Optimal design of a quadratic parameter varying vehicle suspension system using contrast-based Fruit Fly Optimisation

Stratis Kanarachos, Arash Moradinegade Dizqah, Georgios Chrysakis, Michael E. Fitzpatrick

Original citation:
Kanarachos, S; Moradinegade Dizqah, A; Chrysakis, G. and Fitzpatrick, M.E. (2017) Optimal design of a quadratic parameter varying vehicle suspension system using contrast-based Fruit Fly Optimisation *Applied Soft Computing* (in press). DOI: 10.1016/j.asoc.2017.11.005

http://dx.doi.org/10.1016/j.asoc.2017.11.005

Elsevier

Title:

Optimal design of a quadratic parameter varying vehicle suspension system using contrast-based Fruit Fly Optimisation

Abstract:
In the UK, in 2014 almost fifty thousand motorists made claims about vehicle damages caused by potholes. Pothole damage mitigation has become so important that a number of car manufacturers have officially designated it as one of their priorities. The objective is to improve suspension shock performance without degrading road holding and ride comfort. In this study, it is shown that significant improvement in performance is achieved if a clipped quadratic parameter varying suspension is employed. Optimal design of the proposed system is challenging because of the multiple local minima causing global optimisation algorithms to get trapped at local minima, located far from the optimum solution. To this end an enhanced Fruit Fly Optimisation Algorithm – based on a recent study on how well a fruit fly's tiny brain finds food – was developed. The new algorithm is first evaluated using standard and nonstandard benchmark tests and then applied to the computationally expensive suspension design problem. The proposed algorithm is simple to use, robust and well suited for the solution of highly nonlinear problems. For the suspension design problem new insight is gained, leading to optimum damping profiles as a function of excitation level and rattle space velocity.

Authors: Stratis Kanarachos, Arash Moradinegade Dizqah, Georgios Chrysakis, Michael E. Fitzpatrick

Affiliation: Faculty of Engineering, Environment & Computing, Coventry University, Coventry, CV12JH, UK, stratis.kanarachos@coventry.ac.uk

Corresponding author. Stratis Kanarachos, Faculty of Engineering, Environment & Computing, Coventry University, Coventry, CV1 2JH, UK, Tel.: +44(0)2477657720; fax: +44(0)2477657720
E-mail address: stratis.kanarachos@coventry.ac.uk.

Keywords: Swarm intelligence, Fruit Fly Optimisation, suspension design, potholes

1. Introduction

According to statistics, in the UK a car sustains pothole damage every 11 minutes resulting in 50,000 motorists making claims about vehicle damage in 2014 [1]. The poor weather conditions during recent winters have left many European roads covered with potholes at a time when money for repairs is limited [2]. The scale of pothole vehicle damage problem has been intensified due to the low-profile tyre usage trend. The problem has become so important that a number of car manufacturers designated it as one of their priorities [3]. In this context, suspension design needs further improvement to meet today's challenges.

Recent suspension design studies focus on comfort, handling and stability, however they do not consider how to mitigate pothole damages [4]. In principle, passive or active linear suspension systems can reduce shock loads and chassis

accelerations by using 'softer' springs, thus allowing more suspension travel. This comes at the expense of deteriorated road-holding properties, due to the increased tyre load oscillations, and increased probability of hitting the suspension limits. The conflicting performance objectives when linear control is applied, necessitate the investigation of nonlinear controllers. In [5] a fuzzy-PID controller was implemented and compared to a fixed gain PID controller. The results were promising and further improved when the fuzzy-PID controller design problem was formulated as an optimization problem, where each point represented a rule set, membership function, and corresponding system behaviour [6]. The optimized set of values was computed by combining Particle Swarm Optimisation (PSO) and Q-learning. In [7] the fuzzy-PID controller was fine-tuned by combining Cultural and Niche optimisation algorithms. In studies where the classical Ziegler-Nichols gain tuning method was applied moderate results were reported [8]. Furthermore, the robustness of standard PID suspension control was examined and found to be under performing in [9].

Optimal Control extends standard PID control design by considering systems with multiple outputs [10]. An adaptive suspension controller that dynamically interpolates a set of Linear Quadratic Regulators (LQG) was proposed in [11]. In [12] the State Dependent Riccati Equation (SDRE) controller design technique was assessed. In [13] a LQG suspension controller was first designed. Subsequently, the commanded force was clipped to match the damper's controllability range (dampers can only generate negative forces). The controller parameters were determined using the genetic algorithm NSGA II. In [14]-[15], Brezas *et al.*, applied Optimal Control Theory to simultaneously optimise the ride and handling vehicle behaviour. Clipped Optimal Control was compared to standard LQG and found to be performing better. This was a very interesting result because LQG, as an active suspension control concept, requires a more complex and energy consuming system compared to semi-active suspension.

Different Skyhook control concepts were studied in [16], including Skyhook two-state damper control, Skyhook linear approximation damper control, and mixed Skyhook-acceleration-driven damper. There is a trade-off between road holding and comfort when fixed gain Skyhook control is applied [17]. Adaptive Skyhook, with the gains being a function of the road condition, was investigated in [18]. In [19] Skyhook controller gains were tuned by matching the damper force to the output of a Linear Quadratic Regulator. The combination of Skyhook control with a neural network-based feedforward term was evaluated in [20]. In conclusion, Skyhook control cannot reduce simultaneously the resonance peak of the sprung and un-sprung masses [21]-[22]. Skyhook damping also inherits other problems, notably water hammer and/or chucking [23]-[24].

In [25] Linear Parameter Varying (LPV) control was implemented and the controller gains were obtained solving a Linear Matrix Inequalities (LMI) problem. In [26] the concept was further refined by including a scheduling parameter as a function of the difference between commanded and attainable forces. A velocity dependent LPV controller was proposed in [27]. A method for the automated generation of LPV systems was presented in [28], while in [29] the concept was extended to uncertain LPV systems. Hybrid or data based controllers have also been proposed. Examples include [30] where a PID controller and a three-layered feedforward neural network were combined. The Levenberg–Marquardt (LM) algorithm was employed to train the neural network. In [31] a recurrent neural network (RNN) was investigated. In [32] a Magneto-Rheological (MR) damper based on a feedforward neural network was proposed. In [33] two fuzzy logic controllers were combined and tuned using the derivation of a Pareto front. A rule-based nonlinear suspension system was designed

and fine-tuned using GA in [34]. Particle Swarm Optimisation (PSO) was employed to tune a feedback linearization scheme in [35]. Finally, a real-time grey-prediction algorithm was employed in [36] while Particle Swarm Optimisation and Genetic Algorithms were combined to derive the Pareto optimal design of a five-degree-of-freedom vehicle vibration model [37].

In conclusion, most approaches cannot overcome the problem of simultaneously optimizing the sprung and un-sprung mass responses, especially when a broad range of external loads including singular disturbances is considered. In this study, it is shown that it is possible to overcome this limitation with a clipped quadratic parameter varying suspension system. Tuning of the nonlinear suspension system was achieved by applying an enhanced Fruit Fly Optimisation Algorithm (FOA), a new population-based heuristic algorithm discovered through simulation of the intelligent foraging behaviour of fruit flies [38]-[42]. The proposed contrast-based Fruit Fly Optimisation Algorithm (c-FOA) is first studied and evaluated using standard and nonstandard benchmark tests and then applied to the suspension design problem. It is shown that c-FOA is simple to use, robust and well suited for the solution of computationally expensive optimisation problems. To our knowledge this is the first time where FOA is applied to the optimal design of a suspension system.

The paper is structured as follows. In Section 2 the contrast-based Fruit Fly Optimisation Algorithm is presented in detail and discussed in relation to other Swarm Intelligence algorithms. In Section 3, 14 benchmark tests are used to study the new algorithm and compare its performance to standard optimisation algorithms including the Genetic Algorithm, Simulated Annealing, Particle Swarm Optimisation, Differential Evolution, Artificial Bee Colony and the original Fruit Fly Optimisation Algorithm. The quadratic parameter varying suspension system problem, a computationally intensive problem, is formulated in Section 4. In Section 5 the numerical results using c-FOA, Genetic Algorithm and Particle Swarm Optimisation are analysed and discussed. In the last section conclusions and future research directions are presented.

2. The contrast-based Fruit Fly Optimisation Algorithm (*c-FOA*)

2.1 A short introduction to *Fruit Fly Optimisation*

Drosophila is a genus of small flies, belonging to the family Drosophilidae, whose members are often called “fruit flies” or (less frequently) pomace flies, vinegar flies, or wine flies, a reference to the characteristic of many species to linger around overripe or rotting fruit [43]. Fruit flies can smell and locate a food source even if this is 40 km away. This performance is remarkable as their brain has only 100,000 neurons, compared to house fly brains which have 300,000 neurons and human brains with 100 billion [44]. The combination of food search efficiency and reduced complexity makes it very interesting from a biological and optimisation point of view.

Pan was the first drive the Fruit Fly Optimisation Algorithm (FOA) based on the food finding characteristics of a fruit fly swarm [45]. A schematic of the food searching process is shown in Figure 1. The main steps involved in standard FOA are:

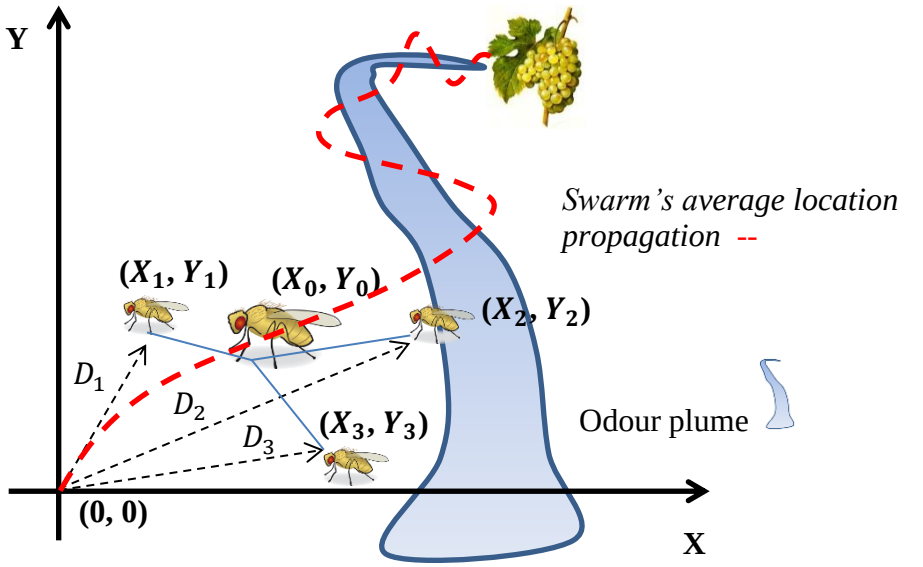

Figure 1. Fruit fly swarm in search for food

- Step 1: Initialization. The average swarm location $[X_0, Y_0]$ the maximum number of iterations K and the size of the swarm N are defined.

- Step 2: Swarm generation. For $i = 1, .., N$ a new population of fruit flies is generated according to:

$$X_i = X_0 + \text{rand}$$
$$Y_i = Y_0 + \text{rand} \quad (1)$$

- Step 3: Localisation. Each fruit fly is assigned a value S_i based on how close the fruit fly $[X_i, Y_i]$ is to the origin:

$$D_i = \sqrt{X_i^2 + Y_i^2} \quad (2)$$

$$S_i = \frac{1}{D_i} \quad (3)$$

S_i is a reciprocal function and therefore sensitive when $D_i \sim 0$. Even a slight change ΔD_i can result in a large difference ΔS_i. This attribute resembles the fruit fly's ability to search food at large distances.

- Step 4: Objective function calculation. For each fruit fly the corresponding smell concentration $Smell_i = f(S_i)$ is calculated, where f is the objective function.

- Step 5: Best member identification. The fruit fly with the highest smell concentration in the swarm is identified:

$$[\, X_b \;\; Y_b \,] \;\rightarrow\; Smell_b = \max(Smell_i) \tag{4}$$

- Step 6: Average location selection. The 'best' fruit fly is compared to the existing average location:

$$\begin{gathered} if\ Smell_b > Smell_0 \\ \text{then } X_0 = X_b \text{ and } Y_0 = Y_b \end{gathered} \tag{5}$$

- Step 7: Termination phase. Is the maximum number K iterations reached? If yes stop, otherwise return to Step 2.

The original FOA has several drawbacks. For example, fruit flies are only attracted in the vicinity of the current best location $[X_0, Y_0]$. This may well be a local extreme. Therefore, it is very probable for a fruit fly swarm to get trapped around a local minimum.

2.2 The proposed contrast-based Fruit Fly Optimisation Algorithm: c-FOA

A recent study of more than 70 hours of fruit flies' motion showed that fruit flies primarily detect food by tracking odour plumes [46]. A plume's motion can be chaotic in the presence of external disturbances, for example an airstream, and may prohibit a fruit fly from detecting the food source [47]. When this occurs fruit flies start to search for visually attractive features and in particular they explore objects with visual contrast. They land, and if where they land is not something to eat, they continue the search. A glass of wine is a contrasting shape, like fruit, that would merit their attention.

The study concluded that in order to localize an odour source, flies exhibit three iterative, independent and reflex-driven behaviours, which remain constant through repeated encounters of the same stimulus:

(a) 190 ± 75 ms after encountering a plume, flies increase their flight speed and turn upwind, using visual cues such as stripes to help them determine wind direction. Owing to this substantial response delay, flies may pass beyond the plume shortly after entering it.

(b) 450 ± 165 ms after losing the plume, flies initiate a series of vertical and horizontal casts, using visual cues to maintain a crosswind heading.

(c) After sensing an attractive odour, flies exhibit an enhanced attraction to visual features such as roundish objects, which increases their probability of finding the plume's source.

The previously described motion pattern is idealised and modelled, for the first time in this paper, using the proposed contrast-based Fruit Fly Optimisation Algorithm (c-FOA). c-FOA amends the original FOA by adding two new search phases: i) the delay detection and ii) visual feature detection. Figure 2 illustrates the proposed algorithm.

The main steps of the algorithm are as follows:

- Step I: Initialization. The average swarm location $[X_0,\ Y_0]$, the maximum number of iterations K, the size of the swarm N, the delay κ, the scaling factor M, and contraction parameter c are defined.

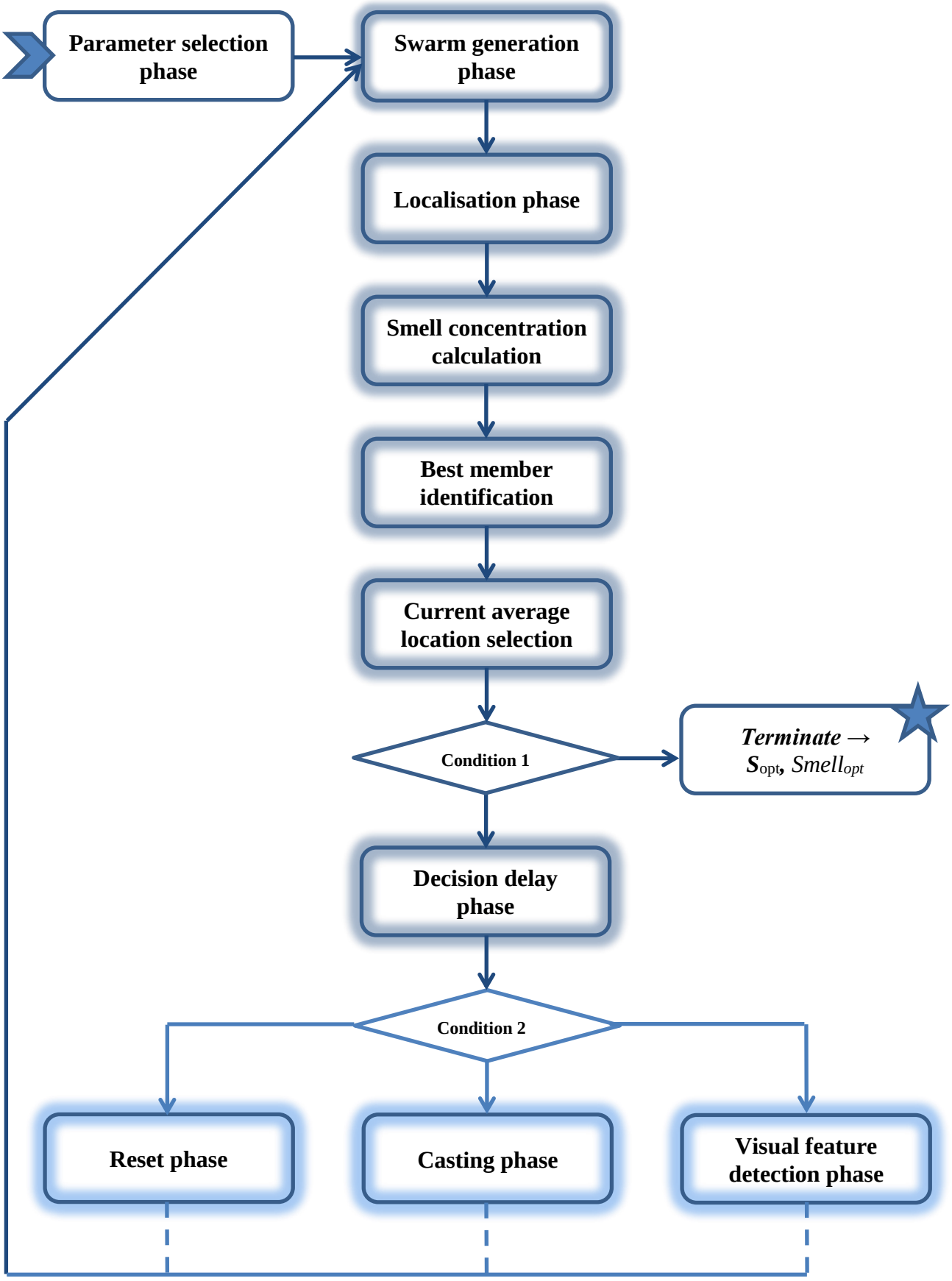

Figure 2. Proposed *c-FOA* algorithm

- Step II: Swarm generation. For $i = 1, .., N$ a new population of fruit flies is created through the following randomised process:

$$X_i = X_0 \cdot (1 + M_i \cdot (2 \cdot rand - 1)$$
$$Y_i = Y_0 \cdot (1 + M_i \cdot (2 \cdot rand - 1) \quad (6)$$

- Step III: Localisation. Each fruit fly is assigned a value S_i based on how close the fruit fly $[X_i, Y_i]$ is to the origin:

$$D_i = \sqrt{X_i^2 + Y_i^2} \quad (7)$$

$$S_i = \frac{1}{D_i} \quad (8)$$

- Step IV: Objective function calculation. The corresponding smell concentration $Smell_i = f(S_i)$ is for each fruit fly i, where f is the objective function.

- Step V: Best member identification. The fruit fly with the highest smell concentration in the swarm is identified:

$$Smell_b = \max(Smell_i) \rightarrow best\ fruit\ fly\ S_b \rightarrow location\ [X_b\ Y_b] \quad (9)$$

- Step VI: Average location selection. The best fruit fly is compared to the existing average location:

$$if\ Smell_b > Smell_0\ then\ X_0 = X_b\ and\ Y_0 = Y_b \quad (10)$$

- *Condition 1:*
  - *If* the maximum number of iterations K has been reached *then* terminate the optimisation process, retrieve the optimal fruit fly S_{opt} as well as the corresponding objective function value $Smell_{opt}$.
  - *Else*, continue to Step VII.

- Step VII: Decision delay. In this phase the fruit fly swarm does not change its food search strategy for κ iterations. This resembles the delay in decision-making that fruit flies exhibit.

- *Condition 2:*
  - *If* the smell concentration $Smell_0$ improves over the last κ iterations, then go to Step VIIIa.
  - *Else if* the smell concentration $Smell_0$ does not change over the last κ iterations, *then* go to Step VIIIb.
  - *If* the smell concentration $Smell_0$ worsens over the last $2 \cdot \kappa$ iterations, *then* go to Step VIIIc.

- Step VIIIa: Casting: Go to Step II without any change.

- Step VIIIb: Visual feature detection: The fruit fly with the worst smell concentration $Smell_w$ is identified and the fruit fly swarm becomes attracted to it. Reduce the scale factor M and go to Step II.

$$[\, X_w \;\; Y_w \,] \;\rightarrow\; Smell_w = \min(Smell_i) \tag{11}$$

$$X_0 = X_w \; and \; Y_0 = Y_w \tag{12}$$

$$M_{i+1} = c \cdot M_i \tag{13}$$

  where i is the current iteration.
  Eventually the flies will explore the area around the fruit fly with $Smell_w$. This resembles the visual cue fruit fly search behaviour.

- Step VIIIc: Reset: Return to the location that encountered the best smell concentration $Smell_0$ up to that point. Then go to Step II.

$$X_0 = X_b \; and \; Y_0 = Y_b \tag{14}$$

This resembles the memory function that fruit flies present.

2.3 Population-based optimisation techniques and c-FOA

c-FOA is a population-based optimisation technique classified under Swarm Intelligence, such as Particle Swarm Optimisation (PSO) [48] and Artificial Bee Colony (ABC) [49]. The main difference between Swarm Intelligence techniques and Evolutionary Algorithms is the strategy behind the creation of new individuals. In Evolutionary Algorithms, like the Genetic Algorithm (GA) and Differential Evolution (DE), operators like "mutation", "recombination" and "survival of the fittest" are employed, while in Swarm Intelligence the new individuals are created through interaction and information sharing between a member and the remaining population [50].

Particle Swarm Optimisation is inspired by flocks of birds swarming [51]. In greater detail, in Particle Swarm Optimisation the position of the individual members is randomly initialised [52]. Subsequently, the members x_i incrementally update their position x_{i+1} based on a weighted average that considers the member's previous speed v_i, the member's current position x_i, the member's previous best position p_i and the neighbouring group's best position p_g:

$$\begin{gathered} v_{i+1} = v_i + \varphi_1 \cdot \beta_1 \cdot (p_i - x_i) + \varphi_2 \cdot \beta_2 \cdot \left(p_g - x_i\right) \\ x_{i+1} = x_i + v_{i+1} \end{gathered} \tag{15}$$

where constants φ_1 and φ_2 determine the balance between the influence of the individual's knowledge and that of the group, while β_1 and β_2 are uniformly

distributed random numbers. The sign in the brackets results in an acceleration of the particles' motion towards the previously-known best points in the space. Different strategies for defining the neighbouring group exist and various modifications of the original Particle Swarm Optimisation have been proposed to make its performance more robust or more efficient in specific problems. Critical, for achieving a good trade-off between exploration and exploitation, is the memory velocity v_i. In some PSO versions it has been proposed to determine the new position x_{i+1} using:

$$v_{i+1} = \omega \cdot v_i + \varphi_1 \cdot \beta_1 \cdot (p_i - x_i) + \varphi_2 \cdot \beta_2 \cdot (p_g - x_i) \qquad (16)$$
$$x_{i+1} = x_i + v_{i+1}$$

where constant ω is a user-defined parameter.

c-FOA and Particle Swarm Optimisation share common characteristics. For example both algorithms initialise the swarm randomly and share the groups' best position to move a member towards a new position. Furthermore, the incremental displacement − difference between a member's old and new position − is restricted and depends on a term, which in the case of Particle Swarm Optimisation is the parameter $\omega \cdot v_i$ and in c-FOA the parameter M_i.

However, the two algorithms present fundamental differences as well. For example, in Particle Swarm Optimisation the movement of the individuals depends on a linear function, while in c-FOA on a reciprocal function. This causes significantly different swarm behaviour during the exploration phase. Another example is that in Particle Swarm Optimisation the new position depends randomly on a weighted average of the individual's and group's best position, while in c-FOA this depends only on the latter. The mechanism for handling noise is also different. In c-FOA the search strategy remains unchanged for a predetermined number of iterations, exactly like fruit flies, while in PSO the search direction is changed continuously. It is believed that fruit flies developed this decision delay mechanism to compensate for the chaotic movement of smells outdoors. Last but not least, in c-FOA, for the first time, a food search strategy that does not depend on the food source is presented. All Swarm Intelligence algorithms search for food on the basis of where the current food source lies (current lowest objective function value). However, the recent study on fruit fly behaviour revealed that fruit flies are attracted not only by smell (location of the food) but also by visually contrasting objects, which eventually may have nothing to do with a food source. Thus, the food search strategy is multi-stimuli. It is believed that fruit flies developed this behaviour through evolution and that this relies on fruit fly's knowledge that a food source has also visually contrasting traits.

3. Benchmark testing

Two studies are employed for demonstrating and analysing the performance of c-FOA, as well as comparing it to other standard optimisation algorithms. The purpose is to assess the algorithm performance for a fixed parameter set and compare it to standard state-of-the-art optimisation tools, commonly used by researchers and engineers.

The first study is a low-dimensional one with the main purpose to understand how c-FOA performs in the presence of noise and barrier functions. This is of great importance because in many engineering problems the optimal solution is dictated by

constraints. The second one concerns a high dimensional optimisation study, where the benchmark tests consist of multi-parameter functions, where the number of parameters is 20.

3.1 Low-dimensional study

Two sets of benchmark tests are employed in the low dimensional study. The first set concerns a group of noisy mathematical functions and evaluates the ability of the algorithm to avoid local minima. The second set amends the first one by introducing additional barrier functions. The low dimensional study is focused on the accuracy of c-FOA, therefore a large number of function evaluations is allowed [53].

c-FOA is evaluated and compared to two standard stochastic optimisation algorithms, the Genetic Algorithm (GA) and Simulated Annealing (SA). Both GA and SA depend on a number of parameters that may influence their performance in different types of problems. A sensitivity analysis was conducted and the best sets of parameters were applied to c-FOA, Genetic Algorithm and Simulated Annealing. Finally, the influence of population size on the optimisation accuracy was examined.

In Genetic Algorithm (GA) the members were randomly selected from a uniform distribution restricted in the problem-dependent design space. A floating-point representation was used. For each member the objective function value was calculated. The GA members were sorted according to their rank. 80% of the new generation was created by crossover and 5% progressed from the old generation. A stochastic uniform algorithm was used for the parent selection. The crossover operator used a weighted average of the parents to create children. The rest of the members were created by mutation. In mutation, new directions were randomly generated and were adaptive so that the design space was satisfied. The genetic algorithm terminated when the maximum number of function evaluations generations was reached, unless it stalled. This happened when for over 200 generations the objective function did not change significantly.

Simulated Annealing (SA) started with a random vector belonging to the problem-dependent design space. Two parameters – the temperature and re-annealing – determined the behaviour. Temperature controlled the extent of search. In this study the initial temperature was $T = 100$. The second one emulated the annealing process; following the generation of a number of new points, the temperature was raised to a higher value to restart the search and move out from local minima. If re-annealing is performed too fast this may not help the solver identify the global minimum. Here, the interval of $anneal = 50$ is chosen. An exponential cooling schedule was selected. The procedure terminated when the total number of function evaluations reached the maximum value.

3.1.1 Benchmark-1

A "noisy" one-dimensional mathematical function, described in Equation (17) is the first benchmark test.

$$f(x) = x^2 - 10 \cdot \cos(2 \cdot \pi \cdot x) + 10 \tag{17}$$

A plot of Equation (17) for $x \epsilon [-10,10]$ is shown in Figure 3. Although numerous local minima exist there is a clear trend towards the minimum $f(0) = 0$.

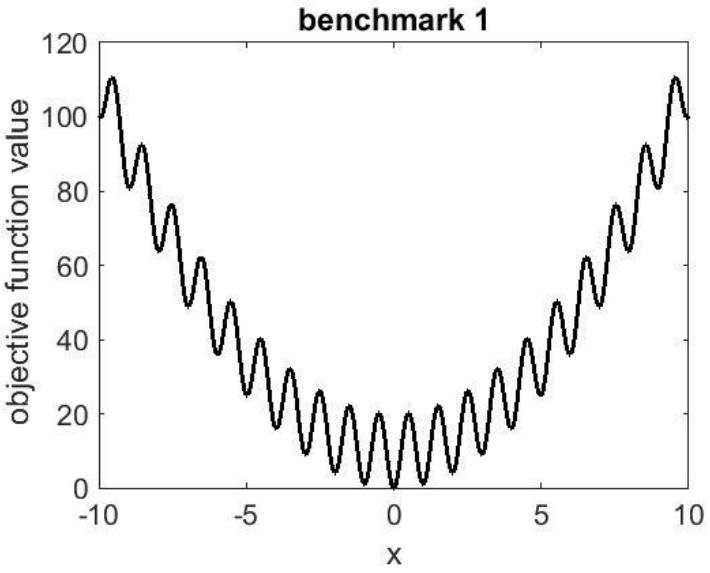

Figure 3. Function plot of benchmark-1: Noisy 1D function

The optimisation problem was solved for 30 repetitions. The parameters used were the following:

In c-FOA the initial value S_0 was randomly selected from $\epsilon\ [-10,10]$, $K = 1000, N = 50, \kappa = 5, M = 1$ and $c = 0.9$. In GA the population comprised 50 members. It was created randomly using a uniform distribution restricted in the design space $[-10, 10]$. The genetic algorithm terminated after 1000 generations unless it stalled. Simulated Annealing started with a random number $\epsilon\ [-10,10]$ and terminated after $N_{fun} = 50000$ function evaluations.

The mean value and standard deviation of the optimal values are listed in Table 1. As observed, all algorithms succeed in finding the optimal value. A typical convergence path for S_b using c-FOA is shown in Figure 4.

Table 1. Statistical evaluation of optimisation results for benchmark-1

Benchmark 1	S_{opt}		
	c-FOA	GA	SA
Mean value	-3.5×10^{-10}	-6.7×10^{-7}	1.2×10^{-3}
Standard deviation	9.4×10^{-10}	2.6×10^{-1}	6.6×10^{-3}

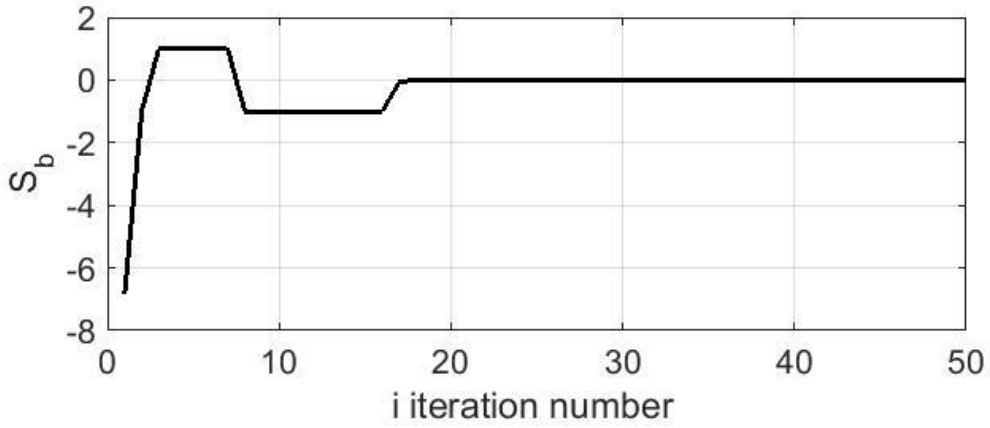

Figure 4. Benchmark-1: Example of convergence path for S_b using c-FOA

3.1.2 Benchmark-2

The second benchmark function, benchmark-2, amends the first one by introducing two barrier functions:

$$f(x) = x^2 - 10 \cdot \cos(2 \cdot \pi \cdot x) + 10 + penalty_1$$
$$penalty_1 = \begin{cases} 0, \ if\ x < -6 \\ 100 \cdot min(|x+3|, |x+6|), \ if -6 < x < -3 \\ 0, \ if -3 < x < 2 \\ 100 \cdot min(|x-2|, |x-4|), \ if\ 2 < x < 4 \\ 0, \ if\ x > 4 \end{cases} \quad (18)$$

A plot of Equation (18) for $x \in [-10,10]$ is shown in Figure 5. The minimum is located at $f(0)=0$.

The problem is solved for 30 repetitions. We keep the same optimisation settings as in Benchmark-1 except for the maximum number of function evaluations. In particular in c-FOA and GA the population size is $N = 20$, while in SA the maximum number of function evaluations is $N_{fun} = 20000$.

The results are listed in Table 2. A comparison to Table 1 reveals that although all optimisation algorithms succeed in finding the optimum value, the standard deviation values are increased. This is most probably due to the introduction of the barrier functions.

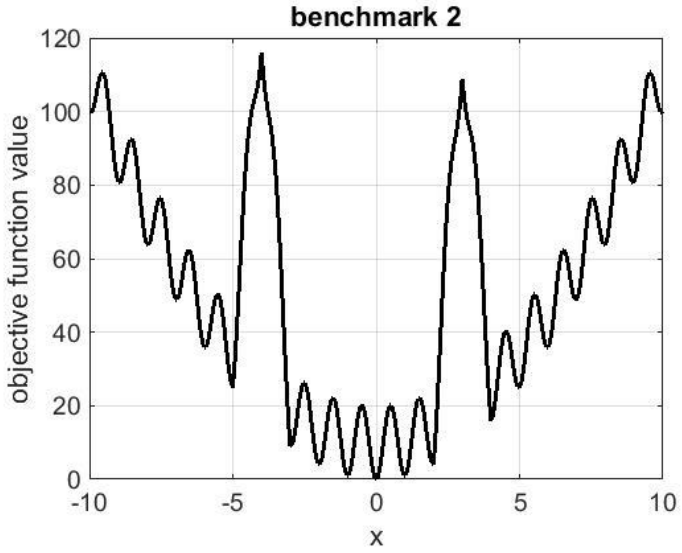

Figure 5. Function plot of benchmark 2: Noisy 1D function with barriers

Table 2. Statistical evaluation of optimisation results for benchmark-2

Benchmark 2	S_{opt}		
	c-FOA	*GA*	*SA*
Mean value	7.2×10^{-11}	-6.6×10^{-2}	1.1×10^{-4}
Standard deviation	1.3×10^{-9}	7.8×10^{-1}	1.9×10^{-2}

3.1.3 Benchmark-3

The third benchmark test is the well-known Rastrigin function:

$$f(x_1, x_2) = 20 + x_1^2 + x_2^2 - 10 \cdot \left(\cos(2 \cdot \pi \cdot x_1) + \cos(2 \cdot \pi \cdot x_2)\right) \quad (19)$$

Equation (19) is plotted in Figure 6, for $x_1 \epsilon [-5.12, 5.12]$ and $x_2 \epsilon [-5.12, 5.12]$. The global minimum is located at $f(0,0) = 0$.

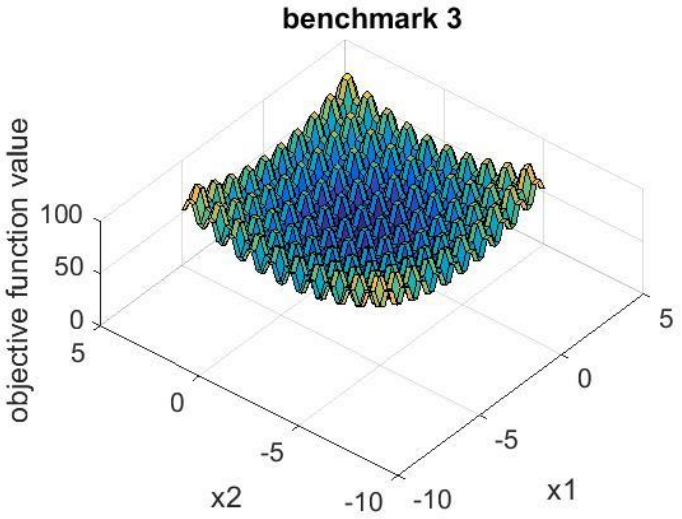

Figure 6. Function plot of benchmark-3: Rastrigin function

The optimisation problem is again solved for 30 repetitions. The design space was $x_1 \in [-5.12, 5.12]$ and $x_2 \in [-5.12, 5.12]$. The same optimisation parameters as in benchmark-2 were used. In SA the starting point was [–rand, rand]. The results are listed in Table 3 indicating that c-FOA almost always succeeded finding the optimal result. Additionally, the average results clearly indicate that in most cases GA and SS found the global minimum. The standard deviation is larger compared to the one achieved using c-FOA.

Table 3. Statistical evaluation of optimal results for benchmark-3

Benchmark 3	S_{opt}		
	c-FOA	GA	SA
Mean value	$[-2.6 \times 10^{-11}, 9.1 \times 10^{-11}]$	$[-1.9 \times 10^{-1}, 9.9 \times 10^{-2}]$	$[-1.0 \times 10^{-3}, -2.0 \times 10^{-3}]$
Standard deviation	$[1.6 \times 10^{-9}, 1.6 \times 10^{-9}]$	$[4.8 \times 10^{-1}, 7.9 \times 10^{-1}]$	$[3.6 \times 10^{-1}, 6.9 \times 10^{-1}]$

3.1.4 Benchmark-4:

The fourth benchmark test is the Rastrigin function augmented with barrier functions. Figure 7 illustrates the function, for $x_1 \epsilon [-5.12, 5.12]$ and $x_2 \epsilon [-5.12, 5.12]$. The global minimum is located at $f(0,0) = 0$. The optimisation problem is again solved for 30 repetitions with the same parameters as in Benchmark-3. The results are listed in Table 4.

A comparison to Table 3 shows that the presence of barrier functions degraded the performance of GA and SA, while c-FOA performed similarly to Benchmark-3. An example of how c-FOA converges is illustrated in Figure 8.

$$f(x_1, x_2) = 20 + x_1^2 + x_2^2 - 10 \cdot \big(\cos(2 \cdot \pi \cdot x_1) + \cos(2 \cdot \pi\, x_2)\big) + penalty1 + penalty2 \quad (20)$$

$$penalty_1 = \begin{cases} 0, \ \ if\ x_1 < -2 \\ 100 \cdot min(|x_1 + 2|, |x_1 + 1|), \ \ if -2 < x_1 < -1 \\ 0, \ \ if -1 < x_1 < 2 \\ 100 \cdot min(|x_1 - 2|, |x_1 - 3|), \quad if\ 2 < x_1 < 3 \\ 0, \quad if\ x_1 > 3 \end{cases}$$

$$penalty_2 = \begin{cases} 0, \ \ if\ x_2 < -2 \\ 100 \cdot min(|x_2 + 2|, |x_2 + 1|), \ \ if -2 < x_2 < -1 \\ 0, \ \ if -1 < x_2 < 2 \\ 100 \cdot min(|x_2 - 2|, |x_2 - 3|), \quad if\ 2 < x_2 < 3 \\ 0, \quad if\ x_2 > 3 \end{cases}$$

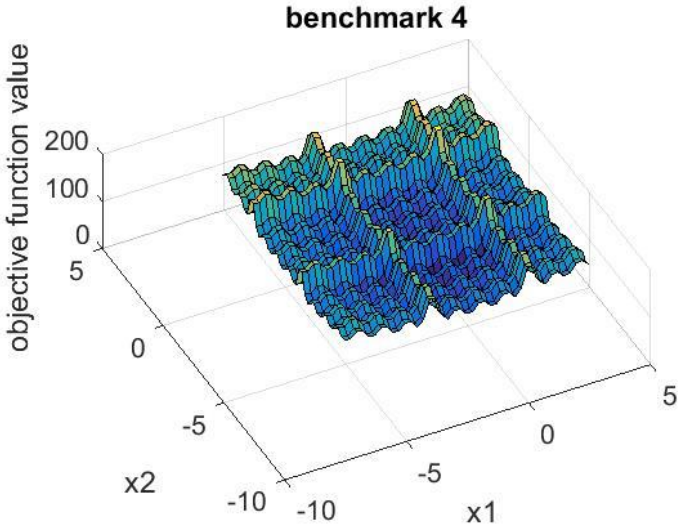

Figure 7. Function plot of benchmark-4: Rastrigin function with barriers

Table 4. Statistical evaluation of optimal results for benchmark-4

Benchmark 4	S_{opt}		
	c-FOA	GA	SA
Mean value	$[3.39 \times 10^{-10}, 7.46 \times 10^{-11}]$	$[1.3 \times 10^{-1}, -1.0 \times 10^{-3}]$	$[-1.6 \times 10^{-1}, 2 \times 10^{-3}]$
Standard deviation	$[1.24 \times 10^{-9}, 1.65 \times 10^{-9}]$	$[6.2 \times 10^{-1}, 9 \times 10^{-1}]$	$[7.4 \times 10^{-1}, 7.7 \times 10^{-1}]$

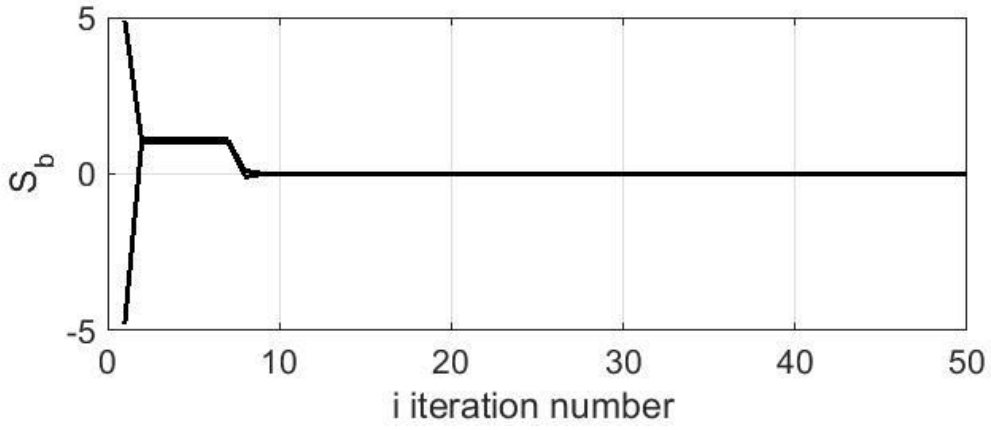

Figure 8. Benchmark 4: Example of convergence path for S_b using c-FOA

3.2 High dimensional study

In the high-dimensional study, c-FOA is evaluated and compared to the original Fruit Fly Optimisation Algorithm (FOA), Differential Evolution (DE), Particle Swarm Optimisation (PSO) and Artificial Bee Colony (ABC). All previously mentioned algorithms depend on a number of parameters that may influence their performance in different types of problems. A sensitivity analysis was conducted and the best sets of parameters were applied. It is highlighted that different versions of the above algorithms exist, however the purpose of this study is not to perform an exhaustive comparison between c-FOA and all different versions of Differential Evolution, Particle Swarm Optimisation and Artificial Bee Colony. The focus of this study is to compare the performance of the algorithms for a specified number of function evaluations, equal to $N_{fun} = 16000$.

The original FOA employed in this study is detailed in [54]. The population size was $N = 50$ members and the maximum number of iterations $K = 320$. For *c-FOA* the following parameters are selected: $K = 320, N = 50, \kappa = 5, M = 1$ and $c = 0.9$.

The DE version utilised is available from [55]. It is the standard DE algorithm augmented with dither to become more robust. The population was $N = 100$ members and a maximum number of $N_{iter} = 160$ iterations were allowed. The scale factor in the mutation operator was $F = 0.85$. The crossover probability in the crossover operator was $Cr = 1$. A uniform distribution was utilised to create the individuals within the bounds defined by the design space. DE internally treats all variables as floating-point values regardless of their type.

The PSO version employed is the one available in MATLAB15a. The initial swarm was randomly generated, however within the specified – problem-dependent − bounds. The algorithm chose the new member positions based on Equation (17). The inertia term $\omega\epsilon[0.1,1.1]$ was calculated in relation to the number of stalls c:

$$\begin{aligned} &if\ c < 2,\ \ \omega_{i+1} = 2 \cdot \omega_i \\ &elseif\ \ c > 5,\ \ \omega_{i+1} = \frac{\omega_i}{2} \end{aligned} \tag{21}$$

In case the objective function does not improve between two consecutive iterations the neighbourhood size Nh was changed according to:

$$Nh_{i+1} = \min(Nh_i + Nh_{min}, N) \tag{22}$$

where $Nh_{min} = 0.25$ is the minimum number of particles; and $N = 100$, the total number of particles. The maximum number of iterations was $N_{iter} = 160$. The parameters φ_1 and φ_2 were equal, $\varphi_1 = \varphi_2 = 1.49$.

The Artificial Bee Colony algorithm version (I-ABC) used is described in [56]. The total number of employed bees was $N = 100$ and the maximum number of iterations $N_{iter} = 160$. The greedy selection mechanism was employed as the selection operator. The upper bound of the acceleration coefficient was $\Phi_2 = 1$.

The list of benchmark functions employed to compare FOA, c-FOA, DE, PSO and ABC is found in Table 5. In all cases, the number of parameters is $m = 20$. Each optimisation problem was solved for 30 repetitions for each optimisation algorithm. The mean values and standard deviation are summarised in Table 6. In all cases c-FOA and PSO achieved the best performance. As observed from the results, there are cases – *F9* and *F10* – in which c-FOA performs better than the rest and cases in which PSO – *F5* and *F7* – does. The output of Kruskal-Wallis test – probability *P* – and the corresponding box plots for PSO and c-FOA optimisation results for functions *F5*, *F7*, *F9* and *F10* are illustrated in Figure 9.

Table 5. Mathematical benchmark functions employed for the comparison

No	**Description**	*m*	$[x_{imin}, x_{imax}]$	f(**x***)
F1	$f(x) = -0.1 \cdot \sum_{i=1}^{m} cos(5 \cdot \pi \cdot x_i) + \sum_{i=1}^{m} x_i^2$	20	$[-1, 1]$	−2
F2	$f(x) = \sum_{i=1}^{m} x_i^6 \cdot \left(\sin\left(\frac{1}{x_i}\right) + 2\right)$	20	$[-10, 10]$	0
F3	$f(x) = \sum_{i=1}^{m} \frac{x_i^2}{40000} - \prod_{i=1}^{m} \left(\frac{x_i}{\sqrt{i}}\right) + 1$	20	$[-100, 100]$	0
F4	$f(x) = \sum_{i=1}^{m} \lvert x_i \rvert^{i+1}$	20	$[-1, 1]$	0
F5	$f(x) = \sum_{i=1}^{m} (x_i^2 - i)^2$	20	$[-500, 500]$	0
F6	$f(x) = \sum_{i=1}^{m} i \cdot x_i^4 + \eta, \eta \in [0,1]$ *η random number from uniform distribution*	20	$[-1.28, 1.28]$	0

F7	$\sum_{i=1}^{m} \lvert x_i^5 - 3 \cdot x_i^4 + 4 \cdot x_i^3 + 2 \cdot x_i^2 - 10 \cdot x_i - 4 \rvert$	20	$[-10, 10]$	0
F8	$f(x) = \sum_{i=1}^{m} \lvert x_i \rvert$	20	$[-100, 100]$	0
F9	$f(x) = 1 + \sqrt{10000 \cdot \sum_{i=1}^{m} \lvert x_i \rvert}$	20	$[-10,10]$	0
F10	$f(x) = 0.5 + \sum_{i=1}^{m} (x_i^4 - 16 \cdot x_i^2 + 5 \cdot x_i)$	20	$[-5,5]$	≈–783

Table 6. Optimisation benchmark results: Mean best value (Mean) and standard deviation (Std) obtained using Differential Evolution (DE), Artificial Bee Colony (ABC), Particle Swarm Optimisation (PSO), Fruit Fly Optimisation Algorithm (FOA) and contrast-based Fruit Fly Optimisation Algorithm (c-FOA). The comparison is made on the basis of a maximum number of function evaluations $N_{fun} = 16000$

Fun	DE		ABC		PSO		FOA		c-FOA	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F1	1.36	0.30	–1.19	0.16	–1.89	0.11	1.36	0.08	–1.89	0.11
F2	0.15	0.06	$1.60 \cdot 10^{-3}$	$1.20 \cdot 10^{-3}$	$1.42 \cdot 10^{-8}$	$2.15 \cdot 10^{-8}$	$7.76 \cdot 10^{-7}$	$4.28 \cdot 10^{-7}$	$4.06 \cdot 10^{-23}$	$3.45 \cdot 10^{-24}$
F3	4.93	0.53	1.10	0.02	$1.45 \cdot 10^{-2}$	$1.7 \cdot 10^{-2}$	$4.83 \cdot 10^{-2}$	$3.71 \cdot 10^{-2}$	$2.21 \cdot 10^{-2}$	$3.20 \cdot 10^{-2}$
F4	0.03	0.01	$4.33 \cdot 10^{-4}$	$2.96 \cdot 10^{-4}$	$5.42 \cdot 10^{-9}$	$6.81 \cdot 10^{-9}$	$9.38 \cdot 10^{-6}$	$2.28 \cdot 10^{-7}$	$5.00 \cdot 10^{-9}$	$1.17 \cdot 10^{-13}$
F5	$1.94 \cdot 10^{10}$	$6.16 \cdot 10^{9}$	$3.49 \cdot 10^{8}$	$1.58 \cdot 10^{8}$	0.57	1.84	$1.45 \cdot 10^{3}$	$1.24 \cdot 10^{2}$	4.78	2.17
F6	15.94	1.64	**7.74**	0.46	6.40	0.53	6.21	0.55	6.21	0.70
F7	$1.69 \cdot 10^{4}$	$6.33 \cdot 10^{3}$	223.01	106.84	0.02	0.03	115.40	11.84	21.30	4.90
F8	443.72	31.40	32.00	3.93	$6.50 \cdot 10^{-3}$	$4.10 \cdot 10^{-3}$	0.21	$2.00 \cdot 10^{-3}$	$2.00 \cdot 10^{-3}$	$2.01 \cdot 10^{-3}$
F9	3.57	1.18	4.77	2.30	3.23	2.91	47.04	0.20	1.23	0.13
F10	$-3.79 \cdot 10^{2}$	$4.39 \cdot 10^{1}$	$-5.51 \cdot 10^{2}$	$1.79 \cdot 10^{2}$	$-5.35 \cdot 10^{2}$	$2.20 \cdot 10^{2}$	–238.33	14.77	–718.17	22.45

a	b

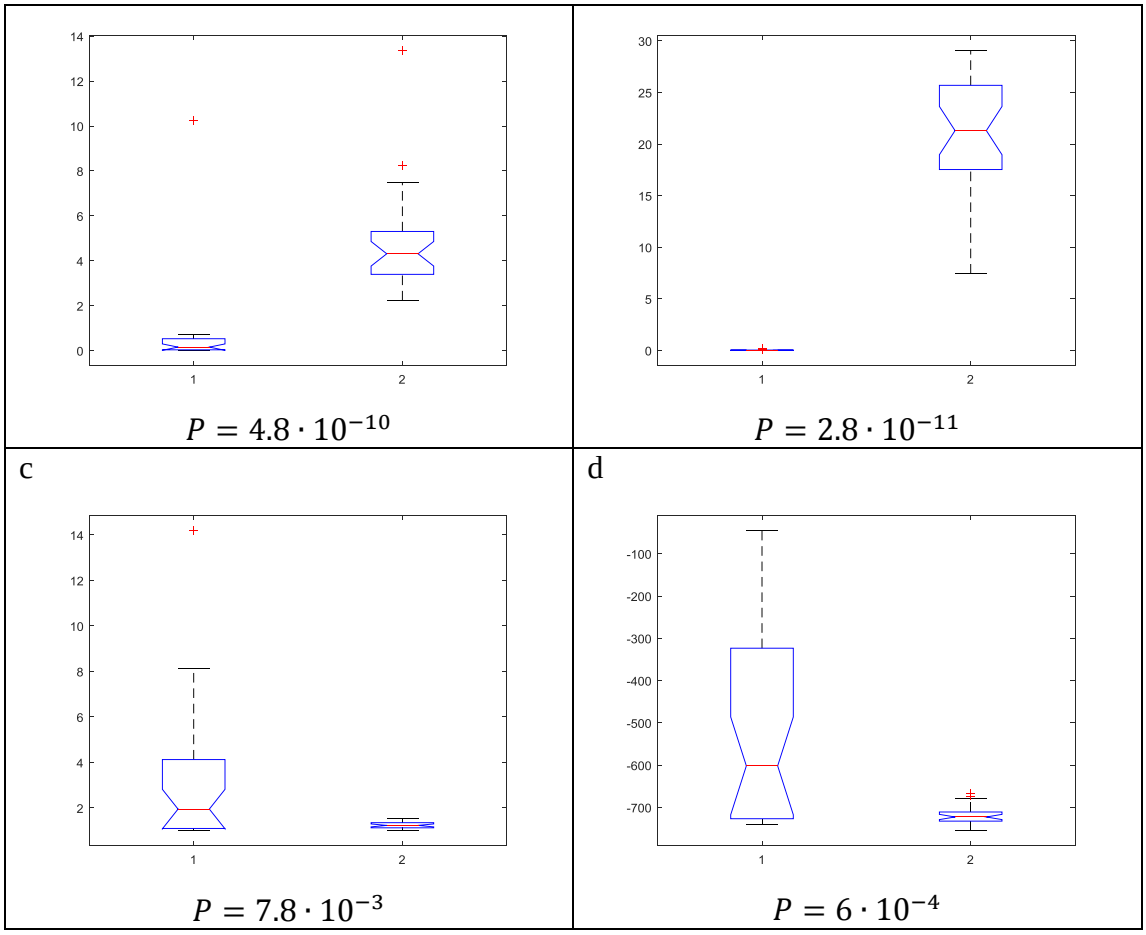

Figure 9. Kruskal-Wallis test output and box plots for the optimisation results obtained using PSO ("1") and c-FOA ("2") for functions a) F5, b) F7, c) F9 and d) F10.

4. Optimised quadratic parameter varying suspension structure

In this section the suspension design problem is described and formulated. As it will be shown the problem involves the iterative solution of a set of nonlinear and coupled differential equations. From an optimisation point of view this problem is classified as highly nonlinear, multi-objective with conflicting requirements and moderately computationally expensive to solve.

4.1 The quarter-car model

This study considers only the vertical vehicle oscillations. Although it is possible to use full-car or half-car models that can also describe the roll and pitch motions, the quarter-car-model is used chiefly because it is simple. Furthermore, the international standard ISO 2631 which is used for objectively evaluating ride quality does not take into account the impact of roll and pitch motions. In Figure 10 the quarter car model with semi-active suspension is shown. Wheel and axle (un-sprung mass m_2) are connected to the car body (sprung mass m_1) through a passive spring k_1 and a nonlinear adaptive damper c_s+c_{adapt}. The tyre is modelled as spring k_2. The road

disturbance is represented by z_0. The equations of motion of the vehicle are the following:

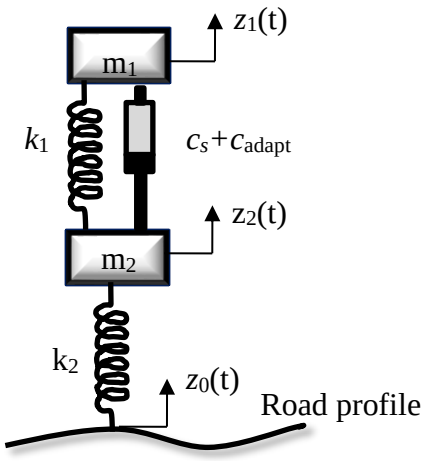

Figure 10**.** Quarter car model with semi-active suspension

$$\begin{aligned} m_1 \cdot \ddot{z}_1 + f_{act} + k_1 \cdot (z_1 - z_2) &= 0 \\ m_2 \cdot \ddot{z}_2 - f_{act} - k_1 \cdot (z_1 - z_2) + k_2 \cdot (z_2 - z_0) &= 0 \end{aligned} \tag{23}$$

where z_1, $\ddot{z}_1$ are the displacement and acceleration of the sprung mass respectively. z_2 and $\ddot{z}_2$ are the displacement and acceleration of the un-sprung mass.

In this study, two typical road disturbances are considered and described in Equation (24).

$$z_{0d}(t) = \begin{cases} = A_b \cdot \dfrac{\left(1 - \cos(8 \cdot \pi \cdot t)\right)}{2}, & for\ 0 \le t \le 0.25s \\ 0, & for\ t > 0.25s \end{cases} \tag{24}$$

$$z_{0r}(t) = random\ road\ profile$$

where A_b is the bump's maximum road height value and z_{0r} a filtered white noise signal [32]. z_{0d} and z_{0r} represent discrete (e.g. bump, pothole) and stochastic (e.g. off-road driving) disturbances respectively.

In many design studies the performance limits of the actuator are neglected, although this can have a significant influence to the solution [57]. In this study we include the dynamic performance of the actuator, describing it with a first-order transfer function [58]:

$$\dot{f}_{act} \cdot T_{act} + f_{act} = f_c \tag{25}$$

f_{act}, $\dot{f}_{act}$ are the actuator force and its rate respectively, f_c is the commanded signal and T_{act} describes the so-called control input rate limit of the actuator. Like any mechanical device, the force generated by the actuator is limited. The maximum actuator force, denoted as $f_{act,\mathrm{lim}}$, is included in the actuator model:

$$|f_{act}| \leq f_{act,\lim} \tag{26}$$

The damper force element f_c is described by the nonlinear quadratic parameter varying equation:

$$f_c = \begin{cases} c_s \cdot (z_1 - z_2), if\ \tilde{z}_R < z_{switch} \\ (c_s + c_{adapt}) \cdot (z_1 - z_2), if\ \tilde{z}_R > z_{switch} \end{cases} \tag{27}$$

where the adaptive damping coefficient c_{adapt} follows a second-order equation:

$$c_{adapt} = c_1 + c_2 \cdot |z_1 - z_2| + c_3 \cdot (z_1 - z_2)^2 \tag{28}$$

and $\tilde{z}_R$ is the predicted rattle space distance

$$\tilde{z}_R = (z_1 - z_2) + T_{pred} \cdot (\dot{z}_1 - \dot{z}_2) \tag{29}$$

The main concept behind the proposed structure, described by Equations (23)-(27), is that the passive system is optimal for normal driving conditions, when $\tilde{z}_R < z_{switch}$. However, in case a road disturbance significantly disturbs the system, such that $\tilde{z}_R > z_{switch}$, the damping coefficient is adjusted to dissipate the disturbance as quickly as possible but without damaging the vehicle.

4.1.1 System constraints

The system is constrained by the available rattle space and road holding requirements. The rattle space constraint z_R is described by:

$$|z_1 - z_2| \leq z_{R,\lim} \tag{30}$$

with $z_{R,\lim}$ denoting the maximum allowed rattle space distance. The absolute value denotes that this is independent of the direction of the road disturbance (bump or pot hole). The road holding requirement is described by the tyre deflection constraint:

$$|z_2 - z_r| \leq z_{tlim} \tag{31}$$

where z_{tlim} is the prescribed tyre deflection limit.

4.2 Optimal robust suspension design problem

Robust optimal suspension design seeks to optimise system performance by taking into account expected variations in vehicle parameters. In real life the vehicle's sprung mass m_1 can change quite frequently because it is dependent on the number of passengers. As it is not always possible to estimate the sprung mass, the system is designed to be robust with respect to mass variations Δm_1.

For the optimal robust suspension design problem we consider the following parameters:

$$\begin{gathered} m_1 = 289\ \text{kg} \\ \Delta m_1 = \pm 30\ \text{kg} \\ m_2 = 59\ \text{kg} \\ c_s = 1000\ \text{N} \cdot \text{s/m} \\ k_1 = 190000\ \frac{\text{N}}{\text{m}} \\ k_2 = 16912\ \frac{\text{N}}{\text{m}} \end{gathered} \tag{32}$$

system constraints:

$$\begin{gathered} z_{Rlim} = 0.08\ \text{m} \\ z_{tlim} = 0.04\ \text{m} \end{gathered} \tag{33}$$

and actuator performance characteristics:

$$\begin{gathered} T_{act} = 0.04\ \text{s} \\ f_{actlim} = 1250\ \text{N} \end{gathered} \tag{34}$$

The switching parameters were chosen based on [34]:

$$\begin{gathered} T_{pred} = 0 \\ z_{switch} = 0.04\ \text{m} \end{gathered} \tag{35}$$

The objective function is expressed as the minimization of the car body acceleration:

$$|\ddot{z}_1| = min \tag{36}$$

while fulfilling the system constraints described in Equations (26)-(27). The main design parameters are the coefficients c_1, c_2 and c_3.

Obviously, there is no analytical solution for the above described problem. One way to find the optimal solution is by trialling out all possible parameter combinations [c_1, c_2, c_3]. However, the simulation time for solving the set of equations (23)-(35) is approximately 8 s, using a laptop equipped with an Intel i5-6200 processor and 8 GB RAM. The computational cost, if all combinations would be trialled out, would be prohibitive therefore efficient search strategies based on optimisation algorithms are required. Based on the results of Section 3, GA, PSO and *c-FOA* were selected for the solution of the optimisation problem. The numerical results are discussed in the following section.

5. Numerical results and discussion

5.1 Optimised suspension using Genetic Algorithm

The suspension design optimisation problem was solved using GA for thirty independent repetitions. The design space was $c_1 \epsilon [-50,50]$, $c_2 \epsilon [-50,50]$ and $c_3 \epsilon [-50,50]$. The same set of GA parameters as in Section 3.1 were used. A statistical evaluation of the optimal set of coefficients is presented in Table 7. The standard deviation is relatively high with respect to the mean values, indicating that the algorithm gets trapped at different local minima. The best result was for $c_1 = 39.40$, $c_2 = -22.04$ and $c_3 = 0.97$. A characteristic plot of the road disturbance and the resulting sprung mass acceleration $\ddot{z}_1$ and rattle space distance z_R is given in Figure 11. The maximum sprung mass acceleration is max($\ddot{z}_1$)=15m/s^2 while the maximum rattle space distance max(z_R)=0.08m.

Table 7. Statistical evaluation of optimal coefficients c_1, c_2 and c_3 obtained using Genetic Algorithm

Optimised damper values using GA	c_1	c_2	c_3
Mean value	5.1	2.4	15.5
Standard deviation	7.5	11.8	31.1

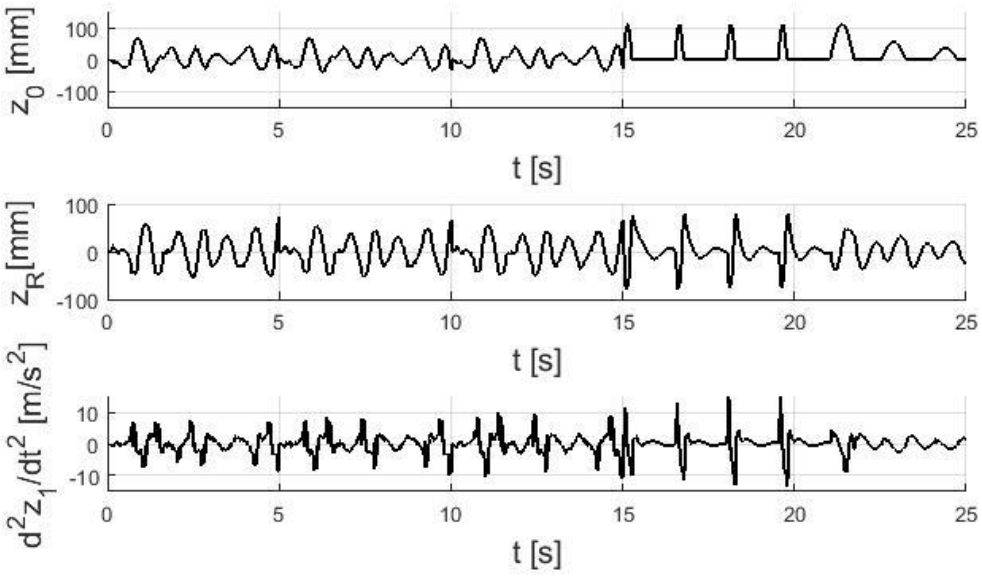

Figure 11**.** Results: Optimised semi-active suspension using *GA*

5.2 Optimised suspension using Particle Swarm Optimisation

PSO was also employed repetitively for the solution of the suspension design problem. The same parameters as in Section 3.2 were used. The design space – for GA – was defined as: $c_1 \epsilon [-50,50]$, $c_2 \epsilon [-50,50]$ and $c_3 \epsilon [-50,50]$. A statistical

evaluation of the obtained optimised coefficients is presented in Table 8. As observed the standard deviation is quite high with respect to the mean values, indicating that the algorithm gets trapped in different local minima. The best result is obtained for $c_1 = 31.37$, $c_2 = -10.86$ and $c_3 = -2.32$. A characteristic plot of the road disturbance and the resulting sprung mass acceleration $\ddot{z}_1$ and rattle space distance z_R is given in Figure 12. The maximum sprung mass acceleration is $\max(\ddot{z}_1)$=15m/s^2 while the maximum rattle space distance $\max(z_R)$=0.08m. GA and PSO perform similar.

Table 8. Statistical evaluation of optimal coefficients c_1, c_2 and c_3 obtained using Particle Swarm Optimisation

Optimised damper values using *PSO*	c_1	c_2	c_3
Mean value	26.00	–19.16	6.95
Standard deviation	26.29	22.24	8.08

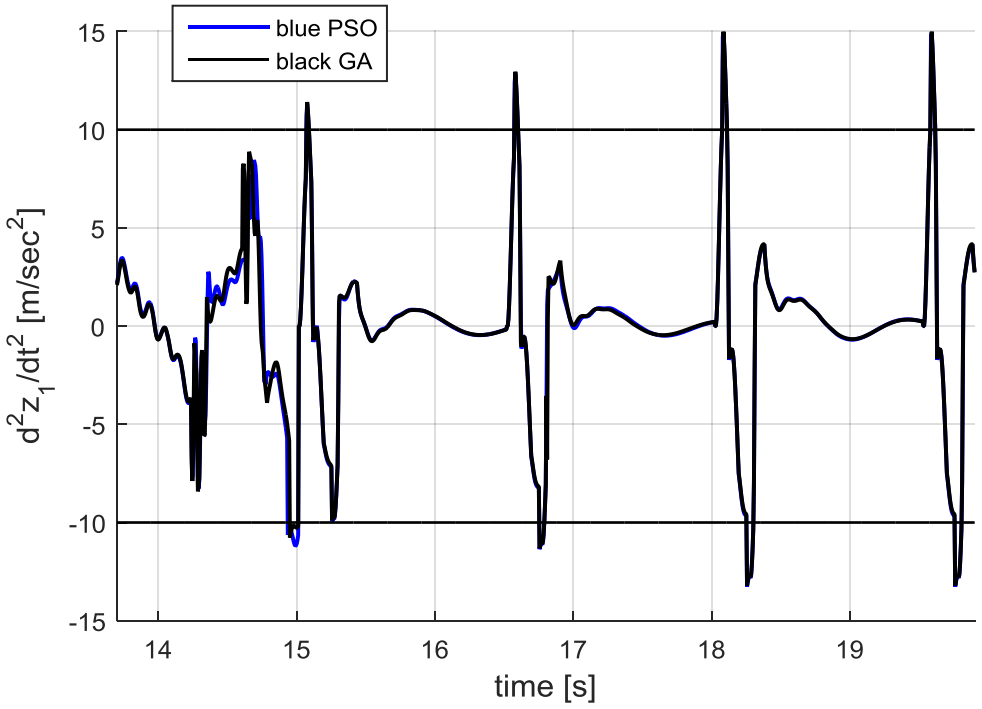

Figure 13**.** Results: Peak accelerations comparison between *GA* and *PSO*

5.3 Optimised suspension using c-FOA

c-FOA was also employed thirty times with $c_1 \epsilon [-50,50]$, $c_2 \epsilon [-50,50]$ and $c_3 \epsilon [-50,50]$. The average values and standard deviation of the design parameters are shown in Table 9. The standard deviation is smaller compared to those achieved using GA and PSO. The best result was for $c_1 = 23.50$, $c_2 = 3.51$ and $c_3 = -7.78$. In Figure 14 the resulting accelerations for the GA and *c-FOA* suspension designs are

illustrated. The results are shown for the time interval where the vehicle encounters the discrete disturbances. As observed the peak accelerations are significantly reduced with *c-FOA*. The maximum sprung mass acceleration is max($\ddot{z}_1$)=13.4m/s^2 while the maximum rattle space distance max(z_R)=0.08m.

Table 9. Statistical evaluation of optimal coefficients c_1, c_2 and c_3 obtained using *c-FOA*

Optimised damper values using *c-FOA*	c_1	c_2	c_3
Mean value	27.50	–4.42	–4.62
Standard deviation	3.42	8.01	3.40

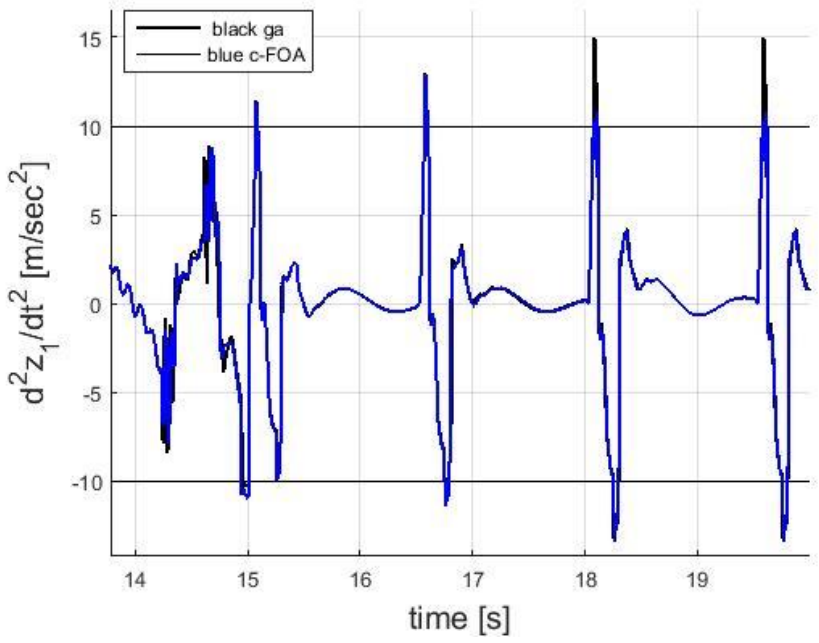

Figure 14**.** Results: Peak accelerations comparison between Genetic Algorithm and c-FOA

5.4 Discussion

The average convergence histories for the three algorithms are presented in Figure 15. The peaks in the convergence history of Figure 15c are because c-FOA searches also the vicinity of design solutions that have very high objective function values (for example at the boundary of the barrier functions). In Figure 15d the lower envelope of *c-FOA's* average convergence history is plotted.

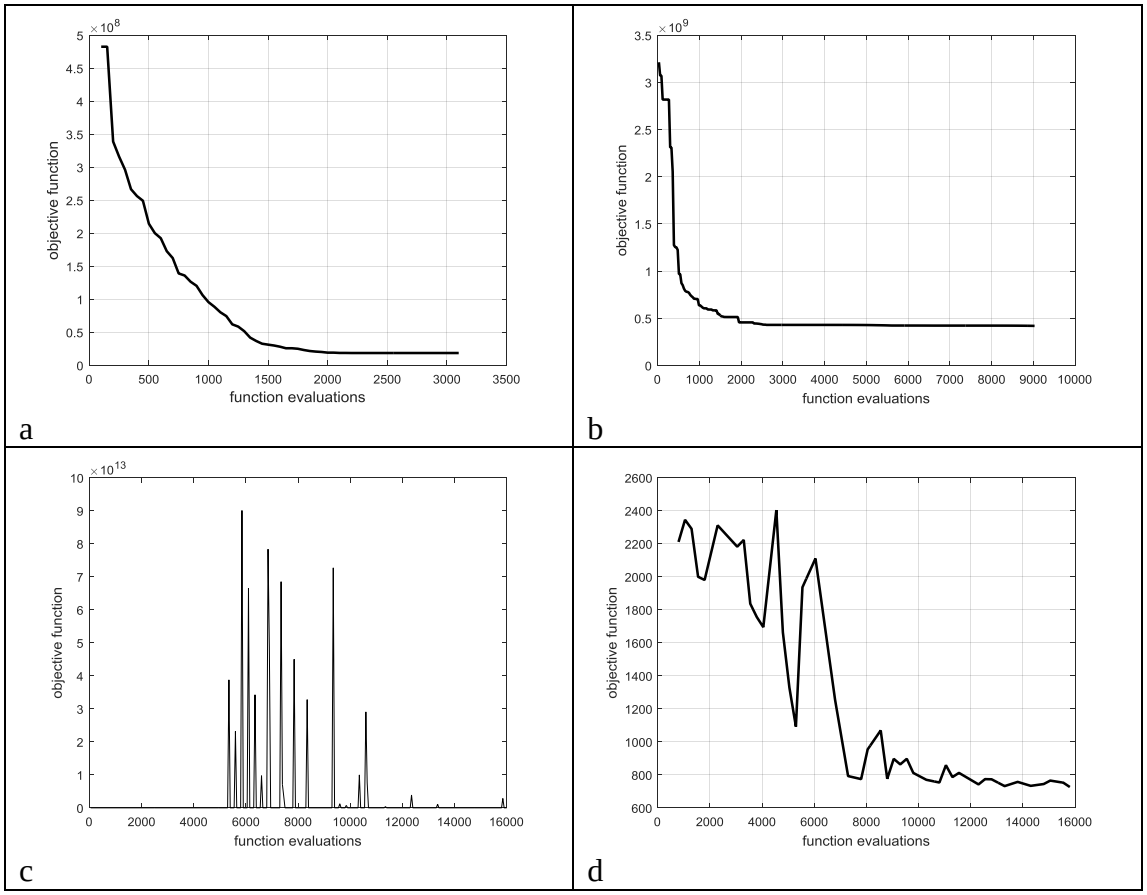

Figure 15**.** Average convergence history for a) Genetic Algorithm, b) Particle Swarm Optimisation, c) c-FOA and d) lower envelope for c-FOA

The numerical results obtained using the passive suspension, and those optimized with GA, PSO and c-FOA, are summarised in Table 10. It is clear that the passive suspension system does not meet the rattle space requirement $|z_R| \leq 0.08$. All optimised suspension designs are meeting the rattle space requirement without degrading the road holding performance expressed by z_t. On the other hand it is observed that the sprung mass acceleration $\max(\ddot{z}_1)$ increases by 25% (from 12 to 15 m/s^2) and 11.3% (from 12 to 13.4 m/s^2) with GA/PSO and c-FOA algorithms, respectively. The increased acceleration values are due to the increased damping required to prevent overcoming the suspension limits.

Table 10. Comparison of results for three different suspension systems – Passive and optimised using Genetic Algorithm, Particle Swarm Optimisation and c-FOA

	$\ddot{z}_1$ / m/s^2	z_R / m	z_t / m
Passive suspension	12	1.14×10^{-1}	2.8×10^{-2}
Optimised suspension using GA	15	8×10^{-2}	2.8×10^{-2}

Optimised suspension using PSO	15	8×10^{-2}	2.8×10^{-2}
Optimised suspension using c-FOA	13.4	8×10^{-2}	2.8×10^{-2}

In order to obtain better insight of the optimised result, we plot the effective damping coefficient versus rattle space velocity $\dot{z}_R$, see Figure 16. The effective damping coefficient c_{eff} is defined as:

$$c_{eff} = \begin{cases} \dfrac{c_{adapt}}{c_s}, if\ \tilde{z}_R > z_{switch}\ and\ f_c \leq f_{actlim} \\ \dfrac{f_{act,lim}/\dot{z}_R}{c_s}, if\ \tilde{z}_R > z_{switch}\ and\ f_c > f_{actlim} \end{cases} \tag{37}$$

and is a metric that shows how much the passive damping coefficient needs to increase.

The graph indicates that damping should reach its peak for low rattle space velocities, $-0.05 \leq \dot{z}_R \leq 0.05$ m/s. For intermediate velocities $0.05 \leq |\dot{z}_R| \leq 0.3$ m/s damping should decrease exponentially, while for $|\dot{z}_R| > 0.3$ m/s it should approach its steady value. It is highlighted that these results hold only when $\tilde{z}_R > z_{switch}$, otherwise the system retains its passive behaviour.

An intuitive explanation of the result is that the optimiser suggests scaling up considerably the damping coefficient in the range where very low damping forces are usually exerted and then decrease it as the relative velocity increases. Of course, this should only happen when the road disturbance excites the system significantly, $\tilde{z}_R > z_{switch}$. An experimental investigation using a prototype magnetorheological damper showed that it is possible to achieve the desired scaling in damping. The results are presented in Figure 17.

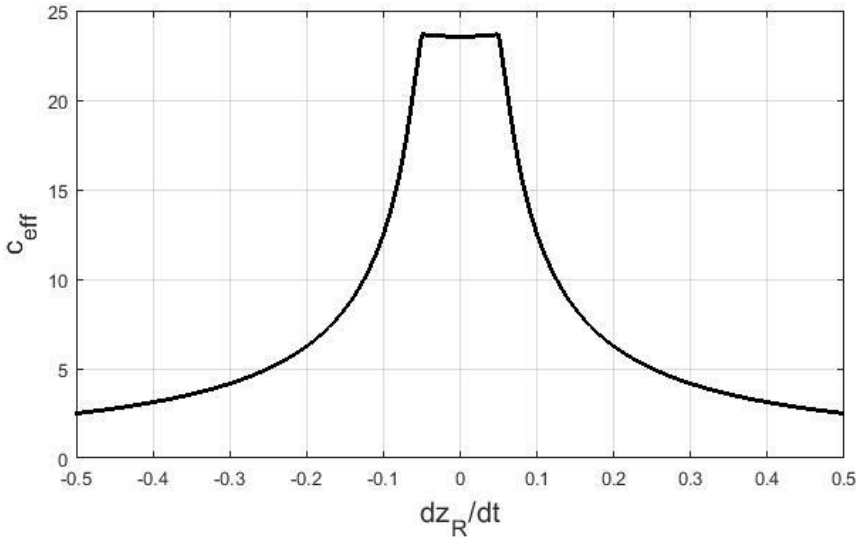

Figure 16**.** Scaling damping coefficient c_{eff} versus rattle space velocity $\boldsymbol{\frac{dz_R}{dt}}$

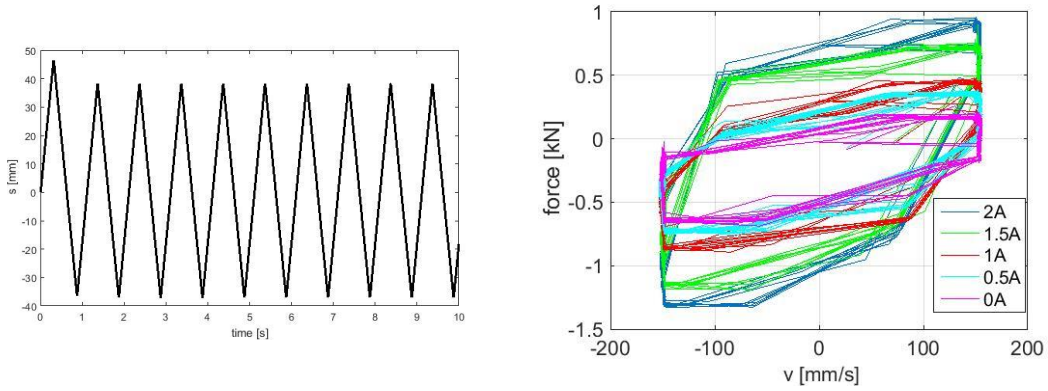

Figure 17**.** Experimental results obtained from magnetorheological damper excited by a toothsaw input signal for different current values 0-2 A

6. Summary and conclusions

Every year thousands of motorists damage their vehicles by hitting severe road anomalies such as large potholes. Poor road conditions in combination with the use of low-profile tyres have increased the scale of the problem and made pothole damage mitigation a priority for a number of car manufacturers. Passive suspension systems cannot meet the competing objectives of comfort, road holding and pothole damage mitigation. In this paper, a clipped quadratic parameter varying suspension system is proposed for this purpose. The following conclusions are drawn:

1. There are no design rules that can help an engineer to design a nonlinear suspension system based on a quadratic parameter varying damper. Standard global optimisation algorithms like Genetic Algorithm and Particle Swarm Optimisation could not find the optimised solution. Both Genetic Algorithm and Particle Swarm Optimisation gave solutions located far from the optimum design values.

2. A new Fruit Fly Optimisation Algorithm – based on a recent study on how well fruit fly's tiny brain finds food – was developed. The standard Fruit Fly Optimisation Algorithm was enhanced by introducing the delay and visual feature detection phases that characterise a fruit fly's food search strategy. The proposed c-FOA is a Swarm Intelligence heuristic, with unique − compared to other heuristics − food search strategies that have been developed through evolution.

3. The new optimisation algorithm, named *c-FOA*, was compared to the Genetic Algorithm, Simulated Annealing, Particle Swarm Optimisation, Differential Evolution, Artificial Bee Colony and the original Fruit Fly Optimisation Algorithm. In total 14 benchmark functions were employed, commonly used for this purpose in the literature. Both low and high dimensional studies were conducted.

4. The comparison between the optimisation algorithms in the low dimensional benchmark tests revealed that the Genetic Algorithm and Simulated Annealing performed similarly well and c-FOA slightly more robustly. Particle Swarm Optimisation and c-FOA performed better than Differential Evolution, Artificial Bee Colony and the original Fruit Fly Optimisation Algorithm in the high dimensional

benchmark tests. In a limited number of benchmark tests Particle Swarm Optimisation performed better while in another limited number of benchmark tests *c-FOA* did. In particular it seems that Particle Swarm Optimisation performs better when the objective function value landscape is flat, while c-FOA performs better when it is steep. The performance was evaluated using simple statistical means and using non-parametric tests like the Kruskal-Wallis test.

5. In the suspension design problem *c-FOA* performed better than Genetic Algorithm and Particle Swarm Optimisation. Both the best result achieved as well as the average optimised results were better. A comparison between the convergence histories reveals that the Genetic Algorithm and Particle Swarm Optimisation become stuck in local minima.

6. The resulting optimal design suggests that advanced suspension systems need to increase damping at low rattle space velocities, when the road disturbance excitation is significant. At higher velocities damping should decrease. Preliminary tests using a prototype magnetorheological damper showed that both design recommendations are possible to achieve.

Future research plans include the design investigation of a direct current controller that will optimize the transient performance of the magnetorheological damper and applying *c-FOA* to other types of optimisation problems.

Acknowledgements

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. http://www.telegraph.co.uk/news/uknews/road-and-rail-transport/11368326/Potholes-cause-damage-to-cars-every-11-mins-figures-show.html
2. http://news.bbc.co.uk/1/hi/world/europe/8556915.stm
3. http://www.at.ford.com/news/cn/Pages/Mitigating%20Pothole%20Damage%20is%20Priority%20for%20Ford.aspx
4. Tseng, H. and Hrovat, D. (2015). State of the art survey: active and semi-active suspension control. Vehicle System Dynamics, 53(7), pp.1034-1062.
5. Kasemi, B., Muthalif, A., Rashid, M. and Fathima, S. (2012). Fuzzy-PID Controller for Semi-Active Vibration Control Using Magnetorheological Fluid Damper. Procedia Engineering, 41, pp.1221-1227.
6. Chiou, J., Tsai, S. and Liu, M. (2012). A PSO-based adaptive fuzzy PID-controllers. Simulation Modelling Practice and Theory, 26, pp.49-59.
7. Wang, W., Song, Y., Xue, Y., Jin, H., Hou, J. and Zhao, M. (2015). An optimal vibration control strategy for a vehicle's active suspension based on improved cultural algorithm. Applied Soft Computing, 28, pp.167-174.
8. Tandel, A., Deshpande, A., Deshmukh, S. and Jagtap, K. (2014). Modeling, Analysis and PID Controller Implementation on Double Wishbone Suspension Using SimMechanics and Simulink. Procedia Engineering, 97, pp.1274-1281.

9. Gosiewski, Z. and Mystkowski, A. (2008). Robust control of active magnetic suspension: Analytical and experimental results. Mechanical Systems and Signal Processing, 22(6), pp.1297-1303.
10. Savaresi, S., Poussot-Vassal, C., Spelta, C., Sename, O., & Dugard, L. (2010). Semiactive suspension control design for vehicles. Elsevier.
11. Koch, G. (2011). Adaptive control of mechatronic vehicle suspension systems. Ph.D. Thesis, Technische Universität München
12. Sharp, R. S., & Peng, H. (2011). Vehicle dynamics applications of optimal control theory. Vehicle System Dynamics, 49, 1073–1111.
13. Unger, A., Schimmack, F., Lohmann, B. and Schwarz, R. (2013). Application of LQ-based semi-active suspension control in a vehicle. Control Engineering Practice, 21(12), pp.1841-1850.
14. Brezas, P., Smith, M. and Hoult, W. (2015). A clipped-optimal control algorithm for semi-active vehicle suspensions: Theory and experimental evaluation. Automatica, 53, pp.188-194.
15. Brezas, P. and Smith, M. (2014). Linear Quadratic Optimal and Risk-Sensitive Control for Vehicle Active Suspensions. IEEE Transactions on Control Systems Technology, 22(2), pp.543-556.
16. Poussot-Vassal, C., Spelta, C., Sename, O., Savaresi, S. and Dugard, L. (2012). Survey and performance evaluation on some automotive semi-active suspension control methods: A comparative study on a single-corner model. Annual Reviews in Control, 36(1), pp.148-160.
17. Crews, J., Mattson, M. and Buckner, G. (2011). Multi-objective control optimization for semi-active vehicle suspensions. Journal of Sound and Vibration, 330(23), pp.5502-5516.
18. Hong, K., Sohn, H. and Hedrick, J. (2002). Modified Skyhook Control of Semi-Active Suspensions: A New Model, Gain Scheduling, and Hardware-in-the-Loop Tuning. Journal of Dynamic Systems, Measurement, and Control, 124(1), p.158.
19. Gopala Rao, L. and Narayanan, S. (2009). Sky-hook control of nonlinear quarter car model traversing rough road matching performance of LQR control. Journal of Sound and Vibration, 323(3-5), pp.515-529.
20. Priyandoko, G., Mailah, M. and Jamaluddin, H. (2009). Vehicle active suspension system using skyhook adaptive neuro active force control. Mechanical Systems and Signal Processing, 23(3), pp.855-868.
21. Karnopp, D., Crosby, M. and Harwood, R. (1974). Vibration Control Using Semi-Active Force Generators. Journal of Engineering for Industry, 96(2), p.619.
22. Hong, K., Sohn, H. and Hedrick, J. (2002). Modified Skyhook Control of Semi-Active Suspensions: A New Model, Gain Scheduling, and Hardware-in-the-Loop Tuning. Journal of Dynamic Systems, Measurement, and Control, 124(1), p.158.
23. Miller, L.R. and Nobles, C.M. (1990). Methods for eliminating jerk and noise in semi active suspensions, SAE 1990 Transactions, Sec. 2, pp.943-951, Paper No. 902284
24. Tong, R.T. (2001). Ride control: a two states design for heavy vehicle suspension, PhD Dissertation, University of Illinois at Chicago.
25. Nguyen, M., da Silva, J., Sename, O. and Dugard, L. (2015). A state feedback input constrained control design for a 4-semi-active damper suspension system: a quasi-LPV approach. IFAC-PapersOnLine, 48(14), pp.259-264.
26. Poussot-Vassal, C., Sename, O., Dugard, L., Gáspár, P., Szabó, Z. and Bokor, J. (2008). A new semi-active suspension control strategy through LPV technique. Control Engineering Practice, 16(12), pp.1519-1534.

27. Li, P., Lam, J. and Cheung, K. (2014). Velocity-dependent multi-objective control of vehicle suspension with preview measurements. Mechatronics, 24(5), pp.464-475.
28. Rotondo, D., Puig, V., Nejjari, F. and Witczak, M. (2015). Automated generation and comparison of Takagi–Sugeno and polytopic quasi-LPV models. Fuzzy Sets and Systems, 277, pp.44-64.
29. Rotondo, D., Nejjari, F. and Puig, V. (2014). Robust state-feedback control of uncertain LPV systems: An LMI-based approach. Journal of the Franklin Institute, 351(5), pp.2781-2803.
30. Eski, İ. and Yıldırım, Ş. (2009). Vibration control of vehicle active suspension system using a new robust neural network control system. Simulation Modelling Practice and Theory, 17(5), pp.778-793.
31. Yildirim, Ş. (2004). Vibration control of suspension systems using a proposed neural network. Journal of Sound and Vibration, 277(4-5), pp.1059-1069.
32. Kanarachos, S. (2012). Intelligent semi-active vehicle suspension systems using neural networks. International Journal of Vehicle Systems Modelling and Testing, 7(2), p.135.
33. Qin, Y., Dong, M., Langari, R., Gu, L. and Guan, J. (2015). Adaptive Hybrid Control of Vehicle Semiactive Suspension Based on Road Profile Estimation. Shock and Vibration, 2015, pp.1-13.
34. Kanarachos, S. and Kanarachos, A. (2015). Intelligent road adaptive suspension system design using an experts' based hybrid genetic algorithm. Expert Systems with Applications, 42(21), pp.8232-8242.
35. Pedro, J., Dangor, M., Dahunsi, O. and Ali, M. (2014). Intelligent feedback linearization control of nonlinear electrohydraulic suspension systems using particle swarm optimization. Applied Soft Computing, 24, pp.50-62.
36. Lin, J. and Lian, R. (2013). Design of a grey-prediction self-organizing fuzzy controller for active suspension systems. Applied Soft Computing, 13(10), pp.4162-4173.
37. Mahmoodabadi, M., Safaie, A., Bagheri, A. and Nariman-zadeh, N. (2013). A novel combination of Particle Swarm Optimization and Genetic Algorithm for Pareto optimal design of a five-degree of freedom vehicle vibration model. Applied Soft Computing, 13(5), pp.2577-2591.
38. Pan, W. (2013). Using modified fruit fly optimisation algorithm to perform the function test and case studies. Connection Science, 25(2-3), pp.151-160.
39. Yuan, X., Liu, Y., Xiang, Y. and Yan, X. (2015). Parameter identification of BIPT system using chaotic-enhanced fruit fly optimization algorithm. Applied Mathematics and Computation, 268, pp.1267-1281.
40. Mousavi, S., Alikar, N., Niaki, S. and Bahreininejad, A. (2015). Optimizing a location allocation-inventory problem in a two-echelon supply chain network: A modified fruit fly optimization algorithm. Computers & Industrial Engineering, 87, pp.543-560.
41. Wang, L., Shi, Y. and Liu, S. (2015). An improved fruit fly optimization algorithm and its application to joint replenishment problems. Expert Systems with Applications, 42(9), pp.4310-4323.
42. Kumar, M. and Rawat, T. (2015). Optimal design of FIR fractional order differentiator using cuckoo search algorithm. Expert Systems with Applications, 42(7), pp.3433-3449.
43. https://en.wikipedia.org/wiki/Drosophila, accessed on 16.01.2017

44. http://www.flyfoa.com/2014/08/introduction-to-fruit-fly-optimization.html, accessed on 16.01.2017
45. Pan, W. (2012). A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example. Knowledge-Based Systems, 26, pp.69-74.
46. van Breugel, F. and Dickinson, M. (2014). Plume-Tracking Behavior of Flying Drosophila Emerges from a Set of Distinct Sensory-Motor Reflexes. Current Biology, 24(3), pp.274-286
47. http://www.futurity.org/fruit-flys-tiny-brain-finds-food-well/, accessed on 16.01.2017
48. Banks, Alec, Jonathan Vincent, and Chukwudi Anyakoha. "A Review Of Particle Swarm Optimization. Part I: Background And Development". Natural Computing 6.4 (2007): 467-484
49. http://mf.erciyes.edu.tr/abc/publ.htm, accessed on 16.01.2017
50. Price, Kenneth V, Rainer M Storn, and Jouni A Lampinen. Differential Evolution. 1st ed. Berlin: Springer, 2005.
51. Silveira, C. L., Mazutti, M. A., Salau, N. P. G. Solid-state fermentation process model reparametrization procedure for parameters estimation using particle swarm optimization. Journal of Chemical Technology and Biotechnology, v. 91, n. 3, p. 762-768, 2016.
52. He, Q., Wang, L., Liu, B. Parameter estimation for chaotic systems by particle swarm optimization. Chaos, Solitons & Fractals, v. 34, n. 2, p. 654-661, 2007.
53. Kanarachos, A., Koulocheris, D. and Vrazopoulos, H. (2003). Evolutionary algorithms with deterministic mutation operators used for the optimization of the trajectory of a four-bar mechanism. Mathematics and Computers in Simulation, 63(6), pp.483-492.
54. https://www.mathworks.com/matlabcentral/answers/uploaded_files/20100/Fruit%20Fly%20Optimization%20Algorithm_Second%20Edition.pdf, accessed on 16.01.2017
55. http://www1.icsi.berkeley.edu/~storn/code.html#matl, accessed on 16.01.2017
56. L. Guoqiang, P. Niu, and X. Xiao. Development and Investigation of Efficient Artificial Bee Colony Algorithm for Numerical Function Optimization, Applied Soft Computing, vol. 12, no. 1, pp. 320-332, 2012.
57. Isermann R. (2003). Mechatronic systems: fundamentals, New York: Springer, (Chapter 12)
58. Nguyen, Q. and Choi, S. (2008). Optimal design of a vehicle magnetorheological damper considering the damping force and dynamic range. Smart Materials and Structures, 18(1), p.015013.