Efficient truss optimization using the contrast-based fruit fly optimization algorithm

Kanarachos, S., Griffin, J., and Fitzpatrick, M. E.

Original citation:
Kanarachos, S., Griffin, J., and Fitzpatrick, M. E. (2017) Efficient truss optimization using the contrast-based fruit fly optimization algorithm. *Computers and Structures*, Vol. 182, April 2017, pp. 137-148

https://doi.org/10.1016/j.compstruc.2016.11.005

DOI 10.1016/j.compstruc.2016.11.005
ISSN 0045-7949

Publisher: Elsevier

Title:

Efficient multi-parameter optimisation using Contrast based Fruit Fly Optimisation

Abstract:

A recent biological study showed that the extremely good efficiency of fruit flies in finding food, despite their small brain, emerges by two distinct stimuli, smell and visual contrast. “Contrast based Fruit Fly Optimisation”, presented in this paper, is for the first time mimicking this fruit fly behaviour and further developed to address more efficiently multi-parameter optimisation problems. To assess its performance a study was carried out on ten mathematical and three truss optimisation problems. The results are compared to those obtained using twelve state-of-the-art optimisation algorithms and confirm its good and robust performance. A sensitivity analysis and an evaluation of its performance under parallel processing were conducted. The proposed algorithm has only a few tuning parameters, is intuitive, and multi-faceted allowing application to complex n-dimensional design optimisation problems.

Authors:

Stratis Kanarachos[a], James Griffin[a],and Michael E. Fitzpatrick[a]

[a] *stratis.kanarachos@coventry.ac.uk, ac0393@coventry.ac.uk, ab6856@coventry.ac.uk, Faculty of Engineering, Environment and Computing, Coventry University, Priory Street, Coventry CV1 5FB, United Kingdom*

Corresponding author: *Stratis Kanarachos, stratis.kanarachos@coventry.ac.uk Tel: +44(0)2477657720, Engineering & Computing Building - EC 4-07, Faculty of Engineering, Environment and Computing, Coventry University, 3, Gulson Road, Coventry, CV1 2JH*

Keywords:

Fruit fly optimisation; multi-parameter; truss optimisation

1. Introduction

Design optimisation is a powerful tool widely utilised by engineers to produce better performing, more reliable and cost-effective products. It originated from the aircraft industry and rapidly expanded in multiple domains like structural and mechatronics engineering [1, 2, 3]. Its success is mainly due to its inherent merit, delivered in combination with a significant increase in computational power and accessibility to practitioners through commercial engineering software [4]. The mathematical formulation of an optimisation problem can be expressed as:

$$minimize\ f(\mathbf{x}),\ \mathbf{x}=[x_1, x_2, \ldots, x_m]^T \qquad (1)$$

subject to: $x_{imin} \leq x_i \leq x_{imax}$, $i=1,2,\ldots,m$

where f(**x**) is the objective function that expresses the performance of a system, **x** is a vector comprised out of design variables x_i, m is the total number of design variables, and x_{imin}, x_{imax} are the lower and upper bound of design variable x_i respectively.

Initially, optimisation technology was based on mathematical formulations involving the calculation of derivatives [5]. For example, in the gradient descent method, one starts from an initial point $\mathbf{x_0}$ where the function value f($\mathbf{x_0}$) is calculated and then takes a step in a downward direction, where the function value will be lower. To make such a step, one utilizes local information $\nabla f^T(\mathbf{x_0})$ and explores the immediate vicinity of the current point. The search for the optimum design vector $\boldsymbol{x}^*$ is expressed by the following iterative formula:

$$\mathbf{x_{k+1}} = \mathbf{x_k} - a_k \cdot \nabla f^T(\mathbf{x_k}) \tag{2}$$

where a_k is a scaling parameter, k is the iteration number, $\mathbf{x_k}$ is the design vector in kth iteration and $\mathbf{x_{k+1}}$ is the new design vector.

Although mathematically rigorous, the gradient-based algorithms get trapped in local minima in case of noisy or highly nonlinear problems. Contrary, meta-heuristic optimisation algorithms like the *Genetic Algorithm* [6], *Particle Swarm Optimisation* [7] and *Harmony Search* [8] do not use gradient information and achieve remarkably better results. On the downside, the performance of non-gradient algorithms depends on a number of tuning parameters which are not known prior to execution. Although, in some cases, empirical rules exist they are not always adequate. There is a need for intuitive meta-heuristic algorithms with a minimum number of tuning parameters.

2. Brief literature review on truss optimisation

Trusses are fundamental in structural engineering and applications can be found from nano to macro levels [9, 10]. Truss optimisation problems are usually multi-parameter optimisation problems due to the large number of members comprising the truss. They are also highly nonlinear because of the multiple constraints considered, including displacement, stress and frequency, and the complex interaction between the structural members. In the general case, the truss optimisation problem is formulated as a mathematical optimisation problem:

find $\mathbf{x}=[x_1, x_2, \ldots, x_m]$

that minimises $R(\mathbf{x}) = \sum_{i=1}^{n} \rho_i \cdot A_i \cdot L_i$

subject to: $\delta_{imin} \leq \delta_i \leq \delta_{imax}$, $i = 1,2,\ldots,n_n$
$\sigma_{imin} \leq \sigma_i \leq \sigma_{imax}$, $i = 1,2,\ldots,n_n$
$A_{imin} \leq A_i \leq A_{imax}$, $i = 1,2,\ldots,n$
$\omega_{imin} \leq \omega_i \leq \omega_{imax}$, $i = 1,2,\ldots, n_\omega$

(3)

where R is the mass of the truss, m is the number of design parameters, n is the number of truss members, n_n is the number of nodes, n is the number of truss members and n_ω the number of desired natural frequencies. ρ_i, A_i and L_i are the density, cross sectional area and length of ith member respectively. δ_i and σ_i are the displacement and stress at the ith node. ω_i is the ith natural frequency. δ_{imin} and δ_{imax} are the lower and upper displacement bounds for the ith node, σ_{imin} and σ_{imax} are the lower and upper normal stress bounds for the ith node, A_{imin} and A_{imax} are the lower and upper cross sectional area bounds for the ith structural member and ω_{imin} and ω_{imax} are the lower and upper bounds for the ith natural frequency.

There is an increasing interest in developing efficient algorithms for large scale truss optimisation. The algorithms are mainly meta-heuristic and broadly classified into three categories.

The first category encompasses the Evolutionary Algorithms (EA). EAs use mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection for calculating new candidates $\mathbf{x}_{k+1}^{i}$, i=1,…, M, where M is the population size. EAs usually suffer from premature convergence and weak exploitation capabilities. Both drawbacks are compensated by choosing bigger populations, however, this leads to larger computational cost. Wei et al [11] proposed, as a solution to this problem, the Niche Hybrid Parallel Genetic Algorithm (NHPGA). NHPGA aims to effectively combine the robust and global search characteristics of the genetic algorithm, strong exploitation ability of Nelder–Mead's simplex method and computational speedup property of parallel computing.

The second category includes population-based algorithms, such as *Particle Swarm Optimisation* (PSO) [12]. PSO is formulated by mathematically modelling the social behaviour of birds and fish colonies in finding food resources or escaping from predators. In the standard PSO each member of the swarm finds its way based on their own experience and the best particle's position, particles do not exchange any information. This causes PSO to get trapped into local optimums. In a recent publication by Mortazavi and Toğan [13] a new version of PSO was proposed. In this version, the concept of a weighted particle, created by exploiting all particles experiences, is introduced. This helps to avoid premature convergence.

In the third category belong physical algorithms that resemble an employed physical process. For example, Kaveh and Bakhshpoori [14] developed an algorithm that mimics the evaporation of a tiny amount of water molecules on a solid surface with different wettability. The "*Water Evaporation Optimisation Algorithm*" was tested and analysed in comparison to other existing methods on a set of 17 benchmark unconstrained functions, a set of 13 classical benchmark constraint functions, and, three benchmark constraint engineering problems. The results obtained indicate that the proposed technique is highly competitive. The performance of the algorithm depends on a number of parameters including the assumption of a monolayer and droplet evaporation phase, the number of water molecules and the minimum and maximum values of monolayer and droplet evaporation probabilities. Another example is the modified *Teaching–Learning-based optimization* (*TLBO*) algorithm [15]. *TLBO* mimics the two types of pedagogy in a classroom to find the optimum solution: class-level learning from a teacher and individual learning between students.

TLBO uses a relatively simple algorithm with no intrinsic parameters controlling its performance.

Fruit flies are very effective in finding food. They can locate a food source from 40 km away even though their brain is very simple: it has only 100,000 neurons while house fly brains have 300,000 neurons and human brains have 100 billion. This remarkable ability makes them very interesting from a biological and optimisation perspective [16, 17]. The main food search mechanism is based on smell. However, a recent biological study shows that fruit flies are stimulated also by visual contrasts, irrelevant to smell. Furthermore, their motion is described by standardised distinct sensory-motor reflexes, independent of each other. The Contrast-based *Fruit Fly Optimisation Algorithm*, proposed in this paper, mimics these new elements of fruit fly behaviour. First, it is evaluated on a set of standard mathematical benchmark tests and then applied to structural truss design benchmark problems. It is highlighted that fruit fly algorithms have never been tested in structural optimisation before. The results show that the algorithm achieves the same or better performance than other optimisation algorithms.

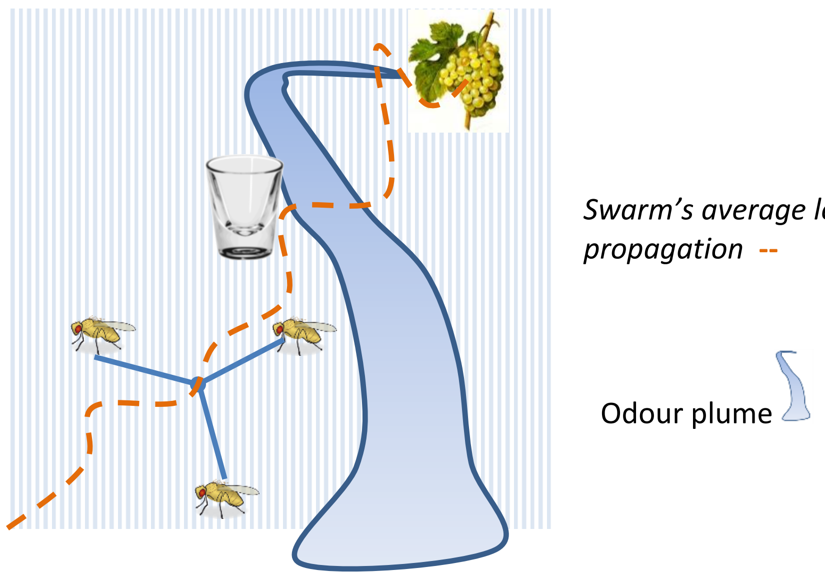

Figure 1. Fruit fly swarm in search for food: Visual patterns and features stimulate fruit flies in their search besides smell

The rest of the paper is structured as follows: in Section 3 the *Contrast based Fruit Optimization Algorithm* is presented and explained. In Section 4 the results for ten mathematical and three structural optimisation problems are discussed and compared to those known from the literature and obtained using standard optimisation tools. In Section 5 a sensitivity analysis is performed, including a performance evaluation when parallel computing is used. Finally, Section 6 gives conclusions and future work is proposed.

3. The Contrast-Based Fruit Fly Optimisation Algorithm

3.1 Contrast based Fruit Fly Optimization Algorithm for multi-parameter problems (*c-mFOA*)

Fruit flies have a keen sense of smell and use their antennae to detect odours. A fly can detect a source of food and where its fellows gather and fly to that direction. This behaviour was modelled in the first fruit fly optimisation algorithm proposed by Pan [18]. Its simplicity and efficiency made it popular and different versions were proposed to improve its performance [19-22]. According to a recent biological study, fruit flies exhibit also an additional food search mechanism [23, 24]. In case they can’t find food using only osphresis, they start to explore objects with visual contrast. They land and if there is not something to eat, they continue to forage. As an example, a glass of wine would be a contrasting shape that would merit their attention. Furthermore, it was found that fruit flies surge when the scent is strong and cast when it becomes weaker. Last but not least, fruit flies present also a response delay. It is believed that fruit flies developed these features to compensate for the chaotic movement of odours, particularly outdoors in the wind. For the first time, in this paper, the particular fruit fly behaviour is idealised, modelled and further developed to address multi-parameter optimisation problems.

The basic steps of *c-mFOA* are summarised by the pseudo code shown in Figure 2. A flowchart is provided in Figure 3.

```
Multi- parameter Contrast Based Fruit Optimisation Algorithm
begin
    Objective function $f(\mathbf{x})$, $\mathbf{x}=[x_1, x_2, \ldots, x_m]$
    Generate initial population of fruit flies $\mathbf{x}_i$, $i=1,2,\ldots, N$ in the vicinity of $\mathbf{x}_0$
    Smell concentration $S_{mi}$ at $\mathbf{x}_i$ determined by $S_{mi} = f(\mathbf{x}_i)$
    Rank the fruit flies and find the current best $\min(S_{mi}) = S_{mi}^*$ for $\mathbf{x}_i^*$
    If $S_{mi}^* < S_{m0}$ then $\mathbf{x}_0 = \mathbf{x}_i^*$
    while ($t$ < MaxGeneration)
        Reposition the fruit flies $\mathbf{x}_{ki}$, $k=1,2, \ldots, K$ and $i=1,2,\ldots, N$ in the vicinity
        of current $\mathbf{x}_{k0}$
        Smell concentration $S_{mki}$ at $\mathbf{x}_{ki}$ determined by $S_{mki} = f(\mathbf{x}_{ki})$
        Rank the fruit flies and find the current best $S_{mki}^*$ for $\mathbf{x}_{ki}^*$
        If $S_{mki}^* < S_{mk0}$ then $\mathbf{x}\mathbf{k}_0 = \mathbf{x}_{ki}^*$
        if ($t_k$>delay time)
            if ($S_{mki} < S_{m(k-\kappa)0}$)
                reduce the area of attraction (surging phase)
            else if ($S_{mki} = S_{m(k-\kappa)0}$)
                set the attraction point at the worst performing
                candidate, $\mathbf{x}_{k0} = \mathbf{x}_{ki}^{\therefore}$ for which $\max(S_{mki}) = S_{mki}^{\therefore}$
            else if ($S_{mki} > S_{m(k-\kappa)0}$)
                return to the previous the current best, $\mathbf{x}_{k0} = \mathbf{x}_{(k-\kappa)0}$
                (casting phase)
            end if
        end if
        Initialise response time $t_k = 0$
    end while
    Post process results and visualisation
end
```

Figure 2. Pseudo-code of the proposed Fruit Fly Optimisation Algorithm

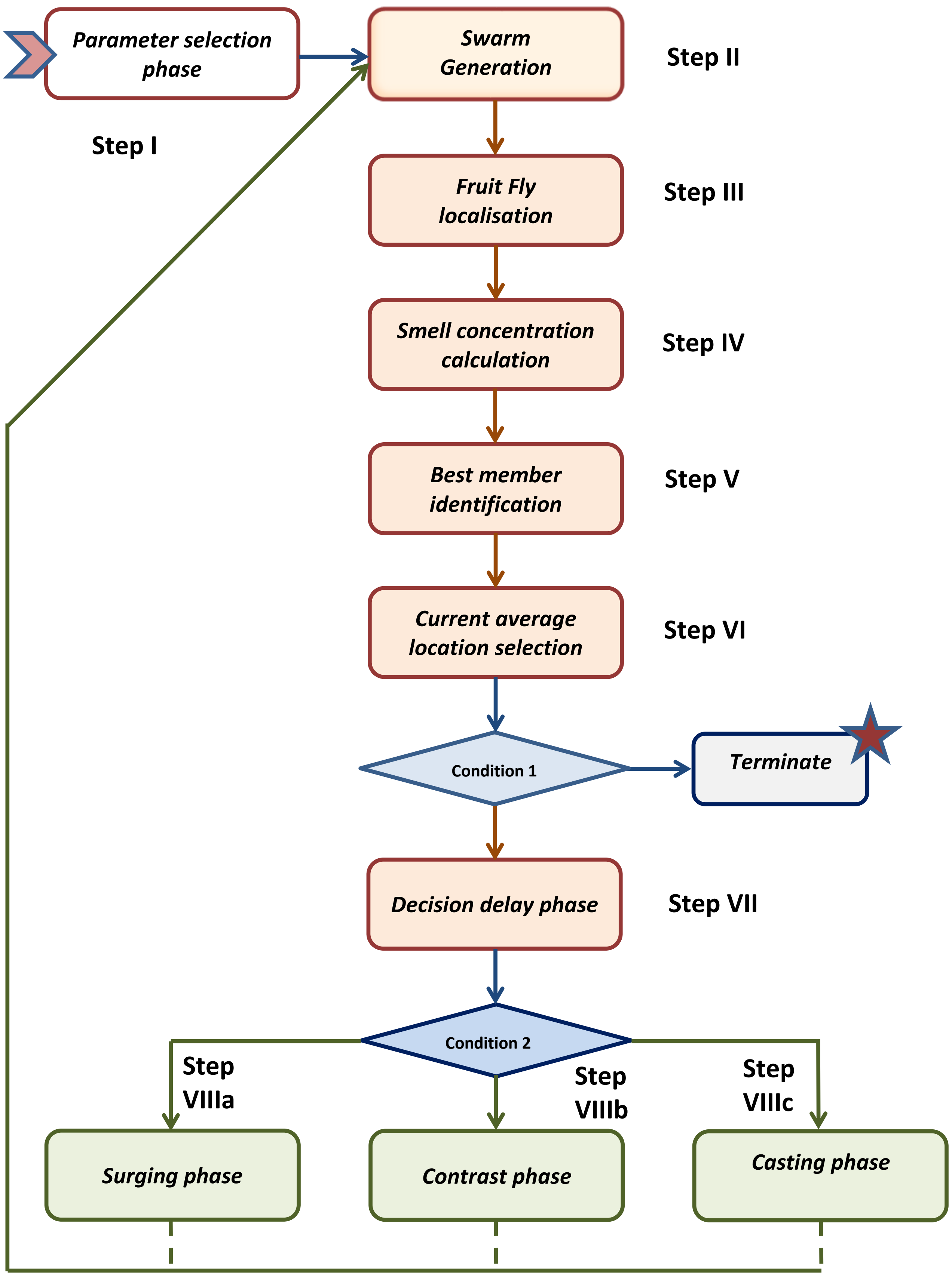

Figure 3. Flowchart of proposed *c-mFOA* algorithm

3.2 Swarm localisation and termination

A coordinate system is defined and the position of a fruit fly with coordinates (X_0,Y_0) is defined, see Figure 4. The other N-1 fruit flies are located, randomly, in the vicinity of (X_0,Y_0) according to Equation 4.

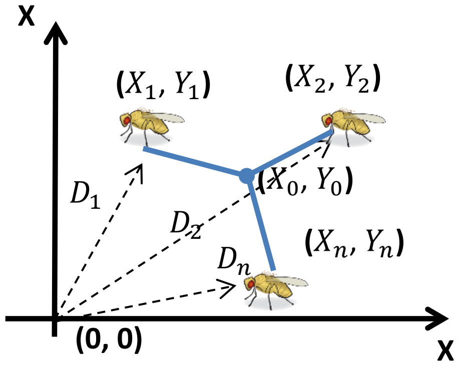

Figure 4. Fruit fly position (X_i, Y_i) is described in a coordinate system

$$X_{ki} = X_{k0}\left(1 + M \cdot (2 \cdot rand_{N_{res}} - 1\right), i=1,\ldots,N \quad (4)$$
$$Y_{ki} = Y_{k0}\left(1 + M \cdot (2 \cdot rand_{N_{res}} - 1\right), i=1,\ldots,N$$

where k=1,2,…,K is the iteration number, N is the size of the swarm and $rand_{N_{res}}$ is a random number from a uniform discrete distribution defined in the interval [1, N_{res}] . The use of a discrete distribution is not observed in nature, but is a feature we introduced to improve the algorithm's performance in multi parameter problems. M is a scaling parameter that defines how coarse or fine the search strategy is.

Each fruit fly is assigned a value DI_i based on how close the fruit fly (X_{ki}, Y_{ki}) is to the origin of a fixed coordinate system:

$$D_{ki} = \sqrt{X_{ki}^2 + Y_{ki}^2} \quad (5)$$

$$DI_{ki} = \frac{1}{D_{ki}} \quad (6)$$

DI_{ki} is sensitive for fruit flies located in the vicinity of the origin, contrary to those that are positioned far away. This implies that a good search strategy should start close to the origin.

Each fruit fly is assigned a "smell concentration" S_{mki} at $\mathbf{x}_{ki}$ determined by the objective function value $S_{mki} = f(\mathbf{x}_{ki})$. A small objective function value corresponds to a position with high smell concentration.

The fruit flies are ranked, on the basis of their smell concentration, and the fruit fly $\boldsymbol{x}^*_{ki}$ that achieves the highest smell concentration S^*_{mki} at position (X^*_{ki}, Y^*_{ki}) is identified. In case the smell concentration S^*_{mki} is better than that of the current point of attraction S_{k0}:

$$\begin{array}{l} if\ S^*_{mki} < S_{mk0} \\ then\ X_{k0} = X^*_{ki}\ and\ Y_{k0} = Y^*_{ki} \end{array} \tag{7}$$

then it substitutes it and becomes the new point of attraction.

The algorithm terminates when the maximum number *K* iterations is reached.

3.3 Delay, casting, surging and visual contrast phases

When the stimulus changes fruit flies don't respond immediately; a delay is taking place before changing the food search strategy. As presented in [24], the delay is constant and independent of other parameters. This behaviour is idealised and modelled in *c-mFOA* algorithm.

In case the objective function improves over the last κ iterations, the swarm enters the "surging" phase, during which the flies move towards the attraction point $\mathbf{x}_{k0}$ at a greater speed:

$$\begin{array}{l} if\ (S_{mki} < S_{m(k-\kappa)0}) \\ M_{k+1} = c \cdot M_k \end{array} \tag{8}$$

In case the objective function doesn't change over the last κ iterations, κ represents the response delay, the swarm enters the "visual contrast" attraction phase, in which flies are attracted by the point $\mathbf{x}^{\therefore}_{ki}$ that achieves the lowest smell concentration $\max(S_{mki}) = S^{\therefore}_{mki}$:

$$\begin{array}{l} if\ (S_{mki} = S_{m(k-\kappa)0}) \\ X_{k0} = X^{\therefore}_{ki}\ and\ Y_{k0} = Y^{\therefore}_{ki} \end{array} \tag{9}$$

where *k* is the current iteration.

In case the objective function worsens over the last κ iterations, the swarm enters the "casting" phase, in which flies return to the previous current best $\mathbf{x}_{(k-\kappa)0}$ and continue the search at a constant speed:

$$\begin{array}{l} if\ (S_{mki} > S_{m(k-\kappa)0}) \\ X_{k0} = X_{(k-\kappa)0}\ and\ Y_{k0} = Y_{(k-\kappa)0} \end{array} \tag{10}$$

It is known from [25] that fruit flies do have memory and they can make a choice based on how good or bad a memory was.

3.4 Constraint handling

In *c-mFOA* the constraints are dealt using the penalty function. The constrained optimisation problem is formulated as an unconstrained one by augmenting the response function R(**x**) as shown in Equation (11):

$$\begin{aligned} minimize\ f(\mathbf{x}) = R(\mathbf{x}) + \sum_{i=1}^{n_n} \kappa_i \cdot \varphi_i^2(\mathbf{x}) + \sum_{i=1}^{n_n} \lambda_i \cdot \psi_i^2(\mathbf{x}) \\ + \sum_{i=1}^{n} \mu_i \cdot \chi_i^2(\mathbf{x}) + \sum_{i=1}^{n_n} \nu_i \cdot \pi_i^2(\mathbf{x}) \end{aligned} \qquad (11)$$

$$\kappa_i > 0, \lambda_i > 0, \mu_i > 0 \text{ and } \nu_i > 0$$

where κ_i, λ_i, μ_i, and ν_i are user defined constants and

$$\varphi_i(\mathbf{x}) = \begin{cases} (\delta_i - \delta_{imax})^2, if\ \delta_i > \delta_{imax} \\ (\delta_i - \delta_{imin})^2, if\ \delta_i < \delta_{imin} \\ 0,\ if\ \delta_{imin} \le \delta_i \le \delta_{imax} \end{cases} \qquad (12)$$

$$\psi_i(\mathbf{x}) = \begin{cases} (\sigma_i - \sigma_{imax})^2, if\ \sigma_i > \sigma_{imax} \\ (\sigma_i - \sigma_{imin})^2, if\ \sigma_i < \sigma_{imin} \\ 0,\ if\ \sigma_{imin} \le \sigma_i \le \sigma_{imax} \end{cases} \qquad (13)$$

$$\chi_i(\mathbf{x}) = \begin{cases} (A_i - A_{imax})^2, if\ A_i > A_{imax} \\ (A_i - A_{imin})^2, if\ A_i < A_{imin} \\ 0,\ if\ A_{imin} \le A_i \le A_{imax} \end{cases} \qquad (14)$$

$$\pi_i(\mathbf{x}) = \begin{cases} (\omega_i - \omega_{imax})^2, if\ \omega_i > \omega_{imax} \\ (\omega_i - \omega_{imin})^2, if\ \omega_i < \omega_{imin} \\ 0,\ if\ \omega_{imin} \le \omega_i \le \omega_{imax} \end{cases} \qquad (15)$$

4. Benchmark testing

4.1 Results on multi-parameter mathematical benchmark problems

A set of multi-parameter mathematical functions, commonly used in the literature, is employed to benchmark *c-mFOA*. In Table 1 the mathematical description, the number of variables used, the design space, the optimal position and function value for each function are provided. The functions are characterized by multiple local minima, singular values, and hyper planes ranging from flat to very steep. Solving the benchmark functions is a strong indicator of the robustness and effectiveness of the developed *c-mFOA*. The benchmark is conducted for *m*=30 variables.

Table 1. Mathematical benchmark functions

No	Description	m	$[x_{imin}, x_{imax}]$	x*	f(x*)
F1	$f(x) = \sum_{i=2}^{m} i \cdot x_i^2$	30	$[-5.12, 5.12]$	0	0
F2	$f(x) = \sum_{i=2}^{m} i \cdot (2 \cdot x_i^2 - x_{i-1}^2)^2 + (x_1 - 1)^2$	30	$[-10, 10]$	0	0
F3	$f(x) = -\exp(-0.5 \cdot \sum_{i=1}^{m} x_i^2)$	30	$[-1, 1]$	0	-1
F4	$f(x) = \sum_{i=2}^{m} (10^6)^{\frac{i-1}{n-1}} \cdot x_i^2$	30	$[-100, 100]$	0	0
F5	$f(x) = \max(\lvert x_i \rvert)$	30	$[-100, 100]$	0	0
F6	$f(x) = \sum_{i=2}^{m} floor(x_i + 0.5)^2$	30	$[-100, 100]$	0	0
F7	$f(x) = \sum_{i=2}^{m} x_i^2$	30	$[-100, 100]$	0	0
F8	$f(x) = \sum_{i=1}^{m} \lvert x_i \cdot sin(x_i) + 0.1 \cdot x_i \rvert$	30	$[-10, 10]$	0	0
F9	$f(x) = \sum_{i=1}^{m} (x_i^2 - 10 \cdot cos(2 \cdot \pi \cdot x_i) + 10)$	30	$[-5.12, 5.12]$	0	0
F10	$f(x) = 1 - cos\left(2 \cdot \pi \sum_{i=1}^{m} x_i^2\right) + 0.1 \cdot \sum_{i=1}^{m} x_i^2$	30	$[-100, 100]$	0	0

c-mFOA is benchmarked against the *Genetic* (*GA*), *Particle Swarm Optimisation* (*PSO*) and *Simulated Annealing* (*SA*) algorithms, commonly used in engineering practice. It is highlighted that many variants of the aforementioned algorithms exist and it is by no means attempted to compare *c-mFOA* to all variants. The purpose of this exercise is to show the differences between well-known and widely employed algorithms (one evolutionary, one population-based and one physics-based), when implemented with their default parameter settings. The variants of *GA*, *PSO* and *SA* used in this study are the ones found in Matlab's Optimisation Toolbox.

In the *GA* a population comprised of 200 members is utilised. The members are selected randomly from a uniform distribution restricted in the design space (*DS*), see Table 1. For each member the fitness value is calculated. The *GA* members are then sorted according to their rank. 80% of the new generation is created by crossover and 5% progresses from the old generation. A stochastic uniform algorithm is used for the parent selection. The rest of the members are created by mutation. The genetic algorithm terminated when the maximum number of function evaluations generations is reached, unless it stalled. This was set to happen if for over 200 generations the objective function did not change significantly.

The inspiration for the *PSO* algorithm is based on flocks of birds swarming. The first step involves the generation of a population of particles with assigned initial velocities. The particles are uniform randomly created within bounds shown in Table 1. A fitness value is calculated for each particle and then the location that achieves the best value is determined. The algorithm chooses new velocities, based on the current velocity, the particles' individual best locations, and the best locations of their neighbours. It then iteratively updates the particle locations based on their old location and velocity and, its' neighbours. The inertia range parameter of the algorithm was set within its standard bound [0.1 1.1]. The self-adjustment and social adjustment weights were set to their standard value 1.49. The swarm size was set to 100. Iterations proceeded until the algorithm reached the maximum number of function evaluations.

The simulated annealing (*SA*) algorithm starts from a random starting vector belonging to DS (Table 1). Two parameters; the temperature and re-annealing determine its behaviour. The first one controls the extent of search. In this study the default initial temperature of 100 was used. The second one emulates the annealing process. After a certain number of new points are accepted, the temperature is raised to a higher value to restart the search and move out of local minima. If re-annealing is performed too fast this may not help the solver identify a minimum. Here, the default interval of 50 was chosen. The procedure terminates when the total number of function evaluations reaches the maximum value.

In *c-mFOA* the starting vector is a random vector bounded by D*S* (Table 1). The other parameters are selected as K=320, N=50, κ=15, M=0.95 and c=0.92.

The benchmark is accomplished on the basis of a maximum number of function evaluations *maxfun*=16000. The parameters used for the evaluation are the mean value of optimised values found (Mean) and their standard deviation (Std), following 30 independent optimisation runs. The results are summarised in Table 2. In the last column the optimal value for each function is provided.

Table 2. Optimisation benchmark results: Mean best value (Mean) and standard deviation (Std) obtained using the *Genetic Algorithm* (*GA*), *Particle Swarm Optimisation* (*PSO*), *Simulated Annealing* (*SA*) and *contrast based multi-parameter*

Fruit Fly Optimisation (*c-mFOA*). Maximum number of function evaluations *maxfun*=16000

Function	***GA***		***PSO***		***SA***		***c-mFOA***	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
F1	7.35	2.67	$2.10\cdot10^{-3}$	$3.00\cdot10^{-3}$	2.31	0.91	0	0
F2	1.82	1.32	$5.62\cdot10^{-4}$	$7.45\cdot10^{-4}$	0.49	0.16	$2.55\cdot10^{-4}$	$2.76\cdot10^{-4}$
F3	–0.95	0.01	–1	$1.55\cdot10^{-6}$	–0.31	0.07	–1	0
F4	$2.43\cdot10^{4}$	$2.23\cdot10^{4}$	$2.73\cdot10^{5}$	$1.30\cdot10^{6}$	$6.00\cdot10^{6}$	$1.38\cdot10^{6}$	$2.55\cdot10^{-14}$	$1.16\cdot10^{-15}$
F5	1.90	0.67	14.05	4.21	68.35	4.67	$5.63\cdot10^{-5}$	$6.47\cdot10^{-5}$
F6	12.03	8.38	0.10	0.30	2.86	1.61	0	0
F7	0.72	0.38	0.040	0.034	6.22	1.51	$4.32\cdot10^{-19}$	$8.08\cdot10^{-20}$
F8	0.67	0.41	0.03	0.06	25.42	1.32	$2.83\cdot10^{-10}$	$1.18\cdot10^{-11}$
F9	26.74	10.67	60.23	21.91	154.68	31.11	14.31	3.38
F10	$2.00\cdot10^{-14}$	$3.80\cdot10^{-14}$	$7.26\cdot10^{-7}$	$1.51\cdot10^{-6}$	$1.57\cdot10^{-9}$	$2.95\cdot10^{-9}$	$2.95\cdot10^{-8}$	$5.25\cdot10^{-8}$

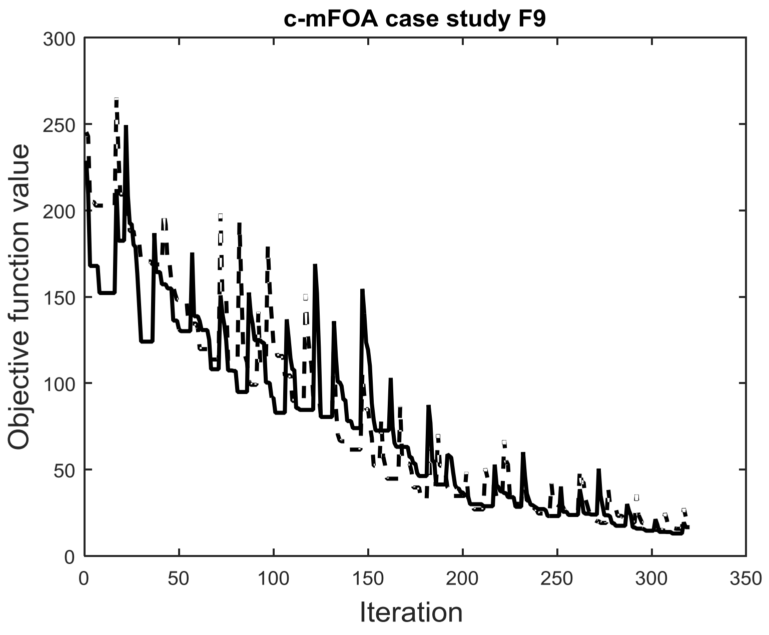

Figure 5. F9 case study: Convergence rate examples for *c-mFOA*

A sensitivity analysis, for function F9 $f(x) = \sum_{i=1}^{m}(x_i^2 - 10 \cdot cos(2 \cdot \pi \cdot x_i) + 10)$, was conducted by varying the tuning parameter *M* in *c-MFOA*. The results obtained are listed in Table 3. For *M*>5 we achieve the same results as with *M*=5 (true optimum).

Table 3. *c-mFOA* sensitivity analysis for different *M* parameter values in solving F9

Fun-ction	***c-mFOA***							
	M=2		***M=3***		***M=4***		***M=5***	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
F9	16.20	6.28	0.56	0.78	0.53	0.50	0	0

4.2 Results on structural optimisation benchmark problems

In this section, *c-mFOA* is compared to state-of-the-art optimisation algorithms in solving truss optimisation problems. Each problem is solved repetitively and independently for thirty times.

4.2.1 Case 1- Twenty-five bar space truss

In Figure 6 the 25-bar transmission bar used for benchmarking is illustrated. It is one of the most common benchmarks for truss optimisation algorithms. The coordinates of the nodes of the 25-bar truss are listed in Table, while Table 5 shows the element connectivity and Table 6 the load case considered. The material considered has a density of ρ=2768 kg/m^3 (0.1 lb/in.^{3}) and a modulus of elasticity of E=0.73·10^6 Pa (107 psi).

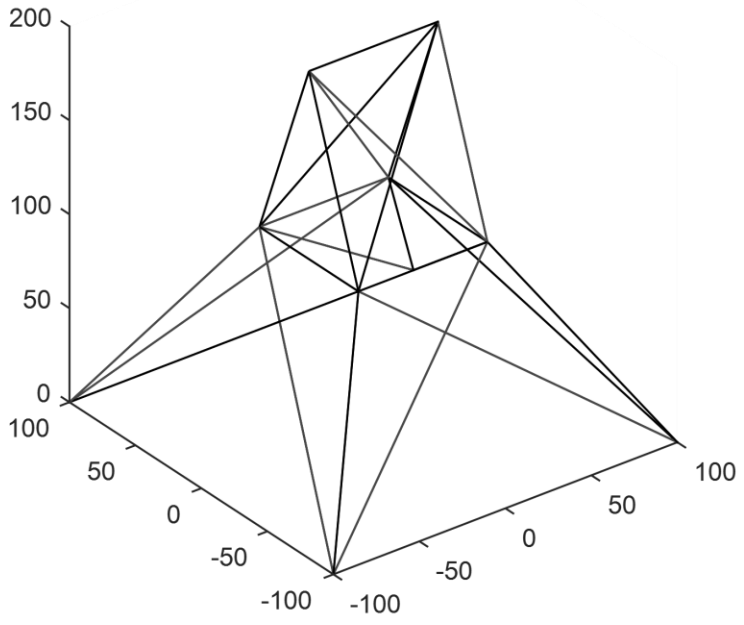

Figure 6. Cases 1 & 2: 25 bar truss

Table 4. Coordinates of the points defining the 25-bar truss

Node	**x** [m]	**y** [m]	**z** [m]
1	−0.95	0.00	5.08
2	0.95	0.00	5.08
3	-0.95	0.95	2.54
4	0.95	0.95	2.54
5	0.95	−0.95	2.54
6	−0.95	−0.95	2.54
7	−2.54	2.54	0.00
8	2.54	2.54	0.00
9	2.54	−2.54	0.00
10	−2.54	−2.54	0.00

Table 5. Element (El.) connectivity for the 25-bar truss

El.	**Node 1**	**Node 2**	**El.**	**Node 1**	**Node 2**	**El.**	**Node 1**	**Node 2**
1	1	2	**10**	6	3	**19**	3	8
2	1	4	**11**	5	4	**20**	5	10
3	2	3	**12**	3	4	**21**	6	9
4	1	5	**13**	6	5	**22**	6	10
5	2	6	**14**	3	10	**23**	3	7
6	2	4	**15**	6	7	**24**	4	8
7	2	5	**16**	4	9	**25**	5	9
8	1	3	**17**	5	8			
9	1	6	**18**	4	7			

Table 6. Case1 - Single load case for the 25-bar truss

Node	P_x [N]	P_y [N]	P_z [N]
1	4448.22	–44482.22	–44482.22
2	0.00	–44482.22	–44482.22
3	2224.11	0.00	0.00
6	2668.93	0.00	0.00

The allowable stresses for each member are:

$$|\sigma_{imin}| = \sigma_{imax} = 275.79 \cdot 10^6 \text{ Pa} = 40 \text{ ksi}, \ i=1,\ldots,10 \tag{16}$$

and the maximum displacements for each node in *x*, *y* and *z* directions are:

$$|\delta_{imin}| = \delta_{imax} = 8.89 \cdot 10^{-3} \text{ m} = 0.35 \text{ in}, \ i=1,\ldots,10 \tag{17}$$

The minimum and maximum cross sectional areas are:

$$|A_{imin}| = 6.45 \cdot 10^{-5} \text{ m}^2 = 0.1 \text{ in}^2, \ i=1,\ldots,25 \tag{18}$$

and

$\lvert A_{imax} \rvert$ =219.35·10⁻⁵ m²= 3.4 in², i=1,…,25	(19)

Three optimisation algorithms are utilised to compare *c-mFOA*. A brief description of each algorithm follows.

- The first algorithm is *Differential Evolution* (*DE*) [26].
- The second one is a new version of DE called *adaptive elitist DE* algorithm (*aeDE*) [26]. There are three main differences with respect to the original one. In the mutation phase, an adaptive technique based on the deviation of objective function between the best individual and the whole population in the previous generation is proposed to select a suitable mutation operator. An elitist selection technique is utilized to increase the convergence rate in the selection phase. Finally, a rounding technique is integrated for problems with discrete design variables.
- The third algorithm is the so called *Mine Blast Algorithm* (*MBA*) [27]. The fundamental concepts and ideas of MBA are derived from the explosion of mine bombs in real world scenarios.

The comparison results are summarised in Table 7.

Table 7. Case1- Optimisation results for the 25-bar truss

Variables		**Cross sectional area** [10^{-5} m^2]			
	Members	**Ho-Huu DE**	**Ho-Huu aeDE**	**Sadollah MBA**	**c-MFOA**
	1	6.45	6.45	6.45	6.45
	2,3,4,5	19.35	19.35	19.35	19.35
	6,7,8,9	219.35	219.35	219.35	219.35
	10,11	6.45	6.45	6.45	6.45
	12,13	135.48	135.48	135.48	135.48
	14,15,16,17	64.52	64.52	64.52	64.52
	18,19,20,21	32.26	32.26	32.26	32.26
	22,23,24,25	219.35	219.35	219.35	219.35
Weight$_{min}$ (kg)		219.93	219.93	219.92	219.93
Weight$_{avg}$ (kg)		219.95	220.00		219.97
Weight$_{std}$ (kg)		0.06	0.12		0.07
N$_{analyses}$		3500	1440	500	800

4.2.2 Case 2- Twenty-five bar space truss

In the second case the same truss as in case 1, described in 4.2.1, is utilised but now a multiple loading condition and different allowable stresses/displacements are considered. Tables 8 and 9 are listing the loading conditions and stresses/displacements allowable respectively. The minimum and maximum cross sectional areas are:

$\lvert A_{imin} \rvert$ =0.65·10^{-5} m^2= 0.01 in^2, i=1,…,25	(20)

and

$\lvert A_{imax} \rvert$ =219.35·10^{-5} m^2= 3.4 in^2, i=1,…,25	(21)

Table 8. Case 2- Multiple load case for the 25-bar truss

	Node	**P_x (N)**	**P_y (N)**	**P_z (N)**
Case 1	1	4448.22	44482.22	–22241.11
	2	0.00	44482.22	–22241.11
	3	2224.11	0.00	0.00
	6	2224.11	0.00	0.00
Case 2	1	0.00	88964.43	–22241.11
	2	0.00	–88964.43	–22241.11

Table 9. Stresses allowable tensile (T) and compressive (C) for each element (EL)

El.	**T [MPa]**	**C [MPa]**	**El.**	**T [MPa]**	**C [MPa]**	**El.**	**T [MPa]**	**C [MPa]**
1	275.79	–241.95	**10**	275.79	–241.95	**19**	275.79	–47.98
2	275.79	–79.91	**11**	275.79	–241.95	**20**	275.79	–47.98
3	275.79	–79.91	**12**	275.79	–241.95	**21**	275.79	–47.98
4	275.79	–79.91	**13**	275.79	–241.95	**22**	275.79	–76.41
5	275.79	–79.91	**14**	275.79	–46.60	**23**	275.79	–76.41
6	275.79	–119.31	**15**	275.79	–46.60	**24**	275.79	–76.41
7	275.79	–119.31	**16**	275.79	–46.60	**25**	275.79	–76.41
8	275.79	–119.31	**17**	275.79	–46.60			
9	275.79	–119.31	**18**	275.79	–47.98			

Here, *c-mFOA* is compared to the:

- *Flower Pollination Algorithm* (*FPA*) [28]. FPA can efficiently combine local and global searches, is inspired by cross-pollination and self-pollination of flowering plants, to perform local and global searches respectively. In addition an iterative constraint handling strategy where trial designs are accepted or rejected based on the allowed amount of constraint violation is utilised.
- *Teaching-Learning-based Optimization* (*TLBO*) algorithm [29]. The method makes use of the analogy between the learning process of learners and searching for designs to optimization problems. Learning can be accomplished either from the teacher or by the interaction between learners.
- *Colliding Bodies Optimization* (*CBO*) algorithm [30]. The algorithm is based on the physics describing collisions between bodies. Here each agent represents a body. After a collision between two moving bodies takes place, these are separated and moved to new positions with new velocities. This process is repeated until a termination criterion is satisfied.

The results are summarised in Table 10.

Table 10. Case2 - Optimisation results for the 25-bar truss

Variables		**Cross sectional area [10^{-5} m^2]**			
	Members	**Bekdas FPA**	**Degertekin TLBO$_{best}$**	**Kaveh CBO**	**c-MFOA**
	1	0.65	0.65	0.65	1.98
	2,3,4,5	118.09	133.59	137.37	127.56
	6,7,8,9	205.33	190.73	186.18	195.51
	10,11	0.65	0.65	0.65	0.65
	12,13	0.65	0.65	0.65	0.66
	14,15,16,17	45.26	44.45	43.81	41.33
	18,19,20,21	111.37	104.55	103.70	108.10
	22,23,24,25	165.85	172.65	173.68	174.83
Weight$_{min}$ (kg)		246.73	246.70	246.35	247.04
Weight$_{avg}$ (kg)		246.99	-	246.78	247.19
Weight$_{std}$ (kg)		0.27	-	0.13	0.15
N$_{analyses}$		8149	15318	9090	800

4.2.3 Case 3- Thirty-seven bar planar truss with frequency constraints

This is considered as a constrained truss optimization problem on size and shape. Figure 7 shows a sketch of the 37-bar truss. All nodes of the upper chord are allowed to vary in the y-axis - symmetry about Y-axis has to be maintained - and all the diagonal and upper chord bars are allowed to vary its cross-sectional areas. No upper bound exists for the cross sectional area but the minimum is:

$$|A_{imin}| = 1\cdot10^{-4}\text{ m}^2,\ i=1,\ldots,37 \qquad (22)$$

The bottom bar elements have fixed cross sectional areas of $A_f=4\cdot 10^{-3}$ m^2. Masses of m_{add} = 10 kg are attached to each of the bottom nodes. The remaining bars are modelled as elements with initial sectional areas of:

$$|A_{ki}| = 1\cdot10^{-4}\text{ m}^2,\ k=0 \qquad (23)$$

The material properties for the bar elements are $E = 2.1\cdot10^{11}$ N/m^2 and ρ = 7800 kg/m^3. The first three natural frequencies are constrained, see Equation (24):

$$\begin{aligned}&\omega_1\geq20\text{ Hz}\\&\omega_2\geq40\text{ Hz}\\&\omega_3\geq60\text{ Hz.}\end{aligned} \qquad (24)$$

The optimisation problem involves three frequency constraints and nineteen design variables (five shape variables + 14 sizing variables).

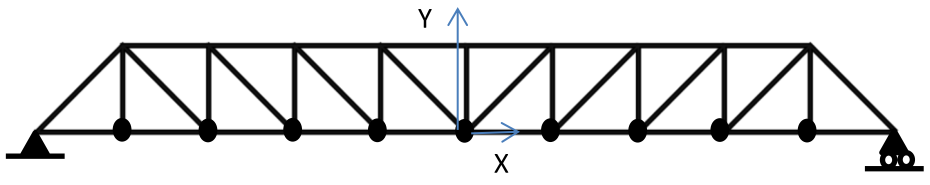

Figure 7. Case 3: 37 bar truss size and shape optimisation problem

In the last case study *c-mFOA* is compared to the:

- *Ray Optimisation* (*RO*) method [31]. In RO, each agent is modelled as a ray of light that moves in the search space in order to find the global or sub-global optimum solution.
- Multi-class teaching-learning-based optimization (MC-TLBO) [32]. MC-TLBO employs a two-step procedure. In the first step, parallel classes explore the search space; while in the second step, the best solutions of the parallel classes form a super class to be the initial population for a TLBO.
- Particle Swarm Optimisation [33].

The results are summarised in Table 11, while the shape of the optimised structure is shown in Figure 8.

Table 11. Case 3 - Optimisation results for the 37-bar truss

Variables			**Cross sectional area (10^{-4} m^2)**			
	Position [m]	**Members**	**Kaveh RA**	**Farshchin MC-TLBO**	**Gomes PSO**	**c-MFOA**
	Y_3, Y_{19}		1.0010	0.98300	0.9637	1.0108
	Y_5, Y_{17}		1.3909	1.38030	1.3978	1.3860
	Y_7, Y_{15}		1.5893	1.56450	1.5929	1.5608
	Y_9, Y_{13}		1.7507	1.68710	1.8812	1.6802
	Y_{11}		1.8336	1.75900	2.0856	1.7580
		A_1, A_{27}	3.0124	2.99127	2.6797	3.1997
		A_2, A_{26}	1.0623	1.00054	1.1568	1.0025
		A_3, A_{24}	1.0005	1.00415	2.3476	1.0000
		A_4, A_{25}	2.2647	2.59576	1.7182	2.5875
		A_5, A_{23}	1.6339	1.21394	1.2751	1.0895
		A_6, A_{21}	1.6717	1.14226	1.4819	1.1261
		A_7, A_{22}	2.0591	2.31699	4.6850	2.5624
		A_8, A_{20}	1.6607	1.50998	1.1246	1.4121
		A_9, A_{18}	1.4941	1.51723	2.1214	1.5758
		A_{10}, A_{19}	2.4737	2.27223	3.8600	2.2461
		A_{11}, A_{17}	1.5260	1.21117	2.9817	1.0694
		A_{12}, A_{15}	1.4823	1.27385	1.2021	1.3193
		A_{13}, A_{16}	2.4148	2.49338	1.2563	2.3846
		A_{14}	1.0034	1.00000	3.3276	1.0001
Weight$_{min}$ (kg)			364.04	359.966	377.20	360.07

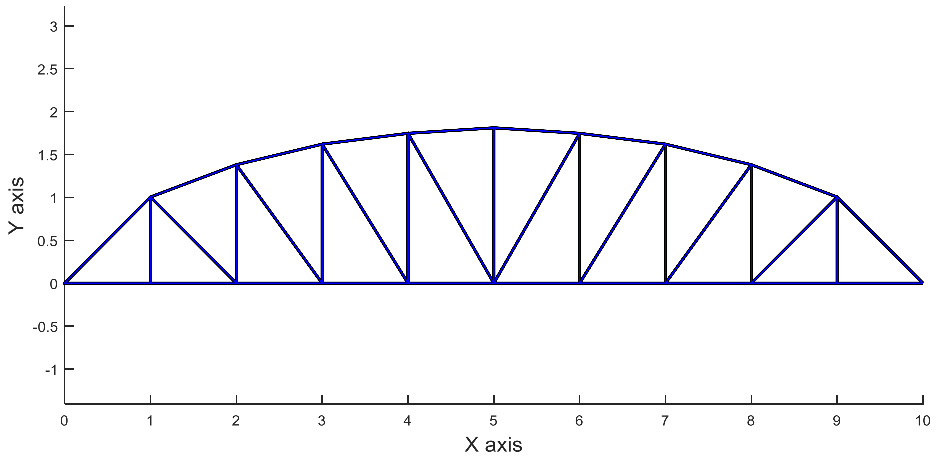

Figure 8. Optimised truss shape in case 3

5. Discussion

5.1 Analysis of results

From the mathematical benchmark study it is concluded that *c-mFOA* perfoms significantly better than *GA*, *PSO* and *SA*. The calculated mean of best values and standard deviation are lower, an indication that *c-mFOA* presents a better and more robust. In all cases – except function 9 – the proposed algorithm achieves finding the optimum within 16000 iterations. A sensitivity analysis, in which only parameter *M* was varied, has shown that it is possible to calculate the optimum value within 16000 iterations. *M* is a parameter with which we can enlarge or decrease the search space.

Two structural optimisation problems where only stress and displacement constraints are active were studied. In the first case *c-mFOA* performs as well as most of the other state-of-the-art optimisation algorithms. The same optimum is found for approximately the same number of function evaluations. In the second and most complex case *c-mFOA* performs better than the rest except for the TLBO algorithm, which achieves a slightly better result. In greater detail, the best value calculated is 0.28% worse than *TLBO* and the average of best values is worse only by only 0.13%.

In the third case study, which is also the most challenging, *c-mFOA* performs significantly better than the rest algorithms except for *MC-TLBO*, which achieves the same result.

In conclusion, the proposed algorithm has been compared to twelve state-of-the-art and three standard optimisation algorithms (e.g. those found in Matlab Optimisation Toolbox) for a range of benchmark problems. The benchmarks concern noisy functions with multiple minima, varying gradients and multiple constraints. *c-mFOA* achieves better results than all other algorithms except for TLBO, where it achieves the same performance. It is highlighted that the results obtained using *c-mFOAa* are for the same parameter setting. A random starting condition was used to initialise the population; thus the performance achieved is independent from the starting solution.

As shown, only one parameter *M* is required to change the algorithm's convergence rate.

5.2 Parallel processing performance

6. Conclusions

Optimisation algorithms are employed routinely in engineering to improve designs, reduce cost, improve efficiency, reduce weight etc. However, the standard optimisation tools used by practising engineers today are not adequate in solving complex, multi-parameter problems.

Fruit flies are extremely efficient in finding food, although their brain is rather simplistic. From an optimisation point of view this is extremely interesting as it means that this is achieved by simple means. In this study, a new fruit fly optimisation algorithm is presented based on the findings of recent fruit fly behaviour study. The proposed version modifies the initial algorithm by introducing the concepts of visual stimulation and reflex delay. Additionally, the population is generated by a uniform discrete distribution; a property that has increased significantly the efficiency of the algorithm to treat multi-parameter and complex problems.

1. The proposed algorithm has been compared to three standard and twelve-state-of-the art optimisation algorithms. The benchmark tests employed are ten noisy and unconstrained mathematical functions, two structural optimisation problems with stress and displacement constraints, and one structural optimisation problem with frequency constraints. *c-mFOA* achieves better results for less or the same number of analyses compared with other algorithms; except for the *Teaching Learning Based Optimisation* algorithm, where it achieves exactly the same performance.

2. It is highlighted that the results obtained using *c-mFOAa* are for the same parameter setting and thus no tuning was required. The performance achieved is independent from the initial condition as this was randomly selected. As shown, only one parameter *M* is required to change the algorithm's convergence rate and this can be intuitively chosen. All these features make the proposed algorithm attractive for implementation in standard engineering tools.

In the future it is foreseen to enlarge the range of applications for the fruit fly algorithm as well as to conduct a more detailed comparison to the *Teaching Learning Based Optimisation.* In addition, there are plans to further evaluate and improve the algorithm to work with highly n-dimensional search and more complex domain spaces.

Acknowledgments
MEF is grateful for funding from the Lloyd's Register Foundation, a charitable foundation helping to protect life and property by supporting engineering-related education, public engagement and the application of research.

References

1. Liu, D., Lohse-Busch, H., Toropov, V., Hühne, C., Armani, U. Detailed design of a lattice composite fuselage structure by a mixed optimization method (2015) Engineering Optimization, pp. 1-14
2. Kanarachos, S., Kanarachos, A. Intelligent road adaptive suspension system design using an experts' based hybrid genetic algorithm (2015) Expert Systems with Applications, 42 (21), pp. 8232-8242
3. Alirezaei, M., Kanarachos, S., Scheepers, B., Maurice, J.P. Experimental evaluation of optimal Vehicle Dynamic Control based on the State Dependent Riccati Equation technique (2013) Proceedings of the American Control Conference, art. no. 6579871, pp. 408-412.
4. https://en.wikipedia.org/wiki/List_of_optimization_software, accessed on 21/03/2016
5. Principles of Optimal Design 2ed: Modeling and Computation Paperback – 21 Aug 2008 by Panos Y. Papalambros, Cambridge University Press; 2 edition (21 Aug. 2008) ISBN-13: 978-0521627276
6. Wong, K.-C. Evolutionary multimodal optimization: A short survey (2015) Advances in Evolutionary Algorithms Research, pp. 1-15
7. Engelbrecht, A.P. Particle swarm optimization with crossover: a review and empirical analysis (2016) Artificial Intelligence Review, 45 (2), pp. 131-165.
8. Saka, M.P., Hasançebi, O., Geem, Z.W. Metaheuristics in structural optimization and discussions on harmony search algorithm (2016) Swarm and Evolutionary Computation, . Article in Press.
9. Bauer, J., Hengsbach, S., Tesari, I., Schwaiger, R., Kraft, O. High-strength cellular ceramic composites with 3D microarchitecture (2014) Proceedings of the National Academy of Sciences of the United States of America, 111 (7), pp. 2453-2458
10. Zhang, Y., Hou, Y., Liu, S. A new method of discrete optimization for cross-section selection of truss structures (2014) Engineering Optimization, 46 (8), pp. 1052-1073.
11. Wei, L., Tang, T., Xie, X., Shen, W. Truss optimization on shape and sizing with frequency constraints based on parallel genetic algorithm (2011) Structural and Multidisciplinary Optimization, 43 (5), pp. 665-682.
12. Gomes, H.M. Truss optimization with dynamic constraints using a particle swarm algorithm (2011) Expert Systems with Applications, 38 (1), pp. 957-968.
13. Ali Mortazavi,Vedat Toğan, Simultaneous size, shape, and topology optimization of truss structures using integrated particle swarm optimizer (2016), Structural and Multidisciplinary Optimization, pp. 1-22, accessed on line: http://link.springer.com/article/10.1007/s00158-016-1449-7
14. Kaveh, A., Bakhshpoori, T. Water Evaporation Optimization: A novel physically inspired optimization algorithm (2016) Computers and Structures, 167, pp. 69-85.
15. Camp, C.V., Farshchin, M. Design of space trusses using modified teaching-learning based optimization (2014) Engineering Structures, 62-63, pp. 87-97.
16. Qisong, Q., Gening, X., Xiaoning, F., Jun, W. A new type bionic global optimization: Construction and application of modified fruit fly optimization algorithm (2015) Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 229 (9), pp. 1614-1621.

17. Lin, W.-Y. A novel 3D fruit fly optimization algorithm and its applications in economics (2015) Neural Computing and Applications, 23 p. Article in Press.
18. Pan, W.-T. A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example (2012) Knowledge-Based Systems, 26, pp. 69-74.
19. Jun-qing Li, Quan-ke Pan, Kun Mao, P.N. Suganthan, Solving the steelmaking casting problem using an effective fruit fly optimisation algorithm, Knowledge-Based Systems, Volume 72, December 2014, Pages 28-36
20. Lianghong Wu, Cili Zuo, Hongqiang Zhang, A cloud model based fruit fly optimization algorithm, Knowledge-Based Systems, Volume 89, November 2015, Pages 603-617
21. Marko Mitić, Najdan Vuković, Milica Petrović, Zoran Miljković, Chaotic fruit fly optimization algorithm, Knowledge-Based Systems, Volume 89, November 2015, Pages 446-458
22. Liming Shen, Huiling Chen, Zhe Yu, Wenchang Kang, Bingyu Zhang, Huaizhong Li, Bo Yang, Dayou Liu, Evolving support vector machines using fruit fly optimization for medical data classification, Knowledge-Based Systems, Volume 96, 15 March 2016, Pages 61-75
23. Van Breugel, F., Dickinson, M.H. Plume-tracking behavior of flying drosophila emerges from a set of distinct sensory-motor reflexes (2014) Current Biology, 24 (3), pp. 274-286
24. http://www.futurity.org/fruit-flys-tiny-brain-finds-food-well/, accessed on 21/03/2016
25. http://www.nncn.de/en/news/Forschungsergebnisse-en/memories-of-fruit-fly-larvae-are-more-complex-than-thought, accessed on 21/03/2016
26. V. Ho-Huu, T. Nguyen-Thoi, T. Vo-Duy, T. Nguyen-Trang, An adaptive elitist differential evolution for optimization of truss structures with discrete design variables, Computers & Structures, Volume 165, March 2016, Pages 59-75
27. Ali Sadollah, Ardeshir Bahreininejad, Hadi Eskandar, Mohd Hamdi, Mine blast algorithm for optimization of truss structures with discrete variables, Computers & Structures, Volumes 102–103, July 2012, Pages 49-63
28. Gebrail Bekdaş, Sinan Melih Nigdeli, Xin-She Yang, Sizing optimization of truss structures using flower pollination algorithm, Applied Soft Computing, Volume 37, December 2015, Pages 322-331
29. S.O. Degertekin, M.S. Hayalioglu, Sizing truss structures using teaching-learning-based optimization, Computers & Structures, Volume 119, 1 April 2013, Pages 177-188
30. A. Kaveh, V.R. Mahdavi, Colliding Bodies Optimization method for optimum design of truss structures with continuous variables, Advances in Engineering Software, Volume 70, April 2014, Pages 1-12
31. A. Kaveh, M. Khayatazad, Ray optimization for size and shape optimization of truss structures, Computers & Structures, Volume 117, February 2013, Pages 82-94
32. M. Farshchin, C.V. Camp, M. Maniat, Multi-class teaching–learning-based optimization for truss design with frequency constraints, Engineering Structures, Volume 106, 1 January 2016, Pages 355-369
33. Gomes, H.M. Truss optimization with dynamic constraints using a particle swarm algorithm (2011) Expert Systems with Applications, 38 (1), pp. 957-968.