ranking			4	6	5	3	1	1

which are: no. of most minimal found, average ranking, and overall ranking. Herein, the last two measures are similar to that in Section VI-A. However, in the case of the first measure, the number of most minimal found is counted for each compared metaheuristic taking the mean values into account due to the fact that no absolute global optimum solutions are reported for the assessed case study. Again, the mean values are employed for the same reason that we have mentioned in Section VI-A.

As of the results of test suite generation for covering array listed in Table 8, FSOS generates the most minimal test suite size by discovering the minimum test cases for 7 out of 12 test scenarios, namely S1, S2, S4, S6, S8, S10, and S11. On the contrary, for 4 scenarios, namely S1, S3, S9, and S12, SOS generates the most minimal test cases and TLBO generates for 2 scenarios (S1 and S5) while Jaya generates for 1 scenario (S7). The other two metaheuristics, SCA and CFOA fail to discover the most minimal solution even once. Again, in terms of average ranking as well as overall ranking, FSOS convincingly outperforms its predecessor, SOS followed by TLBO, Jaya, SCA, and CFOA.

The results of mixed covering array test case generation are presented in Table 9, Here again, FSOS maintains its top position with an average ranking of 1.75. The performance of FSOS can be attributed to the adoption of FIS. Specifically, FIS enables FSOS to select the phases adaptively based on the

need of the current search process. However, its predecessor slips to rank 4 due to following a deterministic sequence for executing the involved phases. Among the other metaheuristics, SCA attains improved performance and receives rank 2 followed by TLBO, SOS, CFOA, and Jaya algorithms.

C. STATISTICAL ANALYSIS

The Wilcoxon Signed Rank test was applied to identify whether the result of FSOS is statistically significant. To be specific, the null hypothesis (H_0) states that FSOS performance is the same as any other algorithms. Results of this test are summarized in Table 10 and 11. The resulting values were compared with the chosen significance level $\alpha = 0.05$ and $\alpha = 0.10$. From Table 10 related to the benchmark functions, FSOS is statistically better than all compared algorithms with both 90 percent confidence level. In fact, with the exception of TLBO, FSOS is also statistically better than Jaya, CFOA, SCA, and SOS. Concerning Table 11 related to t-way test suite generation, FSOS is statistically better than Jaya, CFOA, SCA, and TLBO. On a negative note, there is no significant difference between FSOS and SOS.

D. EXPERIMENTAL OBSERVATION AND DISCUSSION

Based on the aforementioned discussions, a number of thought-provoking observations can be drawn, which are elaborated as follows:

TABLE 7. Results of Fixed-Dimensions Multimodal Benchmark Functions.

No		Min	Jaya	CFOA	SCA	TLBO	SOS	FSOS
F14	Best	1.000E+00	9.980E-01	9.980E-01	9.980E-01	9.980E-01	9.980E-01	9.980E-01
	Mean		9.980E-01	1.394E+01	1.482E+00	9.980E-01	9.980E-01	9.980E-01
	StdDev		1.073E-16	1.878E+01	2.604E+00	2.027E-16	1.110E-16	1.110E-16
	Rank		1	6	5	1	1	1
F15	Best	3.075E-04	3.134E-04	2.438E-03	3.561E-04	3.075E-04	3.075E-04	3.075E-04
	Mean		1.442E-03	6.423E-02	7.299E-04	3.621E-03	3.381E-04	3.075E-04
	StdDev		3.522E-03	4.728E-02	2.515E-04	7.126E-03	1.644E-04	1.575E-13
	Rank		4	6	3	5	2	1
F16	Best	-1.032E+00	-1.032E+00	-1.032E+00	-1.032E+00	-1.032E+00	-1.032E+00	-1.032E+00
	Mean		-1.032E+00	-7.360E-01	-1.032E+00	-1.032E+00	-1.032E+00	-1.032E+00
	StdDev		5.119E-06	3.982E-01	1.166E-05	4.478E-16	4.532E-16	4.441E-16
	Rank		1	6	1	1	1	1
F17	Best	3.979E-01	3.979E-01	3.980E-01	3.979E-01	3.979E-01	3.979E-01	3.979E-01
	Mean		3.979E-01	2.835E+00	3.979E-01	3.979E-01	3.979E-01	3.979E-01
	StdDev		0.000E+00	5.291E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
	Rank		1	6	1	1	1	1
F18	Best	3.000E+00	3.000E+00	3.378E+00	3.000E+00	3.000E+00	3.000E+00	3.000E+00
	Mean		3.000E+00	5.102E+01	3.000E+00	8.400E+00	3.900E+00	3.000E+00
	StdDev		2.293E-15	8.767E+01	2.189E-15	1.620E+01	4.847E+00	1.994E-15
	Rank		1	6	1	5	4	1
F19	Best	-3.863E+00	-3.863E+00	-3.856E+00	-3.863E+00	-3.863E+00	-3.863E+00	-3.863E+00
	Mean		-3.863E+00	-3.335E+00	-3.863E+00	-3.863E+00	-3.785E+00	-3.863E+00
	StdDev		3.109E-15	6.128E-01	3.109E-15	6.686E-16	2.319E-01	3.109E-15
	Rank		1	6	1	1	5	1
F20	Best	-3.322E+00	-3.322E+00	-2.848E+00	-3.322E+00	-3.322E+00	-3.322E+00	-3.322E+00
	Mean		-3.220E+00	-1.733E+00	-3.230E+00	-3.306E+00	-3.275E+00	-3.291E+00
	StdDev		5.524E-02	6.594E-01	5.278E-02	4.052E-02	5.839E-02	5.271E-02
	Rank		5	6	4	1	3	2
F21	Best	-1.015E+01	-1.015E+01	-9.949E+00	-1.015E+01	-1.015E+01	1.015E+01	-1.015E+01
	Mean		-7.198E+00	-2.425E+00	-7.197E+00	-8.382E+00	7.606E+00	-7.944E+00
	StdDev		2.352E+00	2.895E+00	2.001E+00	3.163E+00	2.547E+00	2.526E+00
	Rank		4	6	5	1	3	2
F22	Best	-1.040E+01	-1.040E+01	-7.754E+00	-1.040E+01	-1.040E+01	-1.040E+01	-1.040E+01
	Mean		-8.890E+00	-2.115E+00	-8.142E+00	-8.744E+00	-7.878E+00	-9.163E+00
	StdDev		1.821E+00	1.786E+00	2.250E+00	3.015E+00	2.711E+00	2.248E+00
	Rank		2	6	4	3	5	1
F23	Best	-1.054E+01	-1.054E+01	-7.782E+00	-0.54E+01	-1.054E+01	-1.054E+01	-1.054E+01
	Mean		-9.157E+00	-2.312E+00	8.303E+00	-7.427E+00	-7.727E+00	-8.158E+00
	StdDev		2.391E+00	1.782E+00	2.512E+00	3.340E+00	2.826E+00	2.727E+00
	Rank		1	6	2	5	4	3
No of Global minimum found			4	0	4	3	2	4
Average Ranking			2.100	6.000	2.700	2.400	2.900	1.400
Overall Ranking			2	6	4	3	5	1

- Given its overall ranking performance, we conclude that FSOS is able to judiciously balance its exploration and exploitation process. Unlike the original SOS which executes the three search phases (commensalism, mutualism, and parasitism) in deterministic sequence, FSOS allows the non-deterministic search phase selection based on the individual weighted performance via FIS. Based on the penalized and reward model, the best performing search phase has a higher probability for subsequent selection using the probability bar selection.

In this manner, at any instance of execution, FSOS may run any single search phase, a combination of two search phases or even all the search phases at once. This adaptive behavior is the key component that enhances the performance of FSOS over SOS and other competing metaheuristic algorithms. Statistical analysis shows that FSOS outperforms all other metaheuristic algorithms at 90% confidence level (refer Table 10) with the exception of its predecessor SOS in only one case (refer to Table 11).

TABLE 8. Comparison between Existing Algorithms on *t*-way Test Suite Generation (Covering Array).

No	Configuration		Jaya	CFOA	SCA	TLBO	SOS	FSOS
S1	$CA(N; 2, 3^4)$	Best	9	9	9	9	9	9
		Mean	10.767	9.167	9.033	9.000	9.000	9.000
		StdDev	1.430	0.453	0.180	0.000	0.000	0.000
		Rank	6	5	4	1	1	1
S2	$CA(N; 2, 3^{13})$	Best	19	19	19	17	18	17
		Mean	19.800	20.000	21.400	18.833	19.133	18.767
		StdDev	0.653	0.683	1.143	0.734	0.670	0.761
		Rank	4	5	6	2	3	1
S3	$CA(N; 2, 10^{10})$	Best	201	160	162	161	152	154
		Mean	204.400	164.867	164.600	164.400	154.300	156.300
		StdDev	1.562	2.202	1.562	1.381	0.971	1.187
		Rank	6	5	4	3	1	2
S4	$CA(N; 2, 15^{10})$	Best	456	502	421	346	348	339
		Mean	461.800	515.667	434.167	349.767	350.733	342.467
		StdDev	2.400	8.471	11.671	1.802	1.672	1.628
		Rank	5	6	4	2	3	1
S5	$CA(N; 2, 5^{10})$	Best	48	44	47	42	42	43
		Mean	50.867	45.300	51.167	43.400	43.433	43.700
		StdDev	1.996	0.936	2.324	0.987	0.844	0.936
		Rank	5	4	6	1	2	3
S6	$CA(N; 2, 8^{10})$	Best	105	106	106	106	105	100
		Mean	108.333	108.233	108.133	108.300	108.066	102.233
		StdDev	1.193	1.202	1.176	1.100	1.263	1.146
		Rank	6	4	3	5	2	1
S7	$CA(N; 3, 3^6)$	Best	33	33	33	33	33	33
		Mean	37.900	39.167	43.533	39.667	40.633	39.067
		StdDev	4.028	4.067	3.481	3.467	3.665	3.864
		Rank	1	3	6	4	5	2
S8	$CA(N; 3, 4^6)$	Best	103	139	99	78	69	68
		Mean	106.267	146.667	106.633	98.967	94.467	94.333
		StdDev	2.015	4.002	2.470	6.921	11.188	11.493
		Rank	4	6	5	3	2	1
S9	$CA(N; 3, 5^6)$	Best	187	190	199	193	187	191
		Mean	197.000	197.767	206.900	196.867	193.967	195.300
		StdDev	3.316	2.985	3.789	1.647	3.136	2.492
		Rank	4	5	6	3	1	2
S10	$CA(N; 3, 6^6)$	Best	333	333	337	330	331	329
		Mean	339.933	339.367	352.100	336.200	335.100	332.767
		StdDev	2.632	2.787	6.789	2.159	1.786	2.499
		Rank	5	4	6	3	2	1
S11	$CA(N; 3, 5^7)$	Best	224	226	223	224	223	218
		Mean	227.933	230.100	226.767	227.500	227.500	220.833
		StdDev	1.916	1.955	1.874	1.586	2.045	1.675
		Rank	5	6	2	3	3	1
S12	$CA(N; 3, 10^6)$	Best	1527	1828	1491	1496	1471	1473
		Mean	1539.400	1872.000	1557.233	1503.067	1478.200	1480.300
		StdDev	4.835	23.340	23.341	4.305	3.855	4.018
		Rank	4	6	5	3	1	2
No of most minimum found			1	0	0	2	4	7
Average Ranking			4.583	4.917	4.750	2.750	2.167	1.500
Overall Ranking			4	6	5	3	2	1

- More precisely, the introduction of FIS within FSOS has improved its exploration and exploitation abilities as demonstrated by the benchmarking experiments. More precisely, the exploration and exploitation of FSOS is made dynamic via the fuzzy rules by breaking the pre-set sequence of the individual search operators. Contrary

to typical metaheuristic algorithms which require tuning of the control parameters to gain balance control of the exploration and the exploitation process, FSOS takes the best of both worlds – behaves like a parameterized algorithm without the need for significant tuning.

TABLE 9. Comparison between Existing Algorithms on *t*-way Test Suite Generation (Mixed-Coverage Array).

No	Configuration	Jaya	CFOA	SCA	TLBO	SOS	FSOS	
S13	$MCA(N; 2, 5^1 3^8 2^2)$	Best	23	21	20	20	21	20
		Mean	24.700	22.233	21.833	21.800	22.567	21.567
		StdDev	0.971	0.883	0.820	0.945	0.883	0.989
		Rank	6	4	3	2	5	1
S14	$MCA(N; 2, 7^1 6^1 5^1 4^6 3^8 2^3)$	Best	49	51	49	48	44	46
		Mean	52.433	53.733	52.400	52.500	52.067	50.700
		StdDev	1.726	1.652	1.7048	1.765	2.489	2.132
		Rank	4	6	3	5	2	1
S15	$MCA(N; 3, 5^2 4^2 3^2)$	Best	104	104	104	106	106	108
		Mean	119.400	119.367	116.000	118.533	117.167	118.567
		StdDev	6.560	5.174	6.923	6.627	5.956	4.870
		Rank	6	5	1	3	2	4
S16	$MCA(N; 3, 10^1 6^2 4^3 3^1)$	Best	382	381	385	379	385	382
		Mean	392.700	390.767	393.867	392.567	392.867	389.867
		StdDev	5.610	3.809	5.194	6.087	4.674	3.922
		Rank	4	2	6	3	5	1
No of most minimum found		0	0	1	0	0	3	
Average Ranking		5.000	4.250	2.750	3.250	3.500	1.750	
Overall Ranking		6	5	2	3	4	1	

TABLE 10. Summary of Wilcoxon Signed Rank Test Results for Benchmark Functions.

	R-	R+	Critical Values		Remarks
			$t_{0.05,23}$	$t_{0.10,23}$	
FSOS versus Jaya	253	23	73	83	Reject the null hypothesis with 90% and 95% confidence level
FSOS versus CFOA	276	0	73	83	Reject the null hypothesis with 90% and 95% confidence level
FSOS versus SCA	240	36	73	83	Reject the null hypothesis with 90% and 95% confidence level
FSOS versus TLBO	193	83	73	83	Reject the null hypothesis with 90% confidence level
FSOS versus SOS	248	28	73	83	Reject the null hypothesis with 90% and 95% confidence level

TABLE 11. Summary of Wilcoxon Signed Rank Test Results for *t*-way Test Suite Generation.

	R-	R+	Critical Values		Remarks
			$t_{0.05,16}$	$t_{0.10,16}$	
FSOS versus Jaya	122	14	29	35	Reject the null hypothesis with 90% and 95% confidence level
FSOS versus CFOA	136	0	29	35	Reject the null hypothesis with 90% and 95% confidence level
FSOS versus SCA	123	13	29	35	Reject the null hypothesis with 90% and 95% confidence level
FSOS versus TLBO	109	27	29	35	Reject the null hypothesis with 90% and 95% confidence level
FSOS versus SOS	89	47	29	35	Cannot reject the null hypothesis

- As far as convergence is concerned for various sets of experiments, no major differences are noticed between FSOS and SOS. In fact, our experimental results demonstrate that both FSOS and SOS can reach convergence in

earlier iteration than the other competing metaheuristic algorithms.

- Concerning the time complexity, the Big O notation can be conveniently adopted to compare the performance of

TABLE 12. Time Complexity of the Compared Meta-Heuristics.

Algorithm	Time Complexity	Remarks
FSOS	$= O(Max_{iter} \times c \times n / N_size + 3 \times Max_{iter} \times n / N_size + Max_{iter} \times d \times n / N_size) + C$	The time complexity of FSOS includes the multiplier 3 to capture the fact that the agent update occurs once per iteration (based on the selected operator) and at most three times (mutualism, commensalism, and parasitism phase). The time complexity also defines the constant value C to cater for fuzzy decision making process.
SOS	$= O(Max_{iter} \times c \times n / N_size + Max_{iter} \times n / N_size + 1 + Max_{iter} \times d \times n / N_size)$	Unlike FSOS, the time complexity of SOS does not include the multiplier 3. In SOS, the same agent within a population is updated three times per iteration based on the selected operator (mutualism to commensalism and parasitism operators). The three updates per iteration is modelled as fixed constant value 1.
Jaya	$= O(Max_{iter} \times c \times n / N_size + Max_{iter} \times n / N_size + Max_{iter} \times d \times n / N_size)$	Jaya only performs a single agent update per iteration
CFOA	$= O(Max_{iter} \times c \times n / N_size + Max_{iter} \times n / N_size + Max_{iter} \times d \times n / N_size)$	CFOA only performs a single agent update per iteration
SCA	$= O(Max_{iter} \times c \times n / N_size + Max_{iter} \times n / N_size + Max_{iter} \times d \times n / N_size)$	Despite having two operators, SCA only performs a single agent update per iteration based on probability = 0.5
TLBO	$= O(Max_{iter} \times c \times n / N_size + Max_{iter} \times n / N_size + 1 + Max_{iter} \times d \times n / N_size)$	TLBO updates the same population two times per iteration. The two updates per iteration is modelled as fixed constant value 1.

each metaheuristic algorithm under study. Typically, the time complexity is dependent on the number of search agents (n), the number of dimensions (d), the number of maximum iteration (Max_{iter}), and the fitness function evaluation (c). With the population size = N_size , the time complexity of any algorithm (A) can generally be defined as in Eq. 17:

$$O(A) = O(\text{fitness function evaluation}) + O(\text{agent update in memory}) + O(\text{dimension update}) \quad (17)$$

Based on Eq. 17, the time complexity of all the compared metaheuristic algorithms can be summarized in Table 12. Assuming all other operations can be performed in a constant time, the time complexity for all metaheuristic algorithms in Table 12 can be approximated as $\approx O(Max_{iter} \times c \times n / N_size + Max_{iter} \times d \times n / N_size)$ when n is sufficiently large. From this approximation, it can be deduced that the introduction of FIS within does not have any direct performance penalty.

- In terms of fairness of comparison with other metaheuristic algorithms, a number of issues can be further elaborated. Firstly, we have only compared our work against parameter-free based metaheuristic algorithms involving Jaya, CFOA, SCA, TLBO, and SOS. These algorithms do not provide any parameter controls (with the exception of population size and maximum iteration). As such, they are not subjected to tuning which makes their exploration and exploitation remain

in pre-set mode. Instead of using the maximum iteration, our work opted for maximum fitness evaluation (with the same population size) as the stopping criteria. The significance of our choice stemmed from the fact that each algorithm must have the same number of agent updates. Consider an analogy of throwing a dart. If one person throws a dart 20 times, he has more chances of hitting the bull's eye than another person throwing only 10 times. In the case of TLBO, the agent updates occur twice (once in both the learner and teacher phase). Given a population size of N_size , and iteration of Max_{iter} , there are $2 \times N_size \times Max_{iter}$ fitness evaluations overall within TLBO. Meanwhile, SOS have three fitness evaluations per iteration, thus, contributing to $3 \times N_size \times Max_{iter}$ fitness evaluations overall. In contrast, Jaya, CFOA, and SCA has only $N_size \times Max_{iter}$ fitness evaluation overall. Obviously, considering the same N_size and Max_{iter} for all algorithms would not make the comparison fair. For this reason, our work has considered the same maximum fitness evaluation as stopping criteria instead. In our case, we have set $Max_{iter} = \infty$ (refer line 5 in Algorithm 2) with common N_size for all algorithms. The iteration stops when the maximum fitness evaluation is reached (refer to lines 28-29 in Algorithm 2) regardless of the iteration count. Secondly, unlike the minimum best results for the benchmark functions (or the best suite size for t -way testing) which are absolute (regardless of how the algorithm is implemented), time performances depend on the efficiency of implementation, choice of language implementation, and data structure. For this

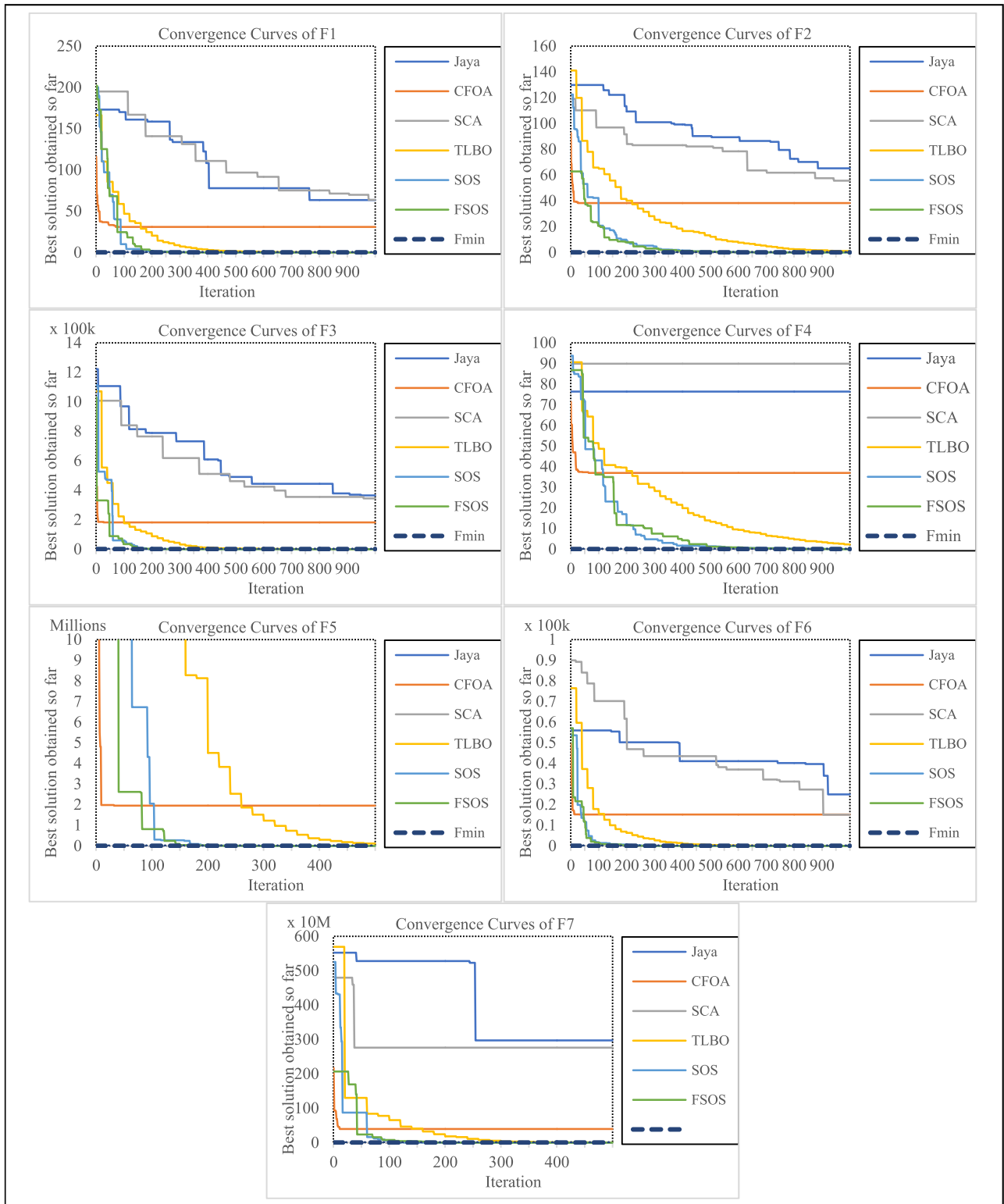


FIGURE 6. Convergence Curves for Multimodal Functions F1 to F7.

reason, we don't make a comparison of time in our work. Finally, we have opted for mean results over the best results for our Wilcoxon Signed Rank test analysis. Given that all metaheuristic algorithms rely on

randomness to do solution updates (via trial and error), their performance can be subjected to chances. One algorithm may produce the best result once by luck, but poor results in a subsequent run. As such, using the mean

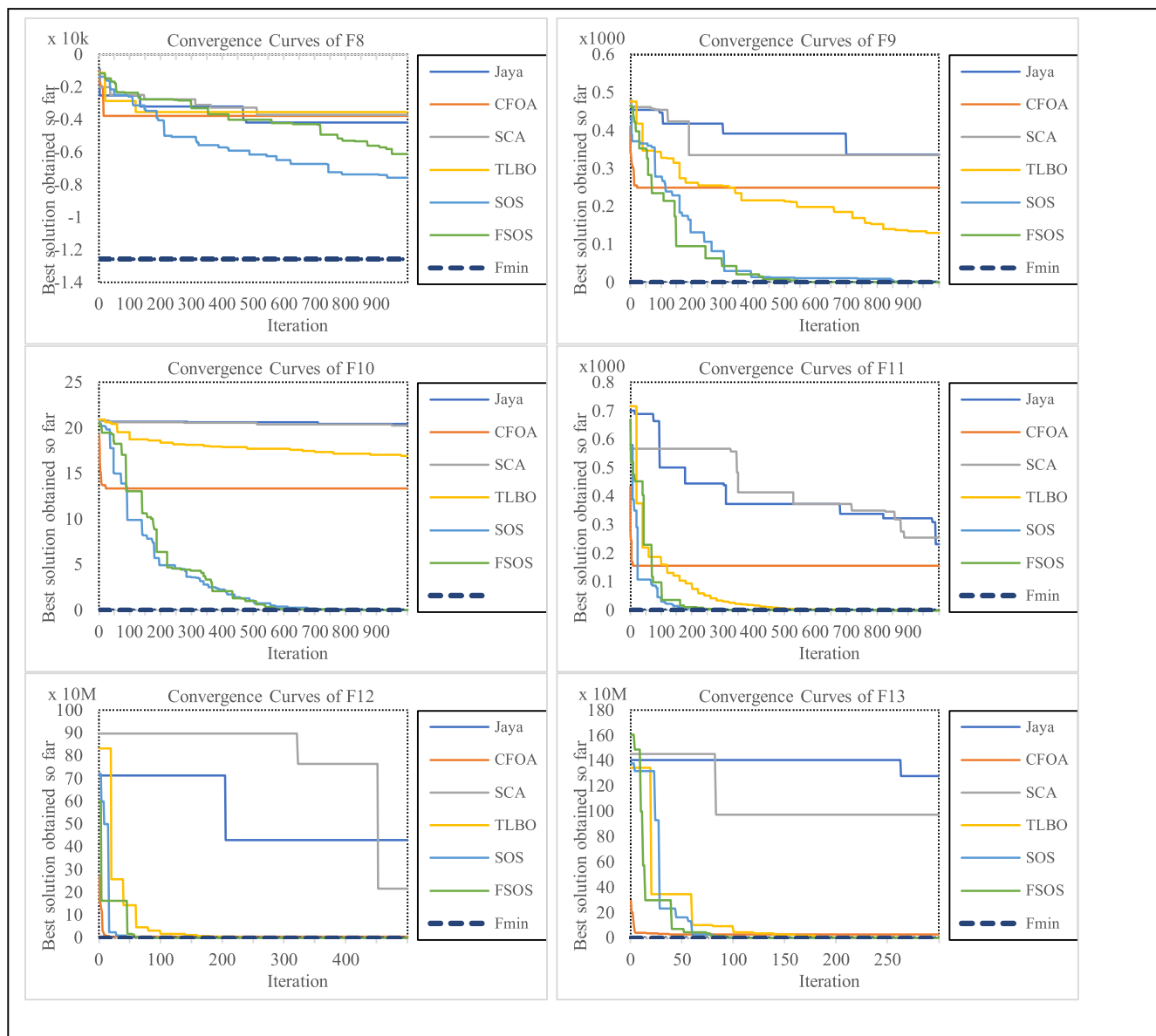


FIGURE 7. Convergence Curves for Multimodal Functions F8 to F13.

can give a better indication of performance than the best results.

- Concerning limitations, we are aware that our FSOS performance may well depend on our fuzzy design choices. Owing to the simplicity of maintaining the fuzzy rules, we have opted for Mamdani over Sugeno based approach. For the same reason, we have opted for the triangular/trapezoidal membership function over the Gaussian one. We have also used overlapping linguistic terms for input membership functions between 0 to 60 (into three members consisting of low, medium, and high) and flatten the other value to 100. Our output membership function is composed of 5 overlapping (and equal-width) membership functions from -50 to 50 to

deal with bar movement (either to the left or to the right). Based on our design choices, we can observe some limitations of our approach. Firstly, the choice of linguistic terms can be seen as two sides of the same coin. On one hand, too many linguistic terms invite more rules, hence, require more computation. On the other hand, too little linguistic terms might not capture the need of the problems. As we have developed our fuzzy FIS based on our design choices, there could be other variants that may well work better. Different design choices (e.g. Mamdani versus Sugeno approach, Gaussian versus Trapezoidal membership function) might lead to different results as there are no guarantees that our

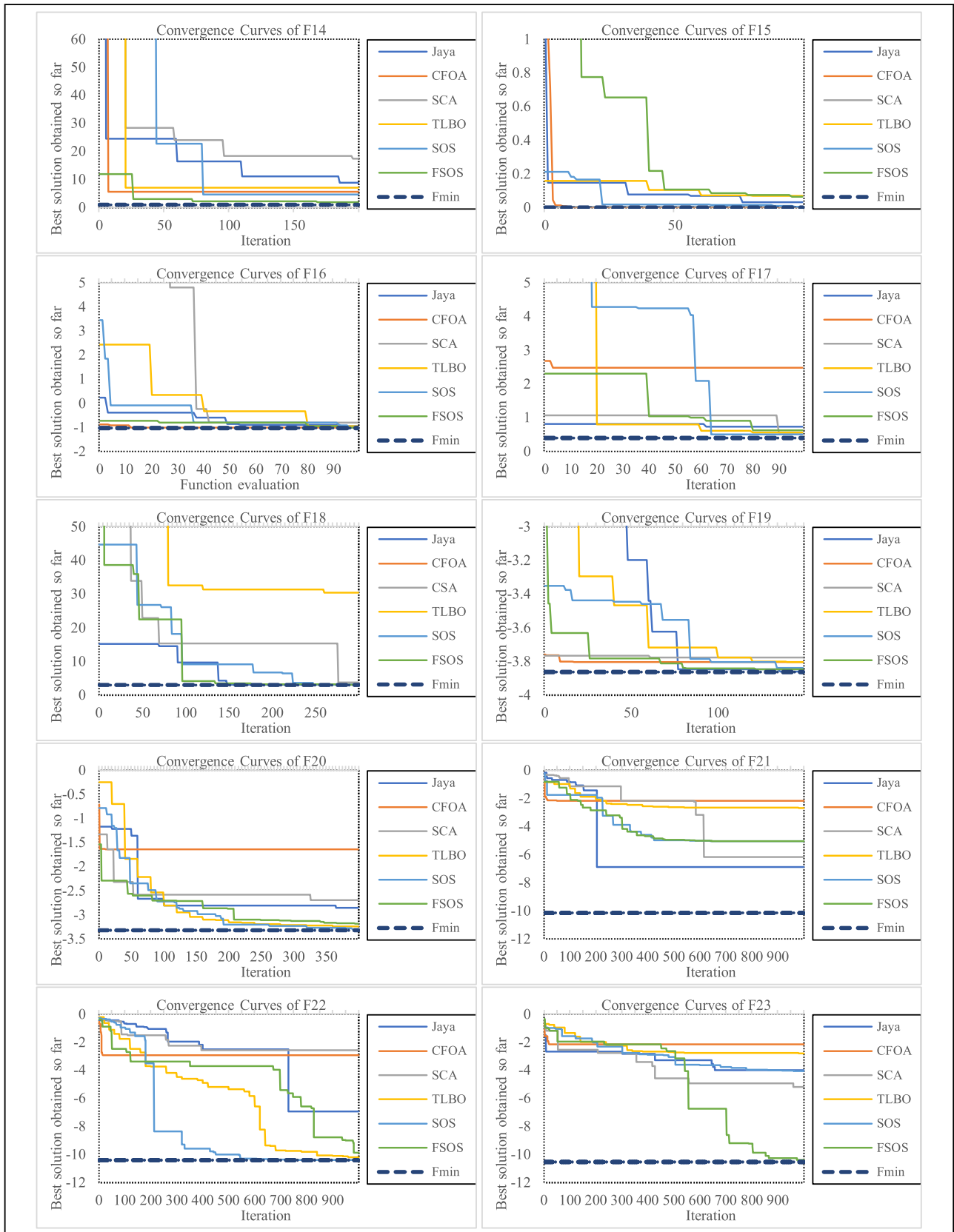


FIGURE 8. Convergence Curves for Fixed-Dimension Multimodal Functions F14 to F23.

choices are sufficiently general for other optimization problems.

VII. CONCLUSION

In this paper, FSOS — an adaptive variant of SOS is proposed to solve general optimization problems. Here, the Mamdani Fuzzy interference system is integrated with SOS for selecting a single or any combination of update operators among mutualism, commensalism, and parasitism adaptively based on the setting of their individual probabilities via fuzzy decision-making process. The performance of the proposed algorithm is tested on 23 benchmark functions under the categories: unimodal, multimodal, and fixed-dimension multimodal and also taking a t -way test generation as a case study. Based on the acquired experimental results, FSOS not only outperforms its predecessor, SOS, but also other compared metaheuristic algorithms for all sets of experiments that are conducted in this work. Summing up, it is worth noting that our FIS approach is also applicable to other metaheuristic algorithms such as Monarch Butterfly Optimization (MBO) [62], Earthworm Optimization Algorithm (EWA) [63], Elephant Herding Optimization (EHO) [64], and Moth Search Algorithm (MS) [65] to address the optimization problems highlighted in the current paper. Here, unlike our work which deals with selection search operators, the FIS also can be used to automatically tune each of the algorithm's parameters during runtime.

REFERENCES

- [1] J. Islam, P. M. Vasant, B. M. Negash, M. B. Laruccia, M. Myint, and J. Watada, "A holistic review on artificial intelligence techniques for well placement optimization problem," *Adv. Eng. Softw.*, vol. 141, Mar. 2020, Art. no. 102767.
- [2] G. Villarrubia, J. F. De Paz, P. Chamoso, and F. D. la Prieta, "Artificial neural networks used in optimization problems," *Neurocomputing*, vol. 272, pp. 10–16, Jan. 2018.
- [3] A. Godio and A. Santilano, "On the optimization of electromagnetic geophysical data: Application of the PSO algorithm," *J. Appl. Geophys.*, vol. 148, pp. 163–174, Jan. 2018.
- [4] N. Razaaly, G. Persico, G. Gori, and P. M. Congedo, "Quantile-based robust optimization of a supersonic nozzle for organic rankine cycle turbines," *Appl. Math. Model.*, vol. 82, pp. 802–824, Jun. 2020.
- [5] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Perth, WA, Australia, vol. 4, Nov. 1995, pp. 1942–1948.
- [6] D. E. Goldberg, *Genetic Algorithms in Search, Optimisation and Machine Learning*. Boston, MA, USA: Addison-Wesley, 1989.
- [7] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems," *Comput.-Aided Des.*, vol. 43, no. 3, pp. 303–315, Mar. 2011.
- [8] R. V. Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *Int. J. Ind. Eng. Comput.*, vol. 7, no. 1, pp. 19–34, 2016.
- [9] M.-Y. Cheng and D. Prayogo, "Symbiotic organisms search: A new metaheuristic optimization algorithm," *Comput. Struct.*, vol. 139, pp. 98–112, Jul. 2014.
- [10] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *Int. J. Bio-Inspired Comput.*, vol. 2, no. 2, pp. 78–84, 2010.
- [11] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 66, pp. 46–61, Mar. 2014.
- [12] K. Zervoudakis and S. Tsafarakis, "A mayfly optimization algorithm," *Comput. Ind. Eng.*, vol. 145, Jul. 2020, Art. no. 106559.
- [13] X.-S. Yang, "Flower pollination algorithm for global optimization," in *Unconventional Computation and Natural Computation*. Berlin, Germany: Springer, 2012, pp. 240–249.
- [14] E. Hosseini, "Laying chicken algorithm: A new meta-heuristic approach to solve continuous programming problems," *J. Appl. Comput. Math.*, vol. 6, no. 1, p. 2, 2017.
- [15] X. Meng, Y. Liu, X. Gao, and H. Zhang, "A new bio-inspired algorithm: Chicken swarm optimization," in *Advances in Swarm Intelligence*. Cham, Switzerland: Springer, 2014, pp. 86–94.
- [16] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, May 2016.
- [17] B. Li, "Research on WNN modeling for gold price forecasting based on improved artificial bee colony algorithm," *Comput. Intell. Neurosci.*, vol. 2014, pp. 1–10, Jan. 2014.
- [18] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
- [19] M. Tahani and N. Babayan, "Flow regime algorithm (FRA): A physics-based meta-heuristics algorithm," *Knowl. Inf. Syst.*, vol. 60, no. 2, pp. 1001–1038, Aug. 2019.
- [20] A. R. Moazzeni and E. Khamehchi, "Rain optimization algorithm (ROA): A new Metaheuristic method for drilling optimization solutions," *J. Petroleum Sci. Eng.*, vol. 195, Dec. 2020, Art. no. 107512.
- [21] H. S. Hosseini, "Problem solving by intelligent water drops," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 3226–3231.
- [22] R. A. Formato, "Central force optimization: A new metaheuristic with applications in applied electromagnetics," *Prog. Electromagn. Res.*, vol. 77, pp. 425–491, 2007.
- [23] A. Kaveh and S. Talatahari, "A novel heuristic optimization method: Charged system search," *Acta Mechanica*, vol. 213, nos. 3–4, pp. 267–289, Sep. 2010.
- [24] A. Narayanan and M. Moore, "Quantum-inspired genetic algorithms," in *Proc. IEEE Int. Conf. Evol. Comput.*, May 1996, pp. 61–66.
- [25] A. Hatamlou, "Black hole: A new heuristic optimization approach for data clustering," *Inf. Sci.*, vol. 222, pp. 175–184, Feb. 2013.
- [26] R. S. Pavithr and Gursaran, "Quantum inspired social evolution (QSE) algorithm for 0-1 knapsack problem," *Swarm Evol. Comput.*, vol. 29, pp. 33–46, Aug. 2016.
- [27] Z. Woo Geem, J. Hoon Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, Feb. 2001.
- [28] Y. Shi, "Brain storm optimization algorithm," in *Advances in Swarm Intelligence*. Berlin, Germany: Springer, 2011, pp. 303–309.
- [29] M. Jahangiri, M. A. Hadianfard, M. A. Najafgholipour, M. Jahangiri, and M. R. Gerami, "Interactive autodidactic school: A new Metaheuristic optimization algorithm for solving mathematical and structural design optimization problems," *Comput. Struct.*, vol. 235, Jul. 2020, Art. no. 106268.
- [30] B. Das, V. Mukherjee, and D. Das, "Student psychology based optimization algorithm: A new population based optimization algorithm for solving optimization problems," *Adv. Eng. Softw.*, vol. 146, Aug. 2020, Art. no. 102804.
- [31] M. S. Gonçalves, R. H. Lopez, and L. F. F. Miguel, "Search group algorithm: A new Metaheuristic method for the optimization of truss structures," *Comput. Struct.*, vol. 153, pp. 165–184, Jun. 2015.
- [32] O. Tunca, S. Carbas, and I. Aydogdu, "Symbiotic organisms search based optimal design of steel rigid frames," in *Proc. Int. Congr. Comput. Mech. (9Gracm)*, Chania, Greece, 2018, p. 57.
- [33] S. Talatahari, "Symbiotic organisms search for optimum design of frame and grillage systems," *Asian J. Civil Eng. (Building Housing)*, vol. 17, no. 3, pp. 229–313, 2016.
- [34] A. E. Ezugwu, "Enhanced symbiotic organisms search algorithm for unrelated parallel machines manufacturing scheduling with setup times," *Knowl.-Based Syst.*, vol. 172, pp. 15–32, May 2019.
- [35] M. Abdullahi, M. A. Ngadi, S. I. Dishing, S. M. Abdulhamid, and B. I. Ahmad, "An efficient symbiotic organisms search algorithm with chaotic optimization strategy for multi-objective task scheduling problems in cloud computing environment," *J. Netw. Comput. Appl.*, vol. 133, pp. 60–74, May 2019.
- [36] M.-Y. Cheng, D. Prayogo, and D.-H. Tran, "Optimizing multiple-resources leveling in multiple projects using discrete symbiotic organisms search," *J. Comput. Civil Eng.*, vol. 30, no. 3, May 2016, Art. no. 04015036.
- [37] A. E. Ezugwu and D. Prayogo, "Symbiotic organisms search algorithm: Theory, recent advances and applications," *Expert Syst. Appl.*, vol. 119, pp. 184–209, Apr. 2019.

- [38] A. E.-S. Ezugwu and A. O. Adewumi, "Discrete symbiotic organisms search algorithm for travelling salesman problem," *Expert Syst. Appl.*, vol. 87, pp. 70–78, Nov. 2017.
- [39] S. Kumar, G. G. Tejani, and S. Mirjalili, "Modified symbiotic organisms search for structural optimization," *Eng. Comput.*, vol. 35, no. 4, pp. 1269–1296, Oct. 2019.
- [40] G. G. Tejani, V. J. Savsani, S. Bureerat, and V. K. Patel, "Topology and size optimization of trusses with static and dynamic bounds by modified symbiotic organisms search," *J. Comput. Civil Eng.*, vol. 32, no. 2, Mar. 2018, Art. no. 04017085.
- [41] G. G. Tejani, N. Pholdee, S. Bureerat, and D. Prayogo, "Multiobjective adaptive symbiotic organisms search for truss optimization problems," *Knowl.-Based Syst.*, vol. 161, pp. 398–414, Dec. 2018.
- [42] G. G. Tejani, V. J. Savsani, and V. K. Patel, "Adaptive symbiotic organisms search (SOS) algorithm for structural design optimization," *J. Comput. Design Eng.*, vol. 3, no. 3, pp. 226–249, Jul. 2016.
- [43] E. Çelik, "A powerful variant of symbiotic organisms search algorithm for global optimization," *Eng. Appl. Artif. Intell.*, vol. 87, Jan. 2020, Art. no. 103294.
- [44] D. T. T. Do and J. Lee, "A modified symbiotic organisms search (mSOS) algorithm for optimization of pin-jointed structures," *Appl. Soft Comput.*, vol. 61, pp. 683–699, Dec. 2017.
- [45] G. G. Tejani, V. J. Savsani, V. K. Patel, and S. Mirjalili, "Truss optimization with natural frequency bounds using improved symbiotic organisms search," *Knowl.-Based Syst.*, vol. 143, pp. 162–178, Mar. 2018.
- [46] A. E.-S. Ezugwu, A. O. Adewumi, and M. E. Frincu, "Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem," *Expert Syst. Appl.*, vol. 77, pp. 189–210, Jul. 2017.
- [47] D. Guha, P. Roy, and S. Banerjee, "Quasi-oppositional symbiotic organism search algorithm applied to load frequency control," *Swarm Evol. Comput.*, vol. 33, pp. 46–67, Apr. 2017.
- [48] D.-H. Tran, L. Luong-Duc, M.-T. Duong, T.-N. Le, and A.-D. Pham, "Opposition multiple objective symbiotic organisms search (OMOSOS) for time, cost, quality and work continuity tradeoff in repetitive projects," *J. Comput. Des. Eng.*, vol. 5, no. 2, pp. 160–172, Apr. 2018.
- [49] G. G. Tejani, N. Pholdee, S. Bureerat, D. Prayogo, and A. H. Gandomi, "Structural optimization using multi-objective modified adaptive symbiotic organisms search," *Expert Syst. Appl.*, vol. 125, pp. 425–441, Jul. 2019.
- [50] L. A. Shalabi, Z. Shaaban, and B. Kasasbeh, "Data mining: A pre-processing engine," *J. Comput. Sci.*, vol. 2, no. 9, pp. 735–739, Sep. 2006.
- [51] W. Pedrycz and X. Wang, "Designing fuzzy sets with the use of the parametric principle of justifiable granularity," *IEEE Trans. Fuzzy Syst.*, vol. 24, no. 2, pp. 489–496, Apr. 2016.
- [52] F. Camastra, A. Ciaramella, V. Giovannelli, M. Lener, V. Rastelli, A. Staiano, G. Staiano, and A. Starace, "A fuzzy decision system for genetically modified plant environmental risk assessment using mamdani inference," *Expert Syst. Appl.*, vol. 42, no. 3, pp. 1710–1716, Feb. 2015.
- [53] M. Mizumoto, "Fuzzy reasoning with various fuzzy inputs," *Inf. Sci.*, vol. 45, no. 2, pp. 129–151, 1988.
- [54] M.-Y. Cheng and D. Prayogo, "Fuzzy adaptive teaching–learning-based optimization for global numerical optimization," *Neural Comput. Appl.*, vol. 29, no. 2, pp. 309–327, Jan. 2018.
- [55] S. Mirjalili, "SCA: A sine cosine algorithm for solving optimization problems," *Knowl.-Based Syst.*, vol. 96, pp. 120–133, Mar. 2016.
- [56] M. Mitić, N. Vuković, M. Petrović, and Z. Miljković, "Chaotic fruit fly optimization algorithm," *Knowl.-Based Syst.*, vol. 89, pp. 446–458, Nov. 2015.
- [57] K. Z. Zamli, M. F. J. Klaib, M. I. Younis, N. A. M. Isa, and R. Abdullah, "Design and implementation of a t-way test data generation strategy with automated execution tool support," *Inf. Sci.*, vol. 181, no. 9, pp. 1741–1758, May 2011.
- [58] A. R. A. Alsewari and K. Z. Zamli, "Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support," *Inf. Softw. Technol.*, vol. 54, no. 6, pp. 553–568, Jun. 2012.
- [59] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' Behaviour: A case study on the CEC'2005 special session on real parameter optimization," *J. Heuristics*, vol. 15, no. 6, pp. 617–644, Dec. 2009.
- [60] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.
- [61] J. H. Zar, *Biostatistical Analysis*. London, U.K.: Pearson Education Inc, 2009.
- [62] G. G. Wang, S. Deb, and Z. Cui, "Monarch butterfly optimization," *Neural Comput. Appl.*, vol. 31, no. 7, pp. 1995–2014, 2019.
- [63] G. G. Wang, S. Deb, and L. D. S. Coelho, "Earthworm optimization algorithm: A bio-inspired Metaheuristic algorithm for global optimization problems," *Int. J. Bio-Inspired Comput.*, vol. 1, no. 1, p. 1, 2015.
- [64] G.-G. Wang, S. Deb, and L. D. S. Coelho, "Elephant herding optimization," in *Proc. 3rd Int. Symp. Comput. Bus. Intell. (ISCBI)*, Dec. 2015, pp. 1–5.
- [65] G.-G. Wang, "Moth search algorithm: A bio-inspired Metaheuristic algorithm for global optimization problems," *Memetic Comput.*, vol. 10, no. 2, pp. 151–164, Jun. 2018.



NURUL ASYIKIN ZAINAL received the B.Eng. degree in electronics (computer and information) from International Islamic University Malaysia, Kuala Lumpur, Malaysia, in 2010, and the M.S. degree in software engineering from Universiti Malaysia Pahang, Pahang, Malaysia, in 2016, where she is currently pursuing the Ph.D. degree in software engineering. From 2011 to 2020, she was a Design Engineer (Automation) with Malaysian Test Equipment Sdn Bhd, Pahang, Malaysia. Her research interests include the development of embedded systems, intelligent control, and also soft-computing in modeling and control.



SAIFUL AZAD (Member, IEEE) received the Ph.D. degree in information engineering from the University of Padova, Italy, in 2013. He is currently serving as a member for the Faculty of Computing, Universiti Malaysia Pahang, Malaysia. He is also the author of many scientific papers published in renowned journals and conferences. His research interests include data mining, machine learning, design and implementation of communication protocols, network security, and simulation software design. He is also a Fellow of the IBM Center of Excellence, Malaysia. He was a recipient of many national and international awards for his services, exhibitions, publications, and so on.



KAMAL Z. ZAMLI (Member, IEEE) received the degree in electrical engineering from the Worcester Polytechnic Institute, Worcester, MA, USA, in 1992, the M.Sc. degree in real-time software engineering from Universiti Teknologi Malaysia, in 2000, and the Ph.D. degree in software engineering from the University of Newcastle upon Tyne, U.K., in 2003. His research interests include search-based software engineering and computational intelligence.

...