ISSN: 1803-3814 (Printed), 2571-3701 (Online)
https://doi.org/10.13164/mendel.2021.1.029

Three Steps to Improve Jellyfish Search Optimiser

Petr Bujok✉
Department of Informatics and Computers, University of Ostrava, Czech Republic
petr.bujok@osu.cz✉

Abstract

This paper describes three different mechanisms used in Jellyfish Search (JS) optimiser. At first, an archive of good old solutions is used to prevent getting stuck in the local-optima area. Further, a distribution coefficient β is adapted during the search process to control population diversity. Finally, an Eigen transformation of individuals in the reproduction process is used occasionally to cope with rotated functions. Three proposed variants of the JS optimiser are compared with the original JS algorithm and nine various well-known Nature-inspired optimisation methods when solving real-world problems of CEC 2011. Provided results achieved by statistical comparison show efficiency of the individual newly employed mechanisms.

Keywords: *Global optimisation, Jellyfish Search optimiser, archive, Eigen transformation, distribution coefficient, experimental comparison.*

Received: 04 May 2021
Accepted: 11 June 2021
Published: 21 June 2021

1 Introduction

This paper proposes a new enhanced variant of a successful jellyfish search (JS) optimiser. A newly proposed method employs three various mechanisms to cope with various optimisation tasks, especially real-world problems.

The JS algorithm belongs to swarm optimisation methods which are inspired by biosystems from nature. Swarm algorithms are very popular in practical (real-world) applications in various areas of research, industry and healthcare services [7, 15].

The main goal of this paper is to propose, apply and compare a new variant of the JS algorithm. In this algorithm, three various enhancing elements are added to achieve higher performance. The variant of JS is applied on real-world problems to illustrate a real efficiency when it will be used in practice. Results of a new algorithm are compared with the original JS algorithm to detect a proper level of efficiency of the employed mechanisms. Moreover, a set of well-known swarm algorithms is used to evaluate the quality of the proposed method in a more practical way.

The rest of the text is organised as follows. Section 2 describes a basic ideas of the original JS algorithm. In section 3 introduces a newly proposed JS variant and its main features. Section 4 briefly describes other swarm methods used in a comparison. The main ideas of the performed experiment are assumed in Section 5. Tables, plots and statistical analysis are discussed in Section 6. Finally, the overall evaluation of the proposed method is concluded in Section 7.

2 Jellyfish Search Optimiser

In 2021, Chou et al. introduced a novel meta-heuristic optimisation technique inspired by the behaviour of jellyfish in oceans called Jellyfish Search (JS) optimiser [5]. The relative complex behaviour of real jellyfish in nature was mathematically summed to two simple rules.

1. Jellyfish follows streams in the ocean or moves inside a swarm. These movements are controlled by 'time control mechanism'.
2. Simply, jellyfish follows areas in the ocean where a greater amount of nurture is available.

The ocean stream attracts jellyfish because it contains nurture. Therefore, a new position of the ith jellyfish in the current phase is influenced by the trend of the ocean stream (1):

$$\boldsymbol{y}_i = \boldsymbol{x}_i + rand \cdot (\boldsymbol{x}_{\text{best}} - \beta \cdot rand \cdot \frac{\sum_{i=1}^{N} \boldsymbol{x}_i}{N}) \qquad (1)$$

where $\boldsymbol{x}_i$ is the current position of the jellyfish, $\boldsymbol{x}_{\text{best}}$ is the position of the current best jellyfish in the swarm, N is swarm size, and β is a distribution coefficient ($\beta > 0$, $\beta = 3$).

Further, jellyfish use two kinds of motion in swarm – passive (2) and active (3). Passive motions are used in the preliminary stages of swarm formation and are represented by local motions around jellyfish position.

Passive motions:

$$\boldsymbol{y}_i = \boldsymbol{x}_i + \gamma \cdot rand \cdot (\boldsymbol{b} - \boldsymbol{a}) \qquad (2)$$

where γ ($\gamma > 0$) represents a motions coefficient, and $\boldsymbol{b}$ and $\boldsymbol{a}$ are the upper and lower bounds of the search space.

Active motions:

$$\boldsymbol{y}_i = \boldsymbol{x}_i + rand \cdot \begin{cases} \boldsymbol{x}_j - \boldsymbol{x}_i & \text{if } f(\boldsymbol{x}_j) \le f(\boldsymbol{x}_i) \\ \boldsymbol{x}_i - \boldsymbol{x}_j & \text{if } f(\boldsymbol{x}_j) > f(\boldsymbol{x}_i). \end{cases} \quad (3)$$

where $\boldsymbol{x}_j$ is randomly selected jellyfish from swarm, different from the current jellyfish $\boldsymbol{x}_i$, and f represents an objective function.

Finally time control mechanism enables simulate regulation between jellyfish swarm in ocean and jellyfish in a swarm by time control function c and constant c_0. When $c > c_0$ jellyfish follows ocean stream, otherwise jellyfish moves inside a swarm (4):

$$c = \left| \left(1 - \frac{FES}{maxFES}\right) \cdot (2 \cdot rand - 1) \right| \quad (4)$$

where FES is a current number and $maxFES$ is a total available number of function evaluations of jellyfish individuals in one algorithm run.

Results of the JS algorithm compared with ten various evolutionary algorithms (including Differential evolution, Particle swarm optimisation or Tree seed algorithm) show superiority of JS in 25 problems of CEC 2005.

Because of the good performance of the JS algorithm, several various studies of the real application of JS were proposed to illustrate the efficiency and simplicity of this method.

In 2021, Gouda et al. propose an experimental study presenting an application of the JS algorithm for extracting unknown parameters of PEM fuel cell models [8]. Performance of JS is employed to determine PEM model parameters where three different models are used in experiments. JS outperforms other optimisation methods in comparison.

In 2021, Abdel-Basset et al. employed an enhanced variant of JS to identify parameters of Photovoltaic models [1]. The authors applied a new Premature convergence strategy to increase JS exploitation capability. Results achieved on four different models illustrate the superiority of JS compared with other employed known optimisation methods.

In 2021, Selvakumar et al. used the JS algorithm to help a Spectrum Defragmentation algorithm to utilise an Elastic Optical Network (EON) [13]. A comparison of the proposed algorithm with state-of-the-art spectrum defragmentation algorithms illustrate better utilisation of spectrum and reduce fragmentation complexity.

Despite promising results of JS in a set of CEC 2005 problems [5] and successful aforementioned enhanced JS variants, in this paper, a new JS variant is proposed to increase performance on real-world problems.

3 A Novel Jellyfish Search Algorithm

The original JS algorithm provides a good performance in spite of its simplicity and clarity. The results are very promising compared to simple optimisation techniques, a comparison with more advanced methods illustrates weak parts of the JS algorithm. In this paper, a novel variant of JS is proposed, based on three independent enhancing approaches. Newly employed mechanisms are gradually incorporated into JS to study a significance of an individual approach. A pseudo-code of a new JS variant is illustrated in Algorithms 1.

Algorithm 1 Improved JS algorithm

```
1:  initialise and evaluate $P = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$
2:  initialise empty archive $A$                          ▷ Comment1
3:  initialise interval for diversity control            ▷ Comment2
4:  while stopping condition not reached do
5:      if $rand < peig$ then
6:          use Eigen coordinate system                   ▷ Comment3
7:      else
8:          use standard coordinate system
9:      if $c \ge 0.5$ then
10:         Jellyfish follows ocean stream
11:     else
12:         if $rand > (1 - c)$ then
13:             Jellyfish passive motions
14:         else
15:             Jellyfish active motions
16:     evaluate and update population
17:     update distribution coefficient $\beta$           ▷ Comment4
```

The pseudo-code is based on steps of the original JS algorithm where newly added parts are noted by the 'Comment' symbol.

3.1 Archive of Good Solutions

In the original JS, the current position of the jellyfish is replaced by the new position of the jellyfish only if the new position is better than the current position. Then, the current position is dismissed. This approach promises faster convergence, but it can increase the probability to get stuck in the algorithm in a local solution (minimum) area.

Therefore, the first enhancing mechanism used in JS is the archive of good old solutions (positions) to store the solutions for usage in the next generations. The idea of the archive is very simple. The archive A is introduced as empty, and in every selection step, when the current and new jellyfish positions are compared, the old (current) solution is inserted into A if the new position is better ($f(\boldsymbol{y}_i < \boldsymbol{x}_i)$). When the archive of size N is fully filled, the newly inserted old good solution is located to a random position of A.

Notice that storing the individuals into A is performed in all three phases of JS. Using the individuals from A is only in the ocean stream (current) phase (eq. (1) for computing the average coordinates of the jellyfish population. In the newly proposed JS variant, the average vector is computed from the union of population P and archive A. It promises a lower speed of convergence (older good positions are used) but a higher probability to avoid the local solution area.

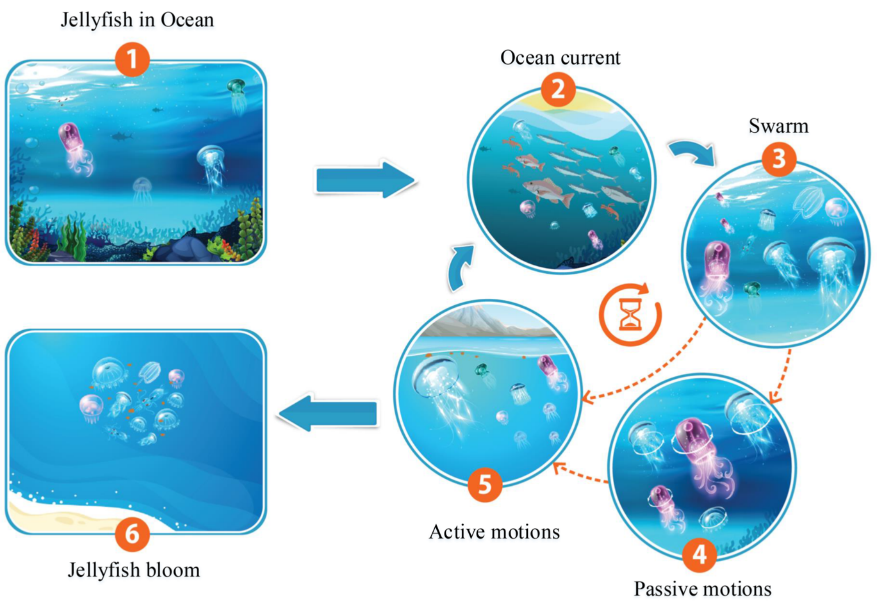

Figure 1: Illustration of jellyfish behaviour in ocean [5]

3.2 Dynamic Diversity Control

The parameters of the original JS algorithm are fine-tuned to achieve the best possible results in various optimisation problems. It is also the case of diversity coefficient β used in the ocean stream phase (eq. (1)). The authors recommended set $\beta = 3$, and this setting achieves very promising results. In fact, distribution coefficient β influences the amount of diversity of population used to determine a new jellyfish position. A higher value of β means a new position farther from the current best position, and vice versa. Naturally, when the diversity of the population is small, higher β helps to generate new positions in more distance from the best solution, and vice versa.

This idea was transformed in adaptation of distribution coefficient during the search based on current diversity of jellyfish population. This diversity is simply estimated by rule (5):

$$Di = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{D}(x_{ij} - \bar{x}_j)^2}\,, \quad \bar{x}_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij} \tag{5}$$

where D represents dimensionality of the problem, and N is population size. At the beginning of the algorithm, β is set to $\beta = 3$, and after each generation, it is updated based on current diversity as follows. The optimal relative diversity of the population is estimated as linearly decreasing from introduced random population to a zero diversity at the end of the run. Then, value of β is increased by one if the relative diversity (compared to randomly generated population) is decreases under 90% of the optimal value and β is decreased by one if relative diversity increases over 110% of optimal value. Notice that maximal and minimal possible values of β are 10 and 1. More detail about the diversity-based adaptation approach is in [12].

3.3 Eigen Transformation

The last step to increase the efficiency of the JS algorithm is based on the Eigen transformation of coordinates. This approach was introduced in [16] to transform parent individual before crossover operation in Differential evolution algorithm. Using the Eigen transformation enables to increase efficiency on rotated functions.

The idea of Eigen transformation in the JS algorithm is simple and it is controlled by two parameters – *ps* and *peig*. The value of *ps* controls portion of the population P used for compute Eigenvectors B (6):

$$C = BD^2B^T \tag{6}$$

where C is covariance matrix of the selected part ($ps \cdot N$) of population and D are Eigenvalues. Then, the Eigenvectors are employed to transform the original coordinates of selected individuals.

In each JS generation, either the original coordinate system or the new Eigen transformed coordinate system is applied, based on the second control parameter *peig*. If *rand* < *peig*, the Eigen transformation is applied in whole generation, in other cases, the standard coordinate system is used.

If the Eigen transformation is selected, only several following steps of JS are influenced by this transformation. In ocean stream phase (eq. (1)), when the new position $\boldsymbol{y}_i$ is computed, all three individuals, ie. $\boldsymbol{x}_i$, $\boldsymbol{x}_{\text{best}}$, and average population vector are transformed by Eigen transformation (7) (an example of the current jellyfish position $\boldsymbol{x}_i$):

$$\boldsymbol{x}_i^{'} = B^T \boldsymbol{x}_i. \tag{7}$$

Then, after the position updating rule is performed the new position is transformed from the Eigen coordinate system back to the original coordinate system using (8):

$$\boldsymbol{y}_i = B\boldsymbol{y}_i^{'}. \quad (8)$$

Similarly, in the active motions phase, when the Eigen transformation is selected, both $\boldsymbol{x}_i$, $\boldsymbol{x}_j$ are transformed, and also $\boldsymbol{x}_i$ is transformed in passive motions phase.

Based on the aforementioned description, the newly proposed variant of JS using all three enhancing mechanisms is called *JSeigDiA* ('eig' for Eigen transformation, 'Di' for distribution coefficient adaptation, and 'A' for the archive of good old solutions). Moreover, the efficiency of the archive is studied when a variant of *JSeigDi* (without the archive) is also applied. Finally, adaptive control of distribution coefficient β is evaluated when it is removed, and *JSeig* variant is also used in experiments. Therefore, three gradual variants of enhanced JS are proposed, applied and compared to achieve more complex conclusions.

4 Nature-Inspired Methods in Comparison

In the experiments, not only the original JS and its enhanced variants are used. Nine various well-known nature-inspired algorithms are employed to make a more complex comparison of newly proposed JS variants. The description of the methods is presented in chronological order.

Particle swarm optimisation (PSO) was introduced in 1995 and it is very popular swarm-intelligence inspired optimiser. In this experiment, the enhanced PSO variant with particle velocities mechanism, controlled by the variation coefficient w and coefficient c is used [14]. The value of w is set as a linear interpolation from $w_{\max} = 1$ to $w_{\min} = 0.3$ during the search. The value of parameter balancing between a local and a global part of the updated velocity is set $c = 1.05$. Velocity is for next generation computed as $\boldsymbol{v}_{i,G+1} = w_{G+1} \times \boldsymbol{v}_{i,G} + c\ U(0,1)\ (\boldsymbol{p}_{\text{best}} - \boldsymbol{x}_i) + c\ U(0,1)\ (\boldsymbol{g}_{\text{best}} - \boldsymbol{x}_i)$, where G represents generation, $U(0,1)$ a random number generated from uniform distribution, $\boldsymbol{x}_i$ is current position of particle, $\boldsymbol{p}_{\text{best}}$ is up-to-now the best position of the current particle, and $\boldsymbol{g}_{\text{best}}$ is the best particle in swarm history. In 2018, an advance cooperative variant of PSO with the Firefly algorithm (called HPSO) was proposed [2]. The parameters of HPSO are set to recommended values $\alpha = 0.2$, $\beta_0 = 2$, $\gamma = 1$, and $c_1 = c_2 = 1.49445$.

In 2000, the self-organising migration algorithm (SOMA) was proposed as a model of a pack of predators hunting the prey [20]. The best settings of the control parameters of SOMA used in this experiment were found in preliminary experiments. The length of the jump of an individual from the leader is set $\mathit{PathLenght} = 2$, the step size is $\mathit{Step} = 0.11$, and perturbation is $\mathit{Prt} = 0.1$. The best performing strategy was *all-to-one*, and it is applied to a comparison.

In 2005, the artificial bee colony algorithm (denoted ABC) was proposed, and it is a very popular optimisation technique. The only input parameter is called $\mathit{limit}= N$, which represents a maximum number of unsuccessful new food positions necessary to find a new random food position [9].

In 2009, the cuckoo search algorithm (denoted CS) modelling cuckoos' breeding strategy was introduced [19]. The probability of the cuckoo's eggs laid in a host-bird nest is $pa = 0.25$, and the control parameter of Lévy flight random walk is set to $\lambda = 1.5$.

The bat algorithm (BAT hereafter) uses parameter settings that follow the original publication [17]. Maximal and minimal frequencies are set up $f_{\max} = 2$, $f_{\min} = 0$, the local-search loudness parameter is initialised $A_i = 1.2$ for each bat-individual and then reduced if a new bat position is better than the old one using coefficient $\alpha = 0.9$. The emission rate parameter is initialised for each bat-individual $r_i = 0.1$ and increased by parameter $\gamma = 0.9$ in the case of a successful offspring.

In 2014, the firefly algorithm (FFL in the experiments), was introduced as the model of real fireflies [18]. The control parameters are set to recommended values, randomisation parameter $\alpha = 0.5$, light absorption coefficient $\gamma = 1$, and attractiveness is updated using its initial and minimal values $\beta_0 = 1$, $\beta_{\min} = 0.2$.

In 2014, the grey wolf optimiser (labelled GWO in results) representing hunting and hierarchic behaviour of grey wolves was introduced [11]. GWO hierarchical strategy employs four different kinds of wolves - alpha, beta, delta, and omega. The main role of wolves-individuals is hunting, which is composed of the three-level process - search, gird and attack the prey. Besides population size, the only control parameter of GWO is component a, and it decreases linearly from 2 to 0.

In 2015, Kiran proposed a new optimisation algorithm inspired by the relation between trees and seeds called Tree-seed (TSA) algorithm [10]. The TSA algorithm models the processes of trees producing the seed in nature, where the ground is defined as search space. Each tree is able to generate a given number of seeds. In TSA, ns seeds for each tree are randomly selected number between $10 - 25\%$ of the number of trees (N), i.e. $ns = round((0.1 + 0.15 * rand) * N)$. The process of the new seed's position (S) distribution is controlled by the parameter ST.

In 2020, Bujok proposed a novel variant of TSrAeig, which significantly outperformed the original TSA [3]. TSrAeig uses an archive of never-used newly generated solutions. Further, after each generation of each tree, all seeds better than the tree are located to pool S_B, and randomly selected seed from S_B replaces the tree coordinates. Finally, the Eigen transformation is used for some tree individuals to achieve better results in rotated functions.

5 Experimental Settings

At first, a test suite of 22 real-world problems of the CEC 2011 competition in the Special Session on Real-Parameter Numerical optimisation is used [6]. The functions differ in the computational complexity and the dimension of the search space from $D = 1$ to $D = 240$, where the dimensionality of most problems exceeds $D = 20$. The experimental settings required for this set is used in our experimental comparison. For each algorithm and problem, 25 independent runs were performed. The run of the algorithm is stopped when the prescribed number of function evaluation $MaxFES = 150000$ is reached. The partial results after one third and two-thirds of MaxFES are also studied. The solution of the problem is given by a point in the terminal population with the smallest function value. It is due to the correct solution of real-world problems is unknown.

The population size of compared algorithms in comparison is set to the best possible values recommended by authors or achieved in previous huge comparisons [4]. Therefore $N = 50$ is for JS, JSeig, JSeigDi, JSeigDiA, TSA and TSrEigA. $N = 30$ is for GWO and HFPSO. $N = 90$ is for SOMA and FFL. ABC uses $N = 125$, CS set $N = 15$, and $N = 40$ is for PSO.

Parameters of Eigen transformation used in JSeig, JSeigDi, and JSeigDiA are set $ps = 0.5$ and $peig = 0.4$. Remaining parameters of the algorithms are set to values recomended by authors. All the algorithms are implemented in Matlab 2020b, where also statistical analysis is assessed. All computations were carried out on a standard PC with Windows 7, Intel(R) Core(TM)i7-4700 CPU 3.0 GHz, 16 GB RAM.

6 Results

In the experiment, three variants of newly proposed JS algorithm are compared with the original JS and nine various nature-inspired methods on set of 22 real-world problems.

At first, the Friedman test is applied to provide an overall insight into the comparison of the algorithms' performance. The test is applied on medians of achieved minimum values at three stages of the search (FES = 50,000, 100,000, and 150,000) for the CEC 2011 set. The results are presented in Table 1 where also the absolute ranks are depicted in brackets. The null hypothesis on equivalent efficiency of the methods is rejected with $p < 5 \times 10^{-6}$. The algorithms in these tables are ordered from left to right based on the mean ranks from the Friedman test at the end of the search for CEC 2011 and regarding average ranks overall dimensions.

It is obvious that the newly proposed JSeigDiA, JSeigSi, and JSeig outperform other algorithms in the experiment. The original JS algorithm takes the eighth position what underlines the significant performance of the enhancing mechanisms. Besides JS, very nice results achieve the original CS algorithm (outperforms enhanced HFPSO and TSrEigA).

When studying the mean ranks of the proposed JS variants, it is clear that using all three mechanisms is the most efficient choice. Interesting are ranks of the remaining two proposed algorithm, where better mean rank achieve variant without adaptation of distribution coefficient (JSeig). But differences between the proposed three JS variants are relatively small.

Further, a comparison of the methods on each real-world problem independently is performed using the Kruskal-Wallis test. For each problem, the null hypothesis was rejected (significance level was less than 1×10^{-10}), and mean-ranks of the methods are provided. Then, the number of problems, where the methods achieve the best result (denoted 1st), the second-best result (2nd), the third-best result (3rd), and the worst result (last), are provided for each algorithm in comparison (see Table 2). Moreover, columns of the table are ordered from left to right based on the numbers. The first three positions are the same as in the Friedman test, newly proposed JS variants achieve the best performance in comparison (they achieve best results in 4, 3, and 3 problems out of 22, and they are never the worst). The superiority of the employed mechanisms (compared to the original JS) is obvious.

Finally, the proposed the best performing JSeigDiA is compared with each of twelve remaining algorithms using the Wilcoxon rank-sum test. The achieved results are divided into three tables, Table 3, 4, and 5. In the second column, the median values of the best performing algorithm for each problem are presented. In remaining columns, medians of other algorithms from comparison are depicted with a specific symbol of achieved significance in brackets. Symbol of '$\approx$' denotes similar results of the best and compared methods. Symbols of '+' ($0.01 < sig. < 0.05$), '++' ($0.001 < sig. < 0.01$) or '+++' ($sig. < 0.001$) mean significantly better results for the compared method. Finally, symbols of '-' ($0.01 < sig. < 0.05$), '- -' ($0.001 < sig. < 0.01$) or '- - -' ($sig. < 0.001$) mean significantly better results for the best performing (JSeigDiA) method. For better overview, the number of achieved significant or similar differences are computed in the last rows of the tables ($+/\approx/-$).

The best performing JSeigDiA is outperformed by other methods in comparison from zero to five problems. Significantly better results achieves JSeigDiA from 2 problems (JSeig and JSeigDi) to 20 problems (FFL). All three newly proposed JS variants use Eigen transformation. Therefore median values of success of the standard JS approach and Eigen transformation approach are computed in Table 6. The presented numbers represent the number of new jellyfish positions better than the currently occupied positions. The more efficient strategy (standard or Eigen) is for each studied algorithms print bold. For the Eigen approach, the percentage of achieved success is printed in a bracket. In the last row of the table, the count of better results is illustrated. It is clear that the efficiency of the Eigen transformation is similar to the original approach (measured by the number of higher values). Also in problems from T01 to T09, T11.1, and T11.2 are obvious percentage values about 40% what follows de-

Table 1: Mean ranks and absolute ranks of all algorithms from the Friedman tests.

FES	JSeigDiA	JSeig	JSeigDi	CS	HFPSO	TSrEigA
50000	4.3 (1)	4.5 (2)	4.7 (3)	5.3 (4.5)	5.3 (4.5)	5.8 (6)
100000	4.2 (2)	4.1 (1)	4.8 (3)	5.9 (5)	5.7 (4)	6.3 (6)
150000	4.3 (1)	4.3 (2)	4.7 (3)	6 (4)	6.1 (5)	6.3 (6)
GWO	JS	SOMA	ABC	PSO	TSA	FFL
9.8 (11)	6.2 (7)	6.5 (8)	8.1 (9)	8.4 (10)	10 (12)	12.1 (13)
8.8 (11)	6.4 (7)	6.7 (8)	8 (9)	8.4 (10)	9.5 (12)	12.3 (13)
6.4 (7)	6.4 (8)	7.2 (9)	8.2 (10)	9 (11)	9.6 (12)	12.5 (13)

Table 2: Number of first, second, third, and last positions of each algorithm.

position	JSeigDiA	JSeig	JSeigDi	HFPSO	CS	ABC
1st	4	3	3	2	2	2
2nd	2	5	3	2	2	0
3rd	5	2	4	4	1	1
last	0	0	0	0	0	1
TSrEigA	JS	GWO	SOMA	PSO	TSA	FFL
1	1	1	1	0	0	0
2	2	1	1	0	0	0
1	0	1	1	0	0	0
0	0	0	0	0	2	18

Table 3: Medians and significance from Wilcoxon rank-sum tests between the best *JSeigDiA* and other methods.

fun	JSeigDiA	JS	JSeigDi	JSeig	ABC
T01	4.46E-11	0.76059 (≈)	7.63E-07 (≈)	6.42531 (≈)	12.262 (- - -)
T02	-20.7965	-19.3604 (≈)	-17.336 (-)	-20.1539 (≈)	-20.1491 (≈)
T03	1.15E-05	1.15E-05 (≈)	1.15E-05 (≈)	1.15E-05 (≈)	1.15E-05 (≈)
T04	13.7708	13.7708 (≈)	13.7708 (≈)	13.7708 (≈)	14.3647 (- - -)
T05	-26.5427	-25.3325 (≈)	-25.9431 (≈)	-26.3003 (≈)	-35.5302 (+++)
T06	-21.3333	-20.589 (≈)	-21.5207 (≈)	-20.6392 (≈)	-26.4367 (+++)
T07	1.52953	1.51177 (≈)	1.54214 (≈)	1.50031 (≈)	1.41103 (++)
T08	220	220 (≈)	220 (≈)	220 (≈)	220 (≈)
T09	28845.2	2928.43 (- - -)	37463.9 (-)	29361.6 (≈)	88653.4 (- - -)
T10	-21.4661	-21.2862 (- - -)	-21.6011 (≈)	-21.5434 (≈)	-16.219 (- - -)
T11.1	52157.2	52446.1 (-)	52393.1 (≈)	52095.1 (≈)	58276.2 (- - -)
T11.2	1.14E+06	1.19E+06 (-)	1.11E+06 (≈)	1.08E+06 (≈)	2.68E+06 (- - -)
T11.3	15444.2	15451.4 (- - -)	15444.2 (≈)	15444.2 (≈)	15459.9 (- - -)
T11.4	18266.8	19127 (- - -)	18256.2 (≈)	18231.6 (≈)	19308.5 (- - -)
T11.5	32842.5	33053.4 (- - -)	32859.3 (≈)	32839.4 (≈)	33115.5 (- - -)
T11.6	135661	140902 (- - -)	134903 (≈)	135071 (≈)	139131 (- - -)
T11.7	1.93E+06	1.97E+06 (- - -)	1.94E+06 (≈)	1.95E+06 (-)	1.34E+08 (- - -)
T11.8	944415	944419 (≈)	942701 (+)	943848 (≈)	2.59E+07 (- - -)
T11.9	1.21E+06	1.36E+06 (- -)	1.27E+06 (≈)	1.20E+06 (≈)	2.44E+07 (- - -)
T11.10	943615	945904 (-)	943039 (≈)	944003 (≈)	2.66E+07 (- - -)
T12	14.9037	15.4036 (≈)	15.1054 (≈)	15.7421 (-)	18.2716 (- - -)
T13	22.6246	22.6895 (≈)	22.8781 (≈)	22.6796 (≈)	20.4791 (+++)
Σ		0/11/11	1/19/2	0/20/2	4/3/15

fined *peig* = 0.4. But in many problems, the success of Eigen transformation is more than 90% (99% for T11.4). This is very interesting information for further development of JS variants. Despite significant differences between algorithms in comparison, there are many problems where several algorithms achieved the same or very similar results. For these cases, the best solution achieved by each algorithm for each real-world problem is recorded in 17 stages of the search process to illustrate the convergence ability of compared methods. Further, median values of the recorded solutions from independent runs are computed. Figs. 2, 3 and 4 depicted convergence lines of all 13 algorithms for each problem individually. It is clear that the worst ability to converge provides FFL. Similarly, although ABC provides promising results in several problems, it

Table 4: Medians and significance from Wilcoxon rank-sum tests between the best *JSeigDiA* and other methods.

fun	JSeigDiA	CS	PSO	GWO	SOMA
T01	4.46E-11	8.41609 (≈)	12.8765 (- - -)	11.6669 (- - -)	11.7076 (- - -)
T02	-20.7965	-17.1041 (- - -)	-15.3444 (- - -)	-23.807 (+++)	-18.7298 (≈)
T03	1.15E-05	1.15E-05 (≈)	1.15E-05 (≈)	1.15E-05 (≈)	1.15E-05 (≈)
T04	13.7708	13.9285 (- -)	14.3291 (- - -)	13.8308 (- - -)	14.3291 (- - -)
T05	-26.5427	-34.1348 (+++)	-31.2519 (+++)	-34.1542 (+++)	-32.3813 (+++)
T06	-21.3333	-25.5777 (+++)	-22.8628 (≈)	-23.0059 (++)	-27.4297 (+++)
T07	1.52953	1.14084 (+++)	1.45337 (≈)	0.77200 (+++)	1.11902 (+++)
T08	220	220 (≈)	220 (≈)	220 (≈)	220 (≈)
T09	28845.2	3152.87 (+++)	429738 (- - -)	24326.4 (≈)	221914 (—)
T10	-21.4661	-21.5453 (≈)	-15.917 (- - -)	-12.9574 (- - -)	-16.8617 (- - -)
T11.1	52157.2	52390.4 (- -)	1.47E+06 (- - -)	474417 (- - -)	1.27E+06 (- - -)
T11.2	1.14E+06	1.13E+06 (≈)	4.78E+06 (- - -)	1.12E+06 (≈)	3.64E+06 (- - -)
T11.3	15444.2	15444.2 (≈)	15457.2 (- - -)	15463.6 (- - -)	15461.2 (- - -)
T11.4	18266.8	18666.1 (- - -)	18860.7 (- - -)	19210.7 (- - -)	19219.6 (- - -)
T11.5	32842.5	33038.5 (- - -)	33134.5 (- - -)	32993 (- - -)	32863.6 (≈)
T11.6	135661	140929 (- - -)	140281 (- - -)	136626 (-)	133372 (+)
T11.7	1.94E+06	1.94E+06 (≈)	2.28E+06 (- - -)	2.49E+06 (- - -)	1.98E+06 (- - -)
T11.8	944415	962556 (- - -)	955368 (- - -)	956785 (- - -)	980383 (- - -)
T11.9	1.21E+06	1.54E+06 (- - -)	1.44E+06 (- - -)	1.30E+06 (≈)	1.32E+06 (≈)
T11.10	943615	958510 (- - -)	1.04E+06 (- - -)	957703 (- - -)	988279 (- - -)
T12	14.9037	20.639 (- - -)	21.1637 (- - -)	24.3415 (- - -)	18.0423 (- - -)
T13	22.6246	22.1331 (≈)	25.3219 (- - -)	21.7653 (≈)	22.8158 (≈)
Σ		4/8/10	1/4/17	4/6/12	4/6/12

Table 5: Medians and significance from Wilcoxon rank-sum tests between the best *JSeigDiA* and other methods.

fun	JSeigDiA	HFPSO	FFL	TSA	TSrAeig
T01	4.46E-11	14.5269 (- - -)	28.4112 (- - -)	1.53821 (≈)	11.4902 (- -)
T02	-20.7965	-26.1267 (+++)	-12.6795 (- - -)	-5.16811 (- - -)	-18.3032 (≈)
T03	1.15E-05	1.15E-05 (≈)	1.15E-05 (≈)	1.15E-05 (≈)	1.15E-05 (≈)
T04	13.7708	14.3291 (- - -)	19.2793 (- - -)	13.9016 (- - -)	14.3291 (- - -)
T05	-26.5427	-33.6333 (+++)	-18.2279 (- - -)	-20.8928 (- - -)	-32.0809 (+++)
T06	-21.3333	-26.5001 (+++)	-14.7311 (- - -)	-17.6841 (- - -)	-23.0059 (≈)
T07	1.52953	0.87531 (+++)	2.37051 (- - -)	1.69353 (- - -)	1.0633 (+++)
T08	220	220 (≈)	350.555 (- - -)	220 (≈)	220 (≈)
T09	28845.2	18356.8 (≈)	3.18E+06 (- - -)	142246 (- - -)	2145.66 (+++)
T10	-21.4661	-20.4018 (- - -)	-7.59723 (- - -)	-18.6144 (- - -)	-12.6498 (- - -)
T11.1	52157.2	52296.4 (≈)	2.50E+06 (- - -)	8.47E+07 (- - -)	160929 (- - -)
T11.2	1.14E+06	1.08E+06 (+)	7.33E+06 (- - -)	5.64E+06 (- - -)	1.08E+06 (≈)
T11.3	15444.2	15480 (- - -)	52975.7 (- - -)	15460.2 (- - -)	15470.4 (- - -)
T11.4	18266.8	19360.3 (- - -)	19243 (- - -)	19214.7 (- - -)	19008.2 (- - -)
T11.5	32842.5	33007.9 (- - -)	495711 (- - -)	32947 (- - -)	32922.1 (- -)
T11.6	135661	145074 (- - -)	7.78E+07 (- - -)	136988 (≈)	138291 (≈)
T11.7	1.94E+06	1.95E+06 (-)	1.48E+10 (+++)	3.12E+06 (- - -)	1.96E+06 (- -)
T11.8	944415	956643 (- - -)	1.47E+08 (- - -)	2.66E+06 (- - -)	953995 (- - -)
T11.9	1.21E+06	1.13E+06 (≈)	1.58E+08 (- - -)	3.43E+06 (- - -)	1.40E+06 (- -)
T11.10	943615	951985 (- - -)	1.48E+08 (- - -)	2.65E+06 (- - -)	949856 (- - -)
T12	14.9037	14.9149 (≈)	46.1224 (- - -)	28.3102 (- - -)	18.4757 (- - -)
T13	22.6246	24.5003 (≈)	39.8236 (- - -)	26.1825 (- - -)	20.1104 (++)
Σ		5/7/10	1/1/20	0/4/18	4/6/12

often converges very slowly. Fast convergence ability is also observed for hybrid HFPSO. In many cases, it gets stuck in the local minima area (it is outperformed by other algorithms in comparison). In problems where newly proposed algorithms are outperformed by some methods in comparison, it is obvious a significantly decreasing trend of the convergence curve for proposed methods and constant convergence curves for remaining methods (T05, T06). It promises better results of the proposed JS variants when a search time is increased.

Table 6: Efficiency of Eigen transformation in JSeig, JSeigDi, and JSeigDiA.

	JSeig		JSeigDi		JSeigDiA	
fun	std	eig (%)	std	eig (%)	std	eig (%)
T01	**7340**	4792 (39)	**5815**	3575 (38)	**6942**	4251 (38)
T02	**969**	549 (36)	**789**	479 (38)	**1174**	662 (36)
T03	**766**	511 (40)	**780**	513 (40)	**773**	522 (40)
T04	**2891**	1938 (40)	**2819**	1884 (40)	**2860**	1921 (40)
T05	**503**	448 (47)	**498**	430 (46)	**500**	405 (45)
T06	**712**	597 (46)	**687**	553 (45)	**727**	566 (44)
T07	262	**281** (52)	265	**308** (54)	265	**288** (52)
T08	**351**	136 (28)	**354**	126 (26)	**351**	132 (27)
T09	**37669**	26492 (41)	**37471**	26316 (41)	**36860**	26633 (42)
T10	7569	**23385** (76)	7192	**22842** (76)	7220	**23258** (76)
T11.1	**19336**	13525 (41)	**19233**	13081 (40)	**19605**	13804 (41)
T11.2	**38106**	27326 (42)	**38398**	27334 (42)	**38440**	27623 (42)
T11.3	345	**6151** (95)	339	**4720** (93)	350	**5425** (94)
T11.4	113	**6931** (98)	122	**7412** (98)	119	**7841** (99)
T11.5	823	**20523** (96)	842	**20404** (96)	802	**19461** (96)
T11.6	408	**3357** (89)	378	**3401** (90)	374	**3510** (90)
T11.7	904	**3108** (77)	840	**2541** (75)	787	**2394** (75)
T11.8	1001	**2163** (68)	982	**2349** (71)	975	**2388** (71)
T11.9	1147	**6606** (85)	1131	**6103** (84)	1072	**6847** (86)
T11.10	970	**2312** (70)	981	**2482** (72)	954	**2395** (72)
T12	**8257**	7508 (48)	7696	**8452** (52)	10658	**11589** (52)
T13	7508	**14263** (66)	6891	**15190** (69)	6979	**15756** (69)
Σ	11	11	10	12	10	12

7 Conclusion

In this paper, a new enhanced variant of the successful jellyfish search optimiser is proposed. In this algorithm, three independent mechanisms are applied to increase the ability to solve various real-world optimisation problems. Such, three gradual versions of the proposed JS variant are compared – the complete JSeigDiA variant, the variant without archive (JSeigDi), and the variant without archive and adaptation of the distribution coefficient (JSeig). These gradual variants enable the study efficiency of the applied enhancing mechanisms, besides the original JS algorithm, nine various nature. Inspired methods are used to evaluate the performance of the newly proposed algorithm on a set of 22 real-world problems CEC 2011.

Presented results of performed data illustration and data analysis are depicted in tables and figures. We can assume that the newly proposed JS variant significantly increases the performance of the original JS algorithm. Also, regarding all 22 problems, new algorithm variants significantly outperform nine employed nature-inspired methods. Studying the algorithms' ability to converge to the true solution, the proposed method is not the fastest method, but in many problems, the trend of convergence of new JS variant is decreasing compared to the rather constant trend of other algorithms in comparison. Comparison of success of the standard JS approach with the newly used Eigen transformation illustrates balanced results. But in many problems, the Eigen approach achieves success up to 99% (T11.4).

The achieved results very promising, and they will be used in future extensions of JS and other nature-inspired algorithms.

References

[1] Abdel-Basset, M., Mohamed, R., Chakrabortty, R. K., Ryan, M. J., and El-Fergany, A. An improved artificial jellyfish search optimizer for parameter identification of photovoltaic models. *Energies 14*, 7 (2021).

[2] Aydilek, I. B. A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems. *Applied Soft Computing 66* (2018), 232–249.

[3] Bujok, P. Enhanced tree-seed algorithm solving real-world problems. In *2020 7th International Conference on Soft Computing Machine Intelligence (ISCMI)* (2020), pp. 12–16.

[4] Bujok, P., Tvrdik, J., and Polakova, R. Comparison of nature-inspired population-based algorithms on continuous optimisation problems. *Swarm and Evolutionary Computation 50* (2019), 100490.

[5] Chou, J.-S., and Truong, D.-N. A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean. *Applied Mathematics and Computation 389* (2021), 125535.

[6] Das, S., and Suganthan, P. N. Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on

real world optimization problems. Tech. rep., Jadavpur University, India and Nanyang Technological University, Singapore, 2010.

[7] Fujisawa, K., Shinano, Y., and Waki, H., Eds. *Optimization in the Real World.* Springer Japan, 2016.

[8] Gouda, E. A., Kotb, M. F., and El-Fergany, A. A. Jellyfish search algorithm for extracting unknown parameters of PEM fuel cell models: Steady-state performance and analysis. *Energy 221* (2021), 119836.

[9] Karaboga, D., and Akay, B. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation 214*, 1 (2009), 108–132.

[10] Kiran, M. S. Tsa: Tree-seed algorithm for continuous optimization. *Expert Systems with Applications 42*, 19 (2015), 6686–6698.

[11] Mirjalili, S., Mirjalili, S. M., and Lewis, A. Grey wolf optimizer. *Advances in Engineering Software 69* (2014), 46–61.

[12] Poláková, R., and Bujok, P. Popular optimisation algorithms with diversity-based adaptive mechanism for population size. *Recent Advances in Soft Computing and Cybernetics: Studies in Fuzziness and Soft Computing 403* (2021), 171 – 182.

[13] Selvakumar, S., and Manivannan, S. S. A spectrum defragmentation algorithm using jellyfish optimization technique in elastic optical network (EON). *Wireless Pers Commun* (2021).

[14] Shi, Y., and Eberhart, R. A modified particle swarm optimizer. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence* (1998), pp. 69–73.

[15] Surantha, N., Lesmana, T., and Isa, S. M. Sleep stage classification using extreme learning machine and particle swarm optimization for healthcare big data. *J Big Data 8*, 14 (2021).

[16] Wang, Y., Li, H.-X., Huang, T., and Li, L. Differential evolution based on covariance matrix learning and bimodal distribution parameter setting. *Applied Soft Computing 18* (2014), 232–247.

[17] Yang, X.-S. A new metaheuristic bat-inspired algorithm. In *Nicso 2010: Nature Inspired Cooperative Strategies for Optimization* (2010), J. Gonzalez, D. Pelta, C. Cruz, G. Terrazas, and N. Krasnogor, Eds., vol. 284 of *Studies in Computational Intelligence*, Univ Laguna; Carnary Govt; Spanish Govt, pp. 65–74. International Workshop on Nature Inspired Cooperative Strategies for Optimization NICSO 2008, Tenerife, Spain, 2008.

[18] Yang, X.-S. *Nature-Inspired Optimization Algorithms.* Elsevier, 2014.

[19] Yang, X.-S., and Deb, S. Cuckoo search via Lévy flights. In *2009 World Congress on Nature Biologically Inspired Computing NaBIC* (2009), pp. 210–214.

[20] Zelinka, I., and Lampinen, J. SOMA – self organizing migrating algorithm. In *MENDEL, 6th International Conference On Soft Computing, Brno, Czech Republic* (2000), R. Matousek, Ed., pp. 177–187.

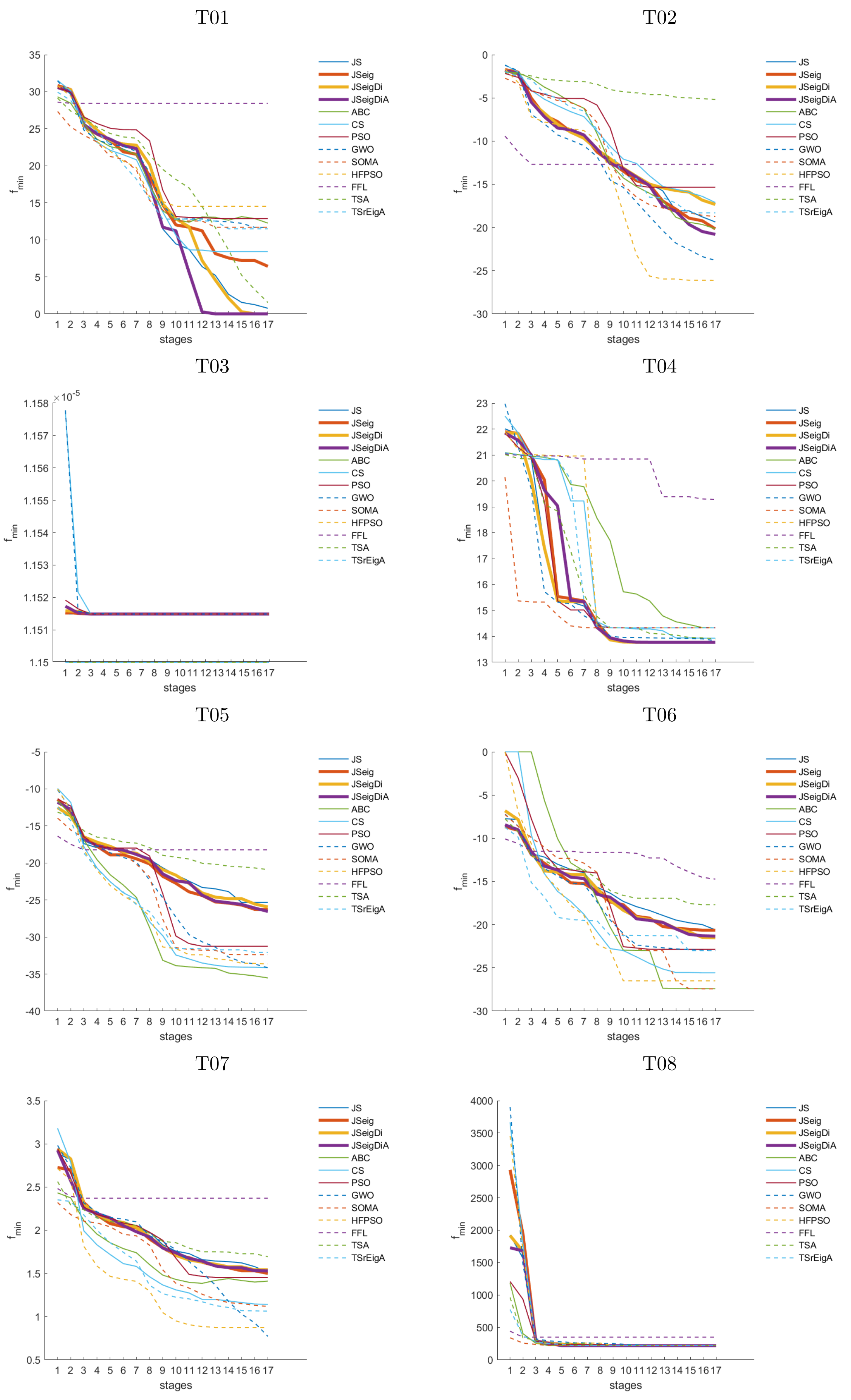

Figure 2: Convergence ability of 13 nature-inspired optimisation methods (T01 - T08).

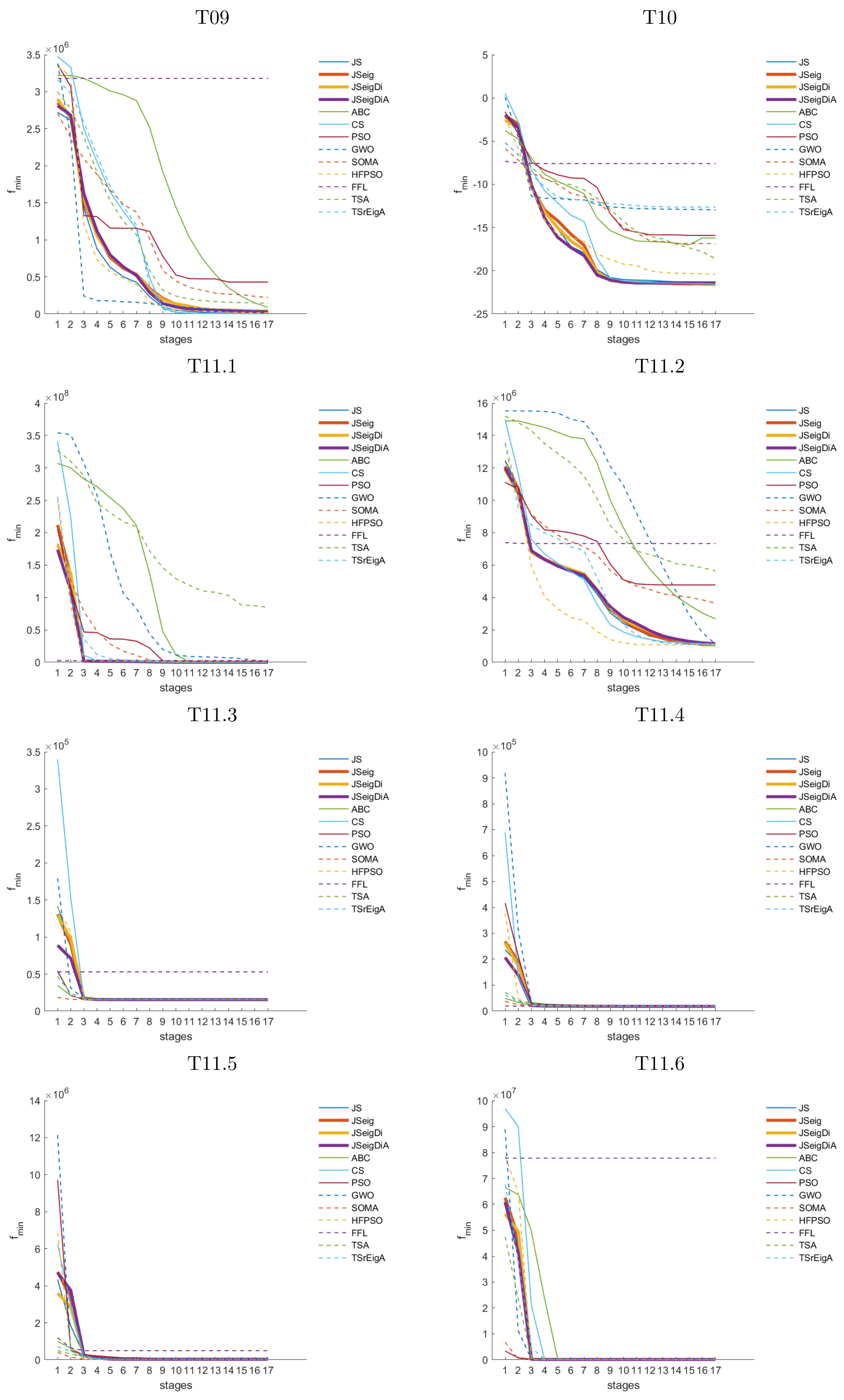

Figure 3: Convergence ability of 13 nature-inspired optimisation methods (T09 - T11.6).

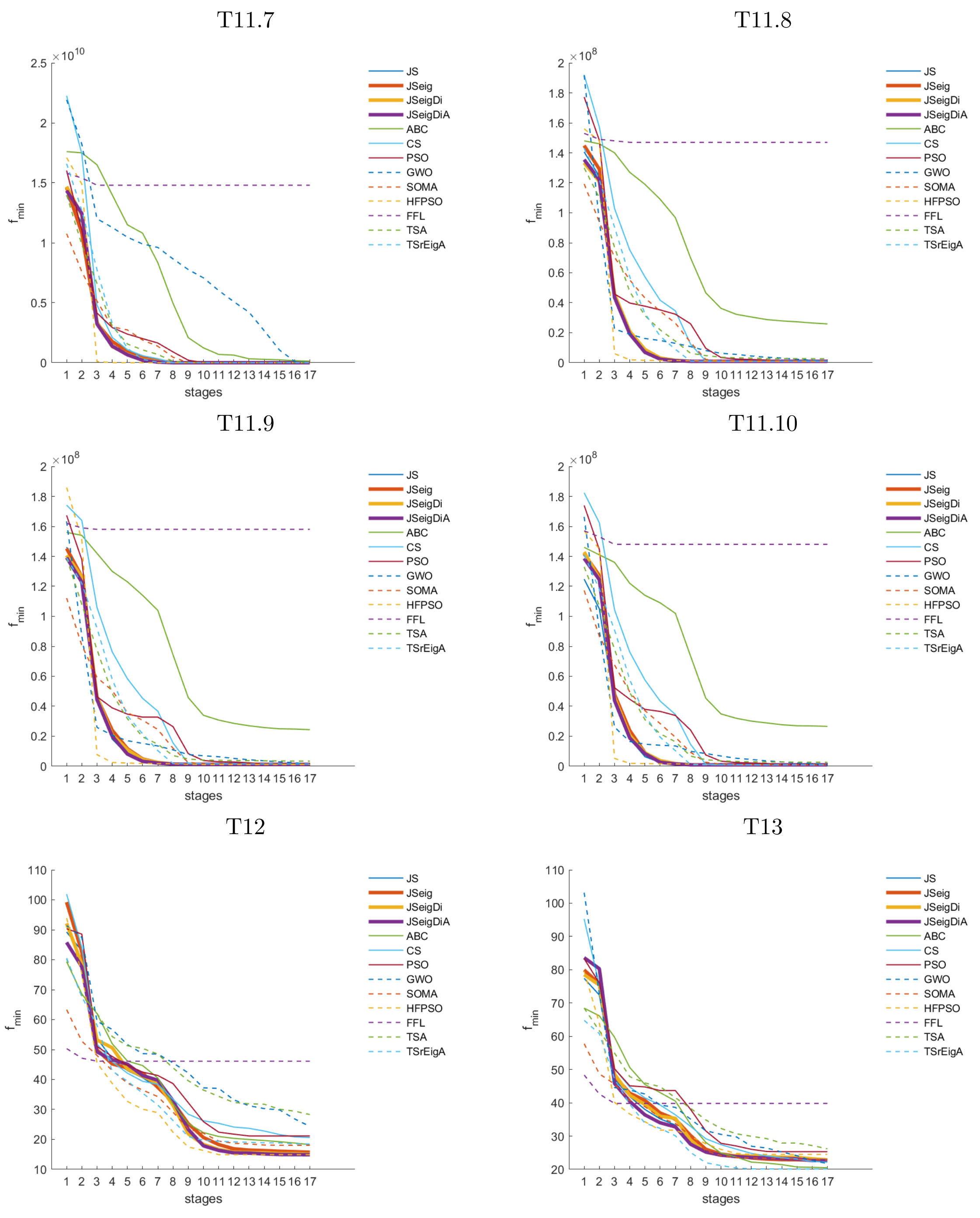

Figure 4: Convergence ability of 13 nature-inspired optimisation methods (T11.7 - T13).