Multiple Crack Detection in Beam-Like Structures Using a Novel Particle Swarm Optimization Approach

Flaviu-Catalin Florea[4][a], Horea Grebla[1][b], Gilbert-Rainer Gillich[2][c], Bogdan Nicușor Bindea[5][d] and Catalin V. Rusu[1,3][e]

[1] *Department of Computer-Science, Babes-Bolyai University, Romania*
[2] *Department of Engineering Science, Babes-Bolyai University, Romania*
[3] *Department of Computer Science, Institute of German Studies, Babes-Bolyai University, Romania*
[4] *Department of Computer Science, Aarhus University, Denmark*
[5] *Department of Computer Science, Technical University of Cluj-Napoca, Romania*

Keywords: Damage Detection, Natural Frequencies, Cost Function, Particle Swarm Optimization.

Abstract: This paper presents a method for assessing two cracks in simply supported beams by identifying their locations and severities (depths). Our method is based on applying the Particle Swarm Optimization (PSO) algorithm with the measured natural frequencies for several bending vibration modes of an intact and cracked beam. We are using calculated relative frequency shifts (RFS) for eight vibration modes for all possible damage cases using a mathematical relation deduced in previous researches. We detect changes, calculate the RFSs and then subtract, separately for all modes, the measured RFSs from all calculated RFSs. Considering previously demonstrated applications of PSO for one crack detection, we propose strategies to enable PSO to determine locations in scenarios involving two cracks. Our method is successful in accurately identifying two damage locations and severities.

1 INTRODUCTION

Fault detection in beam-like structures is an important research field as it has real-life implications, from saving time, money and even lives, if cracks are detected in early stages. There are several studies that use non invasive methods to identify the location and depth of one crack in beam like structures. There are however only a few such studies that deal with multiple cracks at the same time. Most related work involving Evolutionary Algorithms focuses on Genetic Algorithms (GA), but in this study our focus is on Particle Swarm Optimization (PSO). To our knowledge, no prior work uses the Equivalent Healthy Beam (EHB) model with Evolutionary Algorithms for damage prediction.

Mohan, Maiti and Maity evaluated PSO and GA in multiple crack detection (Mohan et al., 2013). PSO outperformed GA in accuracy and robustness, consistently providing more reliable predictions. They used the Frequency Response Function (FRF) model on a simple cantilever beam, demonstrating that combining FRF with PSO enhances accuracy. Khatir et al. (Khatir et al., 2017) also compared GA and PSO for detecting cracks in composite beams. PSO proved superior in accuracy, efficiency, and robustness, especially with noise in the modal data, guiding the objective function using the Modal Assurance Criterion.

Greco et al. (Greco et al., 2018) proposed a static method using GA and a closed-form solution based on a rotational spring model for multiple crack detection. This method, implemented in the NetLogo environment, showed effective crack identification under various conditions. Khai and Mehrjoo (Khaji and Mehrjoo, 2014) introduced a new beam element capable of including multiple transverse edge cracks. Using GA, they solved an inverse problem to determine crack specifics, validating their method against 2D finite element analyses and experimental data. Zheng, Liang, Wang and Fan (Zheng et al., 2014) used a Hierarchical Genetic Algorithm (HGA) with the same model as the one used in (Khaji and Mehrjoo, 2014), proving HGA's superiority over simple GA in reducing finite element computations and avoiding premature convergence. Sahu, Kumar, and Parhi developed a hybrid method (CSAGA) combining GA with the

[a] https://orcid.org/0009-0009-2479-6588
[b] https://orcid.org/0000-0002-8529-5797
[c] https://orcid.org/0000-0003-4962-2567
[d] https://orcid.org/0009-0000-7045-2043
[e] https://orcid.org/0000-0002-2056-8440

Florea, F.-C., Grebla, H., Gillich, G.-R., Bindea, B. N. and Rusu, C. V.
Multiple Crack Detection in Beam-Like Structures Using a Novel Particle Swarm Optimization Approach.
DOI: 10.5220/0013131400003890
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 3*, pages 334-342
ISBN: 978-989-758-737-5; ISSN: 2184-433X
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

clonal selection algorithm (Sahu et al., 2018). This method improved accuracy in locating cracks and assessing their severity by examining changes in natural frequencies. Moradi and Kargozarfard (Moradi and Kargozarfard, 2013) introduced the Bees Algorithm for multiple crack detection, using changes in eigenfrequencies and strain energy parameters to optimize a damage index vector. This method effectively identified the number and locations of cracks.

Machine learning approaches, primarily using Artificial Neural Networks (ANN), are less common in crack detection. Maurya, Mishra and Panigrahi demonstrated ANN's effectiveness in structural analysis, using the first three relative natural frequencies to predict crack locations and depths accurately (Maurya et al., 2018). Pop, Tufisi, and Gillich (Pop et al., 2022) optimized an ANN through feedforward backpropagation to pinpoint crack positions in a 1000mm cantilever steel beam with high precision. Their model utilized the first five natural frequencies, highlighting the impact of crack proximity on modal response.

In multiple crack detection for beam-like structures, GAs and the rotational spring model are predominant. However, each study achieves different error percentages based on varying beam properties, making direct comparisons difficult. Most studies report errors close to 2%, 1%, or even 0%.

Nevertheless, we found no study up to date to tackle the problem from the point of view of the Equivalent Healthy Beam model and Particle Swarm Optimization. Hence, we propose a new approach for multiple damage detection.

2 THEORETICAL BACKGROUND

2.1 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a method within evolutionary computation in artificial intelligence (AI), inspired by natural behaviors such as bird flocking and fish schooling. Imagine a flock of birds: when one bird finds a good spot for food, others follow, adjusting their paths. Similarly, in PSO, each "particle" (representing a potential solution) moves through the solution space, learning from its own experiences and those of others. Each particle keeps track of its best position and is influenced by the best position found by the swarm.

PSO success hinges on balancing exploration (scouting new areas) and exploitation (refining known good areas). This balance is managed through key settings:

- Inertia Weight (w): Helps particles move. A higher weight in the early stages encourages broad exploration, while a reduced weight later on helps fine-tune solutions.
- Cognitive Coefficient (c1): Influences a particle based on its past success, encouraging it to revisit or stay near promising solutions.
- Social Coefficient (c2): Influences a particle based on the swarm's best-found solution, promoting a collective approach.

The velocity and position of each particle are updated using:

$$v_{k+1} = w \cdot v_k + c1 \cdot r1 \cdot (\text{pbest}_i - x_i) + c2 \cdot r2) \cdot (\text{gbest} - x_i) \quad (1)$$

$$x_i = x_i + v_{k+1} \quad (2)$$

where v_k and v_{k+1} are the current and next velocities of particle i, w is the inertia parameter, $c1$ is the cognitive parameter, $c2$ is the social parameter, pbest_i is the best position of particle i, gbest is the best position of the swarm, x_i is the current position of particle i, and $r1$ and $r2$ are random numbers between 0 and 1.

Topology, which dictates how particles communicate, is crucial in Particle Swarm Optimization. Within a global topology, every particle communicates with all others, quickly sharing the best solutions. In contrast, a local topology restricts communication to nearby particles, encouraging more independent exploration. This choice affects how swiftly the swarm converges and how well it finds optimal solutions.

We will go into more depth about the steps we took to get our PSO model to correctly anticipate two defects in a cantilever beam in the upcoming chapters. The techniques and experiments, ranging from hyperparameter tweaking to several unique approaches, are covered in the next chapters.

2.2 Relative Frequency Shift for One Crack Scenario

A crack in a beam decreases the rigidity of a specific area, affecting its vibration. This is seen as changes in eigenfrequencies, which are the natural vibration frequencies of the structure. For a thought experiment, imagine a cantilever beam, fixed at one end and free at the other. If one were to push it down on the free end and then release it, the beam will vibrate at a specific frequency. If there is a crack at a point c with depth

d, the beam will be weaker at that point, causing it to bend more under the same pressure.

To model this, we use the Equivalent Healthy Beam (EHB) model (Gillich et al., 2019). This model suggests that a beam with a crack behaves like a thinner, uniformly healthy beam with the same deflection at the free end. According to Castigliano's theorem, both beams store the same energy and oscillate at the same frequency.

The relationship between the deflection δ at the free end and the eigenfrequency f_i is:

$$f_i = \frac{\lambda_i^2}{2\pi}\sqrt{\frac{g}{8\delta}} \tag{3}$$

where λ_i is the eigenvalue for each i vibration mode and g is the gravitational constant. The deflection δ for the healthy beam is:

$$\delta = \frac{pgAL^4}{8EI} \tag{4}$$

where L is the length of the beam, A is the cross-sectional area ($A = \text{width} \times \text{height}$), p is the mass density, E is the Young's modulus, and I is the moment of inertia. The constant 8 comes from the boundary conditions of the beam (Gillich et al., 2021).

Cracks deepen at point c to d, making the free end bend more ($\delta(c,d)$). The healthy beam needs less area ($A(c,d)$) and less bending resistance ($I(c,d)$) to match. Both vibrate the same because they store the same energy, and you calculate the damaged beam's frequency like this:

$$f_i(c,d) = \frac{\lambda_i^2}{2\pi}\sqrt{\frac{g}{8\delta(c,d)}} \tag{5}$$

This leads to:

$$f_i(c,d) = f_i\sqrt{\frac{\delta}{\delta(c,d)}} \tag{6}$$

A crack at $c = 0$, where the curvature is maximum, will cause the greatest increase in deflection and the largest decrease in frequency. In the literature, it's called the Equivalent Healthy Beam (EHB). It's the least stiff among healthy beams used to model a crack of a certain depth. We assign a damage severity coefficient, γ, to the crack, calculated as shown in (Praisach et al., 2013).

$$\gamma(0,d) = \frac{\sqrt{\delta(0,d)} - \sqrt{\delta}}{\sqrt{\delta(0,d)}} \tag{7}$$

As shown in Figure 1, a beam with a crack at location c has higher deflection than a healthy beam but the same deflection as an EHB with the crack at $c = 0$.

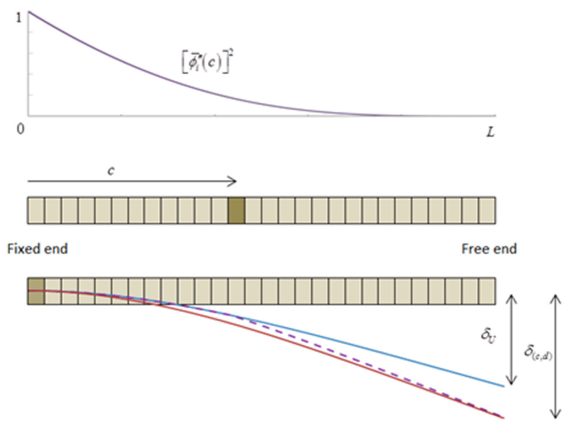

Figure 1: The 2 models of a beam; one with a crack at location c and one with the crack at the fixed end.

The frequency drop Δf_i for a crack at $c = 0$ is:

$$\begin{aligned}\Delta f_i(0,d) &= f_i - f_i(0,d) = f_i\left(1 - \sqrt{\frac{\delta}{\delta(0,d)}}\right) \\ &= f_i\frac{\sqrt{\delta(0,d)} - \sqrt{\delta}}{\sqrt{\delta(0,d)}} = f_i\gamma(0,d)\end{aligned} \tag{8}$$

For a general crack position c:

$$\Delta f_i(c,d) = f_i - fi(c,d) = f_i\frac{\sqrt{\delta(c,d)} - \sqrt{\delta}}{\sqrt{\delta(c,d)}} \tag{9}$$

The damage severity γ can be related to the normalized modal curvature $\left(\bar{\phi}_i''(c)\right)^2$ (Gillich and Praisach, 2014):

$$\begin{aligned}\gamma(c,d) &= \frac{\sqrt{\delta(c,d)} - \sqrt{\delta}}{\sqrt{\delta(c,d)}} = \frac{\sqrt{\delta(0,d)} - \sqrt{\delta}}{\sqrt{\delta(0,d)}}\left(\bar{\phi}_i''(c)\right)^2 \\ &= \gamma(0,d)\left(\bar{\phi}_i''(c)\right)^2\end{aligned} \tag{10}$$

where $\left(\bar{\phi}_i''(c)\right)^2$ is the normalized modal curvature, indicating the local curvature's effect due to bending. For cantilever beams, the normalized curvature is:

$$\begin{aligned}\bar{\phi}_i''(x) = 0.5\,\{&\cos(\lambda_i x) + \cosh(\lambda_i x) \\ &- \frac{\cos(\lambda_i) + \cosh(\lambda_i)}{\sin(\lambda_i) + \sinh(\lambda_i)}(\sin(\lambda_i x) + \sinh(\lambda_i x))\Big\}\end{aligned} \tag{11}$$

The frequency drop for a crack at c is:

$$\Delta f_i(c,d) = f_i \cdot \gamma(0,d) \cdot \left(\bar{\phi}_i''(c)\right)^2 \tag{12}$$

The Relative Frequency Shift (RFS) for a cantilever beam with one crack is:

$$\Delta\bar{f}_i(c,d) = \gamma(0,d) \cdot \left(\bar{\phi}_i''(c)\right)^2 \tag{13}$$

This allows quick calculation of RFS for any crack depth and location, crucial for our PSO model.

2.3 Relative Frequency Shift for Multiple Cracks Scenario

When multiple cracks are present, each crack independently alters the stiffness of the beam and affects modal parameters, such as natural frequencies, mode shapes and damping factors. If the cracks are sufficiently far apart (more than 5mm), the superposition principle applies, allowing the effects to be considered independently (Gillich et al., 2021).

The total RFS for two cracks is:

$$\Delta\overline{f_i}(c1,d1,c2,d2) = \Delta\overline{f_i}(c1,d1) + \Delta\overline{f_i}(c2,d2) \quad (14)$$

Experiments show that for cracks at least 5mm apart, the EHB model applies with less than 0.005% error (Gillich et al., 2021).

In our study, we use these relations to calculate RFSs for any crack combination. This allows us to train our PSO model on various crack scenarios, solving the optimization problem of identifying damage properties from RFS data.

3 APPROACH AND METHODOLOGY

For our simulations we will consider a cantilever beam of 1000mm in length and a depth of 20mm.

We will be using a *GeneralOptimizerPSO* from the *Pyswarms* library (PYS,) as our PSO model. The following is the form of the objective function that the model must optimize:

$$f(solution) = -\left[\sum_{i=1}^{8}(\mathrm{RFS}_i(\mathrm{solution}) - \mathrm{InputRFS}_i)^2\right]^{-5} \quad (15)$$

where RFS_i is the function from equation 14, i denotes the frequency modes, the solution denotes two places and two depths for the cracks, and $\mathrm{InputRFS}_i$ denotes the relative frequencies shifts received as input for the fractured beam.

Going forward, whenever we discuss the characteristics of a fractured beam (or target) using a 4-value array such as [0.12, 0.65, 0.34, 1.34], the first two values, 0.12 and 0.65, denote the locations of the cracks on a scale ranging from 0 to 1, while the final two values, 0.34 and 1.34, represent the depths of the cracks on a scale ranging from 0 to 2. For locations, the boundaries of the space search are therefore [0 - 1] and for depths, [0 - 2]. The RFS equation will be used to construct the $InputRFS_i$ when we test our model by creating random targets using the previously outlined structure.

The following formula is used to calculate the error:

$$\mathrm{err(predicted)} = \frac{|\mathrm{predicted} - \mathrm{real}|}{\mathrm{length\ of\ interval}} * 100 \quad (16)$$

The length of the interval is $1 - 0 = 1$ in the case of location and $2 - 0 = 2$ in the case of depth.

3.1 Hyper-Parameters Tuning

We employed an exhaustive search (GridSearch) to optimize the parameters $c1$, $c2$, and w. Initially, we used 100 particles and 200 iterations, exploring the range [0.0 - 1.0] for each parameter with a target of [0.2, 0.3, 0.2, 0.1] and a Star topology. The optimal parameters found were $c1 = 0.4$, $c2 = 0.5$, and $w = 0.1$, but predictions with these settings were inconsistent, with errors up to 80

To address this, we ran GridSearch on 48 random targets, saving the parameters and costs after each run, and computed weighted averages: $c1 = 0.4$, $c2 = 0.6$, and $w = 0.2$. Despite the close similarity to the initial results, the error remained unsatisfactory.

We manually tested a broader range of values, discovering that higher $c1$ values improved performance due to the function's numerous local minima. Eventually, we identified $c1 = 3$, $c2 = 0.25$, and $w = 0.5$ as promising settings, occasionally achieving errors close to 1%.

To validate these findings, a comprehensive GridSearch with $c1$ and $c2$ in [0.2 - 5] and w in [0.1 - 1.1] was conducted, targeting [0.05878, 0.08467, 0.49865, 0.25434]. The results, $c1 = 4.2$, $c2 = 0.6$, and $w = 1.0$, confirmed the need for higher $c1$. However, due to concerns about local minima, we retained $c1 = 3$, $c2 = 0.25$, and $w = 0.5$. Despite these adjustments, error rates varied from 0.5% to 36%.

Further experimentation revealed the Ring topology to be more effective than the Star topology. The Ring topology limits communication to a finite number of neighbours, enhancing exploration and reducing the risk of convergence to local minima. We found that 50 neighbors per particle yielded the best results, regardless of total particle count (100, 200, or 700).

Finally, increasing the number of iterations from 200 to 1000 and the total number of particles from 100 to 700 was necessary for proper convergence. However, we later found that increasing the number of particles to 700 was not the most effective strategy, as will be discussed in the subsequent sections.

We conducted a comprehensive set of tests to identify the subintervals where the model performs well and where it encounters difficulties. The location interval was divided into four subintervals: [0.0 - 0.1], [0.1 - 0.8], [0.8 - 0.9], and [0.9 - 1.0], while the depth interval was divided into: [0.0 - 0.1], [0.1 - 0.5], [0.5 - 1.7], and [1.7 - 2.0]. We created 256 test cases using these location and depth values.

The error for each test case was calculated by averaging four errors (two for locations and two for depths). The model exhibited significant challenges when the depth was near 0 or close to 2, which, despite being practical for monitoring beam failure, highlighted its limitations. The model performed better for depths between 0.1 and 1.7, though inconsistencies persisted.

Notably, the model struggled with locations near the clamped end or the free end, particularly within the intervals [0.0 - 0.1] and [0.9 - 1.0]. This indicates specific areas where the model's accuracy needs improvement.

3.2 Our Approach

Observations of the model's performance indicated that results varied significantly for the same target, with error rates ranging from less than 1% to over 10%. To address this inconsistency, we implemented a strategy of running the model multiple times and aggregating the results.

First, we determined that the model performed similarly with 100 particles compared to 700 particles if the number of iterations was increased. Thus, we increased the iterations from 1000 to 3000 and ran the model three and five times per target. This approach improved overall performance but required subjective interpretation to identify the best results from multiple runs, as shown in Table 1.

Table 1: Multiple runs per target results.

# runs	Error	Location error	Depth error
3	5.7517%	4.1953%	5.9317%
5	3.8481%	2.2716%	3.1317%

For instance, we could obtain the optimal number for the first location from the first run of the model, and the optimal number for the second location from the third run, if we ran the model three times for a target.

To address this shortcoming we developed a novel "Best-of-Best" approach that involves merging outcomes from various iterations and honing a swarm population in order to create a new PSO model. Figure 2 shows our approach.

Our refined PSO model thus creates a new swarm population based on the best results from multiple runs. For each result, we generated additional particles by slightly adjusting the predicted locations and depths using a calculated step value, as in formula:

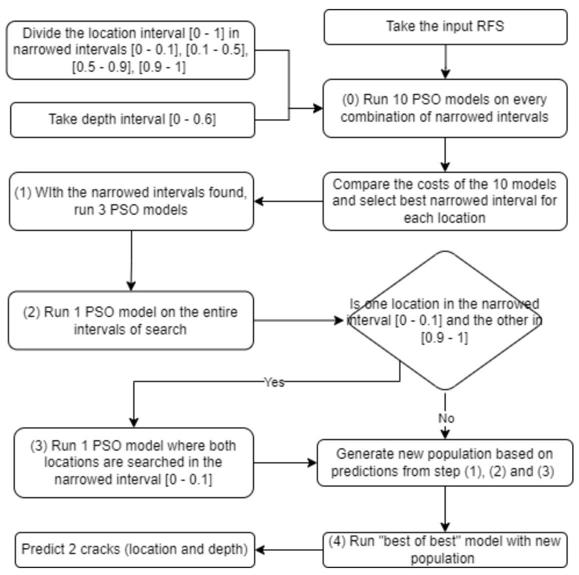
Divide the location interval [0 - 1] in narrowed intervals [0 - 0.1], [0.1 - 0.5], [0.5 - 0.9], [0.9 - 1]
Take the input RFS
Take depth interval [0 - 0.6]
(0) Run 10 PSO models on every combination of narrowed intervals
(1) With the narrowed intervals found, run 3 PSO models
Compare the costs of the 10 models and select best narrowed interval for each location
(2) Run 1 PSO model on the entire intervals of search
Is one location in the narrowed interval [0 - 0.1] and the other in [0.9 - 1]
Yes
No
(3) Run 1 PSO model where both locations are searched in the narrowed interval [0 - 0.1]
Generate new population based on predictions from step (1), (2) and (3)
Predict 2 cracks (location and depth)
(4) Run "best of best" model with new population

Figure 2: Flowchart of proposed algorithm.

$$\text{step} = \frac{\text{upper\_bound} - \text{lower\_bound} + 3}{\text{additional\_particles\_needed}//\text{nr\_of\_results}} \tag{17}$$

where $//$ is the floor division and the $upper_bound$ and $lower_bound$ are the intervals of search. For example, if we wanted to calculate the step for location, given 10 results and a new population of 200, the step would equal $(1-0+3)/(190//10) = 4/19 \approx 0.21$. With this step, the new particles would be created by adding to each predicted location a random number from the interval $(-\text{step}, \text{step})$.

Our updated PSO model required appropriate hyper-parameters, which we determined through GridSearch and manual testing. We found that $c1 = 3$, $c2 = 0.2$, and $w = 0.4$ worked best, with a Ring topology and 40 neighbours. We also increased the number of particles from 100 to 200.

Testing this new model, we ran it 5 and 10 times for the same target, collected the results, and ran a final "Best-of-Best" PSO model. Although the overall performance error was not significantly higher than before, the location detection improved considerably, achieving a low error of 0.71%, while depth detection was less accurate at 4.79% in the 10 runs scenario (Table 2, row 1 and 2).

The "Best-of-Best" model not only selected the best results from previous runs but also occasionally identified better predictions by centring the new population around the prior best results.

Table 2: Performance overview.

Nr.	Runs on entire intervals	Runs with narrowed intervals	Runs on low locs interval	Depth narrowed intervals Y/N	Loc bounds	Depth bounds	Avg err %	Avg loc err %	Avg depth err %
1	5	-	-	-	[0 - 1]	[0 - 2]	5.96	2.93	8.99
2	10	-	-	-	[0 - 1]	[0 - 2]	2.75	0.71	4.79
3	2	5	1 or 3	Y	[0 - 1]	[0 - 2]	3.73	2.44	5.01
4	0	5	1 or 3	N	[0 - 1]	[0 - 2]	4.83	3.77	5.89
5	2	5	1 or 3	N	[0 - 1]	[0 - 2]	3.38	0.66	6.10
6	2	5	1 or 3	N	[0 - 1]	[0 - 1]	1.30	1.35	1.26
7	5	-	-	-	[0 - 1]	[0 - 1]	1.86	1.72	1.99
8	2	5	1 or 3	N	[0 - 0.99]	[0 - 1]	0.76	0.72	0.81
9	1	3	1	N	[0 - 1]	[0 - 0.3]	1.27	0.80	1.75
10	1	3	1	N	[0 - 1]	[0 - 0.6]	1.06	0.89	1.23

3.2.1 Improving the Performance Using Narrowed Intervals of Search

One can however notice that the model's error margin was minimal when it successfully identified the target, but significantly escalated (up to 30%) when it failed. This led us to conclude that narrowing the search intervals would improve precision. We divided the location range [0 - 1] into four smaller intervals: [0 - 0.1], [0.1 - 0.5], [0.5 - 0.9], and [0.9 - 1]. PSO models were executed within these intervals, resulting in ten runs to cover all combinations. A similar approach was applied to the depth range [0 - 2], dividing it into: [0.0 - 0.2], [0.2 - 1], [1 - 1.8], and [1.8 - 2].

Next, we ran 5 PSO models to search within these new narrowed intervals of search and 2 PSO models over the entire intervals ([0 - 1] for location and [0 - 2] for depth) to maintain diversity in the population of the "best of best" model. We noticed that the model often misinterpreted cases where both cracks were close to the clamped end, within [0 - 0.1], sometimes detecting one crack in [0 - 0.1] and the other at [0.9 - 1]. To address this, we added a check to run additional PSO models if only one damage was detected in the [0 - 0.1] subinterval.

Summarizing the algorithm steps: run 10 PSO models for the best narrowed intervals for locations and depths; run 5 PSO models within these intervals; run 2 PSO models over the entire intervals; if necessary, run additional PSO models within the [0 - 0.1] interval. This process is followed by the "best of best" PSO model.

However, the algorithm's performance was unsatisfactory, as shown in Table 2, row 3: average error of 3.73%, location error of 2.44%, and depth error of 5.01%. The 10 runs for narrowed depth intervals often produced misleading outcomes, and removing the 2 runs over entire intervals worsened results to an average error of 4.83% (row 4). Reintroducing the 2 runs improved the location error to 0.66% (row 5), but the depth error remained high.

Given the impracticality of predicting cracks deeper than half the beam's height, we revised the depth search range to [0 - 1], which significantly reduced the depth error to 1.26%, although location error slightly increased (row 6).

The significant difference between tests 5 and 6 led us to question whether the technique we used to try to focus our search for cracks was really essential. In a different experiment, we just used five runs over the whole interval and the "best of best" model at the conclusion. The results showed that our suggested approach was more efficient than the previous one, with average errors of 1.86%, 1.72%, and 1.99%, respectively, greater than in the previous test.

Further experiments suggested that narrowing the search range was effective. Limiting the location search to [0 - 0.99] produced the best results (average error of 0.76%, location error of 0.72%, and depth error of 0.81%), though this approach didn't align with the objective of searching the entire beam length and it is more informative.

To optimize the algorithm's efficiency, we reduced the number of PSO runs: from 2 to 1 for entire intervals, from 5 to 3 for narrowed intervals, and conditionally adjusted runs within [0 - 0.1]. Additionally, we reduced the depth search range to 15% and 30% of the beam's height (rows 9 and 10), with the 30% depth experiment showing the most promise.

These adjustments indicate that while narrowing the search intervals improves precision, maintaining some broader searches ensures robustness. The evolving approach demonstrates the potential for enhancing the model's efficiency and accuracy, with ongoing refinements needed for practical Structural Health Monitoring applications.

4 PERFORMANCE OVERVIEW

The errors dependent on the search intervals are presented in Table 2 (row 10). For location [0 - 1] and for depth [0 - 0.6], the results are very satisfactory. An average location error of 0.89% on a 1000mm long beam translates to 0.89cm (or 8.9mm). For depth, covering the first 60mm of a 200mm deep beam, an average error of 1.23% translates to 0.738mm. This means our algorithm achieves an error for location under 1cm and an error for depth under 1mm.

Expanding on experiment 10 from Table 2, which was conducted on 300 uniform targets, Table 3 shows the percentage of locations and depths found under different errors. It should be noted that the errors for location and depth are calculated differently.

Table 3: Percentage of found with different errors.

Type	Error		
	$< 1\%$	$< 2\%$	$< 2\%$
Locations	96.16%	97.5%	97.66%
At least 1 out of 2 locations	100%	100%	100%
Depths	89.5%	91%	93.33%
At least 1 out of 2 depths	96%	96.33%	97.66%

Table 6 highlights 15 random samples from the 300 targets, showing that our model performs poorest when the cracks are at the clamped end and has higher accuracy for locations than depths.

Our results are comparable to related work. For instance, the study (Sahu et al., 2018) used a CSAGA model (a combination of clonal selection algorithm and genetic algorithm). As shown in Table 4, our model surpassed CSAGA in prediction accuracy.

Our model also matches the ANN model developed by the authors in (Maurya et al., 2018). Table 5 demonstrates that both models made nearly identical predictions, with error rates close to 0%.

While it is challenging to compare our model's performance directly with related work due to variations in beam properties and benchmarks, we can confidently state that our PSO model maintains a high standard, with an overall error close to 1%.

Table 4: CSAGA model vs our model.

Target	CSAGA	Our Model
[0.125, 0.1375, 0.4375, 0.1625]	[0.12228, 0.1344, 0.4273, 0.159]	[0.12493315, 0.13697042, 0.43517278, 0.1685001]
[0.15625, 0.225, 0.53125, 0.2625]	[0.15276, 0.2173, 0.5196, 0.25633]	[0.15625005, 0.22500022, 0.53125016, 0.26249971]
[0.21875, 0.1875, 0.46875, 0.225]	[0.21347, 0.18335, 0.4569, 0.2198]	[0.21874927, 0.18749748, 0.46875855, 0.22498143]

Table 5: ANN model vs Our model.

Target	ANN	Our Model
[0.006, 0.166, 0.4, 0.4]	[0.006, 0.166, 0.4, 0.4]	[0.006, 0.16599, 0.39997, 0.40001]
[0.166, 0.333, 0.4, 0.4]	[0.165, 0.332, 0.417, 0.417]	[0.166, 0.333, 0.40001, 0.39999]
[0.333, 0.5, 0.4, 0.4]	[0.333, 0.5, 0.4, 0.4]]	[0.33296, 0.4995, 0.39995, 0.4003]

5 CONCLUSIONS

We investigated several parameters for PSO when applying this method for predicting the locations and severities of 2 cracks in prismatic beams based on the RFSs and PSO. From the tests performed, we identified good parameters by hyper-parameter tuning and obtained excellent estimations, both for the 2 crack positions and their respective depths using our own developed ”Best of Best” approach. The average error of less than 1% and less than 1.25% for the severity means that our proposed method can be successfully applied to the stated problem.

Two problems would arise with our methodology. First off, if the cracks are closer than 5 mm, the superposition principle will not hold. Secondly, our algorithm might not scale properly. Although the superposition can be used for any number of cracks, extending the approach to accommodate more cracks in subsequent work may provide difficulties. In particular, challenges could surface during the first stage when each location's optimal narrowed interval is being searched for.

Table 6: Example of predictions.

Target	Predicted	Error %	Mean Error %
[0.4598, 0.7152, 0.5841, 0.4266]	[0.4598, 0.7152, 0.5841, 0.4265]	[0, 0, 0.0028, 0.0042]	0.0018
[0.6893, 0.1113, 0.5933, 0.1565]	[0.6893, 0.1118, 0.5928, 0.1573]	[0.0018, 0.0474, 0.0794, 0.1409]	0.0674
[0.1803, 0.2550, 0.4382, 0.0676]	[0.1803, 0.2553, 0.4382, 0.0676]	[0.0003, 0.0375, 0.0016, 0.0075]	0.0117
[0.1078, 0.8705, 0.0022, 0.1403]	[0.0926, 0.8705, 0.0487, 0.1402]	[1.5236, 0.0024, 7.7486, 0.0198]	2.3236
[0.7733, 0.8200, 0.3782, 0.0261]	[0.7732, 0.8398, 0.3776, 0.0421]	[0.0122, 1.9783, 0.0946, 2.6562]	1.1853
[0.5701, 0.2046, 0.1945, 0.4737]	[0.2046, 0.5701, 0.4737, 0.1945]	[0, 0, 0, 0]	0
[0.2875, 0.7504, 0.0967, 0.4868]	[0.2875, 0.7504, 0.0967, 0.4868]	[0, 0, 0, 0]	0
[0.1468, 0.2139, 0.4900, 0.5009]	[0.1468, 0.2139, 0.4900, 0.5009]	[0, 0, 0, 0]	0
[0.1280, 0.6299, 0.4566, 0.4638]	[0.1280, 0.6299, 0.4566, 0.4638]	[0, 0, 0, 0]	0
[0.9556, 0.8967, 0.5819, 0.5564]	[0.9207, 0.8910, 0.4809, 0.4474]	[3.4891, 0.5681, 16.8419, 18.1578]	9.7642
[0.3273, 0.4975, 0.0075, 0.1942]	[0.3273, 0.4975, 0.0429, 0.1942]	[0.0005, 0.0004, 5.9108, 0.0001]	1.4780
[0.0028, 0.4900, 0.1298, 0.2172]	[0.0030, 0.4905, 0.1303, 0.2167]	[0.0159, 0.0515, 0.0800, 0.0851]	0.0581
[0.9128, 0.6575, 0.4073, 0.2701]	[0.9128, 0.6575, 0.4073, 0.2701]	[0, 0, 0, 0]	0
[0.8813, 0.8050, 0.2279, 0.5397]	[0.8813, 0.8050, 0.2279, 0.5397]	[0, 0, 0, 0]	0
[0.0623, 0.5662, 0.4172, 0.5537]	[0.0623, 0.5662, 0.4172, 0.5537]	[0, 0, 0, 0]	0

For future research, we will also focus on comparing the proposed PSO approach with other Neural Network based techniques.

REFERENCES

Pyswarms's documentation. https://pyswarms.readthedocs.io/en/latest/. Accessed: 2024-05-19.

Gillich, G.-R., Aman, A. T., Abdel Wahab, M., and Tufisi, C. (2019). Detection of multiple cracks using an energy method applied to the concept of equivalent healthy beam. In *Proceedings of the 13th International Conference on Damage Assessment of Structures: DAMAS 2019, 9-10 July 2019, Porto, Portugal*, pages 63–78. Springer.

Gillich, G.-R., Maia, N. M., Wahab, M. A., Tufisi, C., Korka, Z.-I., Gillich, N., and Pop, M. V. (2021). Damage detection on a beam with multiple cracks: a simplified method based on relative frequency shifts. *Sensors*, 21(15):5215.

Gillich, G.-R. and Praisach, Z.-I. (2014). Modal identification and damage detection in beam-like structures using the power spectrum and time–frequency analysis. *Signal Processing*, 96:29–44.

Greco, A., Pluchino, A., Cannizzaro, F., Caddemi, S., and Caliò, I. (2018). Closed-form solution based genetic algorithm software: application to multiple cracks detection on beam structures by static tests. *Applied Soft Computing*, 64:35–48.

Khaji, N. and Mehrjoo, M. (2014). Crack detection in a beam with an arbitrary number of transverse cracks using genetic algorithms. *Journal of Mechanical Science and Technology*, 28:823–836.

Khatir, S., Belaidi, I., Khatir, T., Hamrani, A., Zhou, Y.-L., and Wahab, M. A. (2017). Multiple damage detection in composite beams using particle swarm optimization and genetic algorithm. *Mechanics*, 23(4):514–521.

Maurya, M., Mishra, R., and Panigrahi, I. (2018). Multi crack detection in structures using artificial neural network. In *IOP Conference Series: Materials Science and Engineering*, volume 402, page 012142. IOP Publishing.

Mohan, S., Maiti, D. K., and Maity, D. (2013). Structural damage assessment using frf employing particle swarm optimization. *Applied Mathematics and Computation*, 219(20):10387–10400.

Moradi, S. and Kargozarfard, M. H. (2013). On multiple crack detection in beam structures. *Journal of mechanical science and technology*, 27:47–55.

Pop, M.-V., Tufisi, C., and Gillich, G.-R. (2022). Determining the position of two cracks in a cantilever beam using artificial neural networks. *Vibroengineering Procedia*, 46:14–20.

Praisach, Z. I., Gillich, G. R., Protocsil, C., and Muntean, F. (2013). Evaluation of crack depth in beams for known damage location based on vibration modes analysis. *Applied Mechanics and Materials*, 430:90–94.

Sahu, S., Kumar, P. B., and Parhi, D. R. (2018). A hybridised csaga method for damage detection in structural elements. *Mechanics & Industry*, 19(4):407.

Zheng, S., Liang, X., Wang, H., and Fan, D. (2014). Detecting multiple cracks in beams using hierarchical genetic algorithms. *Journal of Vibroengineering*, 16(1):341–350.