PSO-Based Optimization for Identifying Worst-Case Crash Scenarios in Vehicle Safety Systems

Fabio Berberi[1] and Paolo Mercorelli[2]

[1] Department of Engineering, University of Siena, Italy
fabio.berberi@student.unisi.it
[2] Institute of Product and Process Innovation, Leuphana University of Lüneburg, Germany paolo.mercorelli@leuphana.de

Abstract. Ensuring vehicle safety requires analyzing both active and passive systems under a wide range of crash conditions. Traditional crash testing is expensive, while exhaustive simulations are computationally demanding. Crash parameters such as impact velocity, collision angle, vehicle mass ratio, and structural stiffness are modeled as optimization variables, following previous studies on vehicle crashworthiness optimization with multi-objective PSO [5].The method directs the search toward scenarios maximizing injury criteria such as Head Injury Criterion (HIC) and chest deformation [9]. Furthermore, a Domain-as-Particle PSO (DaP-PSO) is introduced, where groups of particles explore subdomains hierarchically, improving convergence and robustness compared to standard PSO [4]. Simulation results show that DaP-PSO detects critical crash conditions with fewer evaluations than classical PSO, offering a promising tool for cost-effective virtual safety testing [1].

Crash simulation, vehicle safety, passive safety, active safety, worst-case scenarios, PSO, Domain-as-Particle.

1 Introduction

Vehicle safety evaluation involves both active safety systems, which prevent collisions, and passive safety systems, which mitigate injury severity during a crash. Assessing passive safety performance requires simulating a wide range of collision scenarios, varying parameters such as impact speed, angle, and structural stiffness. Exhaustive exploration of all combinations is infeasible, motivating the need for optimization-driven approaches.

Metaheuristic methods such as Particle Swarm Optimization (PSO) [3] provide efficient alternatives to brute-force search. By guiding a swarm of candidate solutions toward high-risk crash conditions, PSO can significantly reduce the number of simulations while still capturing critical scenarios. However, standard PSO may stagnate in local optima or fail to adequately cover large parameter spaces.

This paper introduces a PSO-based approach for worst-case crash detection, extended with a Domain-as-Particle (DaP) variant to enhance search diversity and convergence reliability.

2 Background and Related Work

Crashworthiness evaluation typically relies on physical crash tests or numerical simulation models such as finite element methods (FEM). Although accurate, these approaches are computationally demanding when applied to large design spaces. Prior works have applied optimization and machine learning to reduce testing effort, but few have explicitly focused on identifying worst-case crash conditions.

PSO has been applied in automotive design optimization, yet classical PSO often suffers from premature convergence. Hierarchical strategies, such as multi-swarm or domain-based PSO, have been shown to improve search efficiency in other engineering problems. Building on these insights, we propose DaP-PSO for crash safety applications.

3 Problem Formulation

A crash scenario is represented as a vector of parameters that describe the main conditions of the impact. The crash parameter space $\mathcal{W}$ is defined by the following variables:

- v: the impact velocity [km/h],
- θ: the impact angle [°], distinguishing frontal ($\theta \approx 0°$), oblique, and lateral ($\theta \approx 90°$) collisions,
- m_r: the vehicle mass ratio, accounting for asymmetric collisions,
- k: the structural stiffness coefficient, related to deformation properties of the vehicle.

Each candidate scenario is therefore represented by:

$$\mathbf{x} = (v, \theta, m_r, k) \in \mathcal{W}.$$

The objective is to identify the vectors $\mathbf{x}$ that lead to the most severe crash outcomes. To quantify crash severity, we define the objective function:

$$J(\mathbf{x}) = w_1 \cdot HIC(\mathbf{x}) + w_2 \cdot CD(\mathbf{x}),$$

where $HIC(\mathbf{x})$ is the Head Injury Criterion and $CD(\mathbf{x})$ is the chest deformation. The weights $w_1, w_2 > 0$ balance the contribution of each metric depending on the safety standards adopted.

Figure 1 provides an illustrative example of this problem formulation. The plot shows a Monte Carlo sampling of the parameter space with respect to two main variables, velocity v and impact angle θ, while the other two variables (m_r

and k) are varied randomly. The color scale indicates the severity $J(\mathbf{x})$, and the highlighted point marks the worst-case scenario identified among the sampled crashes. This visualization clarifies the optimization goal: systematically explore the crash parameter space to detect the combinations that maximize injury severity.

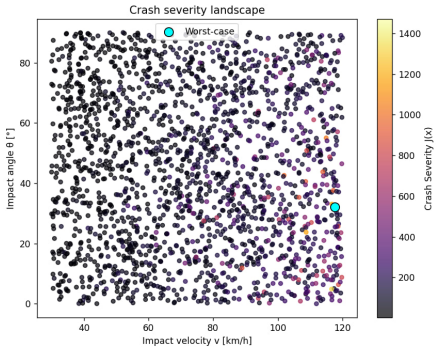

Fig. 1. Illustrative crash severity landscape: sampled crash scenarios in the (v, θ) space with severity $J(\mathbf{x})$ represented by the color map. The highlighted point corresponds to the worst-case scenario found.

4 Classical PSO for Crash Scenarios

In standard PSO, each particle represents a candidate crash scenario. Particle velocities are updated using cognitive and social components, driving particles toward personal and global best solutions. While effective in moderate search spaces, classical PSO may require many iterations to identify extreme worst-case scenarios.

5 Domain-as-Particle PSO

The proposed Domain-as-Particle PSO (DaP-PSO) introduces a hierarchical exploration strategy designed to overcome the main limitations of classical PSO, namely premature convergence and insufficient coverage of the parameter space. Instead of treating each particle as an isolated candidate solution, the global parameter space $\mathcal{W}$ is partitioned into multiple subdomains, each of which is treated as a higher-level particle.

Within each subdomain, a set of candidate crash scenarios is generated and evaluated. The subdomain is represented by its center, which evolves over time. The motion of each subdomain is determined by a characteristic vector, computed from the best-performing candidates inside it. This ensures that the movement reflects the local structure of the search space and avoids the erratic updates typical of standard particles.

Moreover, each subdomain does not evolve in isolation: it is influenced by (i) its own best-performing solutions, (ii) discoveries made in neighboring subdomains, and (iii) the global best scenario found so far. This combination enables a coordinated search process where promising information propagates across regions. As a result, DaP-PSO maintains diversity in the exploration phase while ensuring reliable convergence.

The advantages of this formulation can be summarized as follows:

- **Improved exploration**: dividing $\mathcal{W}$ into subdomains allows different regions to be explored in parallel, reducing the risk of premature convergence to a local optimum.
- **Adaptive convergence**: characteristic vectors provide smoother and more stable updates compared to classical velocity rules, ensuring that the evolution of subdomain centers reflects consistent trends rather than noise.
- **Coordinated information sharing**: neighbor interactions allow knowledge transfer between subdomains, spreading promising directions across the parameter space.
- **Scalability**: the decomposition into subdomains distributes the computational effort, making the method more suitable for large and complex crash parameter spaces.

Figure 2 illustrates this mechanism. Each ellipse represents a subdomain of $\mathcal{W}$, shifting from its initial (solid) to its updated (dotted) position under the combined effect of its characteristic vector and feedback from neighboring regions.

5.1 Parameter Space and Subdomain Initialization

The crash parameter space $\mathcal{W}$ represents the global domain of all possible crash scenarios. In the proposed DaP-PSO framework, $\mathcal{W}$ is partitioned into multiple subdomains D_j, which are distributed uniformly across the space. Each subdomain is associated with a center point (initially placed in a grid-like configuration) and acts as a higher-level particle in the swarm.

This uniform initialization ensures that the search starts with a balanced coverage of the parameter space, avoiding early bias toward specific regions. Figure 3 illustrates this initialization strategy, where circles represent subdomains distributed across the global parameter space $\mathcal{W}$.

6 Particle Generation and Surrogate Modeling

After defining the subdomains (see Fig. 3), the next step is to generate candidate weight vectors inside each region. The goal is to sample the parameter space in a way that guarantees both a sufficiently accurate coverage of each subdomain and an efficient management of computational resources [6, 8, 7].

Two distinct types of particles are generated:

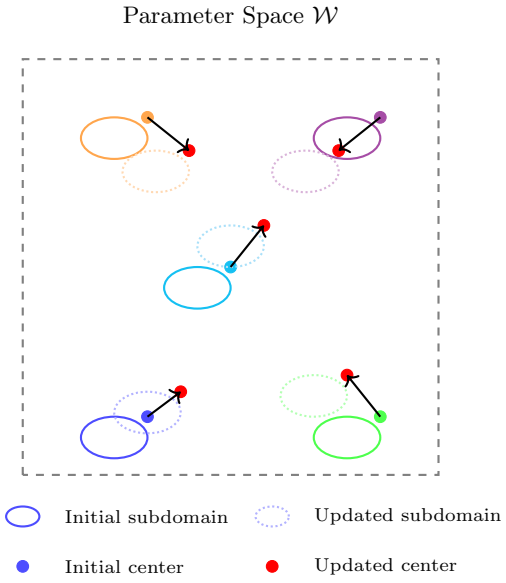

Fig. 2. Illustration of subdomain movement in parameter space.

- **Full particles (black)**: these are densely and uniformly distributed within the subdomain D_j. Each black particle represents a candidate weight vector that is directly evaluated using the true cost function. Because the cost function evaluation is expensive, the number of full particles must be carefully balanced: too few would lead to insufficient coverage of the search space, while too many would dramatically increase computational time. Their role is to ensure a fine-grained exploration, so that no promising regions are overlooked.
- **Surrogate particles (red)**: these are also uniformly distributed inside D_j, but with a significantly lower density than the full particles. Their main purpose is not to be directly evaluated, but rather to provide training data for a surrogate model (such as Random Forest or Gradient Boosting). The surrogate acts as an approximation of the real cost function, enabling us to quickly estimate the quality of candidate solutions. Only a limited subset of surrogate-predicted candidates, typically those with the best predicted performance, are then promoted to full evaluation. This hierarchical strategy allows us to discard unpromising regions early, saving computational resources.

In summary, the dense grid of black particles guarantees *reliability* in exploring the subdomain, while the sparser red particles support the construction of a surrogate that provides *efficiency* by filtering candidates. The synergy between the two particle types creates a balance between exploration and computational cost, making the search process both scalable and robust [2].

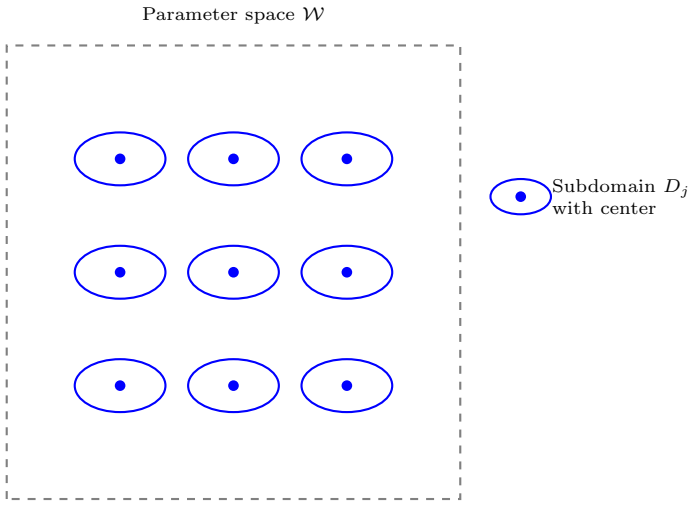

Fig. 3. Uniform distribution of oval-shaped subdomains D_j inside the parameter space $\mathcal{W}$. Each ellipse represents a subdomain with its center.

6.1 Surrogate Model Construction and Usage

In order to reduce the number of expensive evaluations of the true cost function, a surrogate model is introduced. The surrogate exploits the information provided by the surrogate particles (red), which are uniformly distributed within each subdomain D_j but evaluated less frequently. Instead of being directly validated, these particles provide approximate labels (composite values) that are used to train an ensemble learning model.

Figure 4 illustrates this process: surrogate particles are first associated with composite values derived from preliminary evaluations. These data points are then used to train an ensemble of machine learning regressors, in our case Random Forest and Gradient Boosting. The ensemble prediction is obtained by averaging the outputs of both models, resulting in a stable and robust approximation of the cost function. This surrogate model can then be queried to estimate the value of normal (black) particles without performing the costly original evaluation.

Once the surrogate is trained, all particles in D_j can be rapidly scored. The workflow, illustrated in Fig. 5, proceeds as follows:

1. Each candidate particle is assigned a predicted composite value by the surrogate.
2. Only the top-K candidates (those with the best predicted performance) are selected.
3. These top-K candidates are then validated with the true cost function (e.g., a neural network evaluation), obtaining their real composite values.

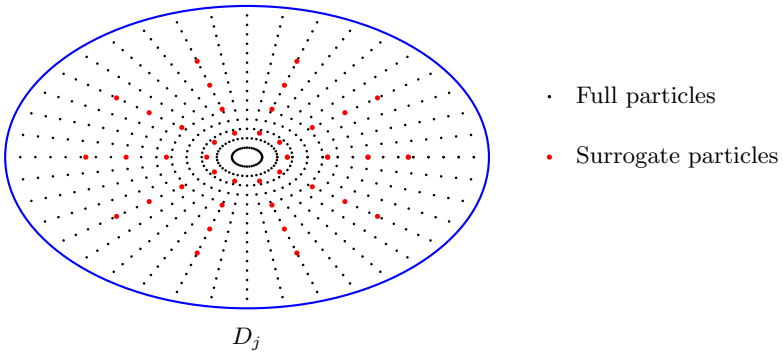

Fig. 4. Dense uniform distribution of full (black) and sparse surrogate (red) particles inside a subdomain D_j, excluding the boundary.

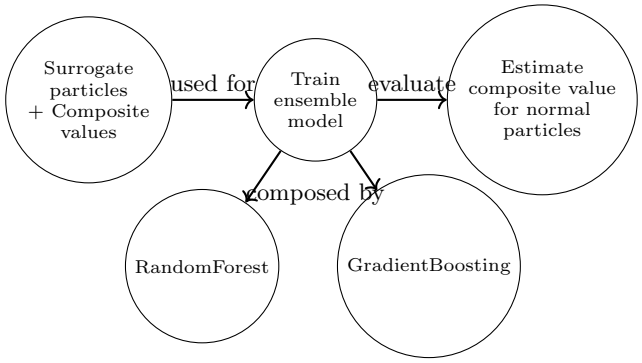

Fig. 5. Schematic of surrogate model creation with circular nodes: surrogate particles provide composite values to train an ensemble (RandomForest and GradientBoosting), which is then used to estimate composite values for normal candidate particles.

This strategy ensures that only a small fraction of particles undergo the expensive full evaluation, while the surrogate filters out unpromising candidates. In this way, we achieve a favorable trade-off between exploration of the search space and computational efficiency.

7 Methodology

The proposed framework enhances classical PSO by combining a *domain-as-particle* representation with surrogate-assisted evaluation. The workflow is structured into three main components: *domain update*, *surrogate management*, and the *iterative optimization loop*.

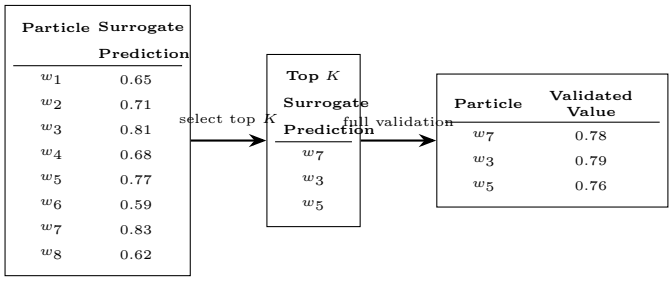

Fig. 6. Workflow of candidate selection: All candidate particles with surrogate predictions (left); selection of top K candidates (center); true composite values after full validation with the neural network (right).

7.1 Domain Update

Each subdomain D_j is represented by a *characteristic vector* c_j that determines its current position in the search space. The update of c_j follows a Particle Swarm Optimization (PSO)-like dynamic, where four contributions jointly define the new direction and magnitude of movement:

- **Inertia** ($\omega v^{(t-1)}$): retains part of the previous velocity, ensuring continuity in motion. This prevents abrupt oscillations and preserves momentum from past explorations.
- **Personal best influence** (Δ^p): pulls the domain center toward its historically best position p_j, acting as a local exploitation mechanism.
- **Neighbor influence** (Δ^n): drives exploration in coordination with adjacent subdomains, enabling information sharing across the search space.
- **Global best influence** (Δ^g): attracts all domains toward the globally best solution g, ensuring convergence at the global level.

The update rules are expressed as:

$$v^{(t)} = \omega v^{(t-1)} + \Delta^p + \Delta^n + \Delta^g,$$

$$c_j^{(t)} = c_j^{(t-1)} + v^{(t)}.$$

Here, ω is the inertia weight, while Δ^p, Δ^n, and Δ^g are stochastic contributions defined as: $\Delta^p = \phi_p r_p \big(p_j - c_j^{(t-1)}\big)$,
$\Delta^n = \phi_n r_n \big(n_j - c_j^{(t-1)}\big)$,
$\Delta^g = \phi_g r_g \big(g - c_j^{(t-1)}\big)$, with $r_p, r_n, r_g \sim U(0,1)$ random coefficients.

7.2 Surrogate Management

Since full evaluations are computationally expensive, a two-stage evaluation strategy is introduced:

- **Surrogate prediction**: all candidate particles are first assessed using lightweight surrogate models.
- **Top-K selection**: only the most promising candidates are promoted to full evaluation.
- **Surrogate refresh and retraining**: outdated surrogate particles are discarded and replaced, while the surrogate model is updated with new validated results.

This ensures that computational resources are concentrated on candidates with higher potential, while preserving exploration of the search space.

7.3 Iterative Workflow

The optimization proceeds through a nested iteration process, summarized in Figures 6 and **??**. For each domain and iteration:

1. Extraction of surrogate particles from the region of interest.
2. Training of the surrogate model and prediction of candidate scores.
3. Selection of top-K candidates based on surrogate estimates.
4. Full validation of the selected candidates with the reference cost function.
5. Update of the characteristic vector according to validated results.
6. Movement of the domain center, enabling progressive exploration of the parameter space.

The process repeats across all domains until one of the stopping criteria is satisfied (target score, iteration budget, or computational resource limit).

7.4 Risk Value Definition

In order to identify worst-case scenarios, each particle explores the crash parameter space $(v, \theta, o, \ldots)$ and is associated with a scalar metric, called the *Risk Value*:

$$R(v, \theta, o, \ldots) = \alpha \cdot a_{\max} \; + \; \beta \cdot \Delta v \; + \; \gamma \cdot P_{injury}, \tag{1}$$

where:

- $a_{\max}$ denotes the peak deceleration experienced during the crash,
- Δv is the relative change in velocity between colliding bodies,
- P_{injury} is the probability of injury derived from safety models,
- α, β, γ are weighting coefficients that balance the contribution of each term.

This composite metric provides a unified measure of crash severity. The optimization process is therefore formulated as a maximization problem: particles iteratively update their positions in the parameter space, moving towards regions that yield higher values of R in Eq. eq:risk-value. In this way, the swarm converges towards critical and potentially unsafe crash configurations, which are of particular interest for safety validation.

8 Experimental Results

The goal of the experimental campaign is to evaluate the ability of two optimization strategies — the standard Particle Swarm Optimization (PSO) and the proposed Domain-as-Particle PSO (PSO-DAP) — to maximize the risk value defined in Eq. (X). Both methods are applied to the same parameter space, consisting of velocity v, angle θ, and overlap ratio o. The criticality of each crash scenario is quantified by the objective function, and the optimizer aims at finding configurations that yield the highest risk values.

8.1 Standard PSO Setup

The first experiment employs a classical PSO implementation, where each particle directly explores the search space. The main settings are reported in Table 1.

Table 1. Parameter configuration for standard PSO.

Parameter	**Value**
Number of particles	3000
Iterations	100
Velocity weight w	0.7
Cognitive coefficient c_1	1.5
Social coefficient c_2	1.5
Bounds on v	[10, 100] km/h
Bounds on θ	[0, $\pi/2$] rad
Bounds on o	[0.1, 1.0]

The convergence behaviour is monitored by recording the best global risk value across iterations, and a final scatter plot illustrates the distribution of particles with respect to velocity and angle.

8.2 Domain-as-Particle PSO Setup

The second experiment adopts the Domain-as-Particle approach (PSO-DAP). Here, the search space is initially partitioned into multiple domains, which act as high-level particles. Each domain generates internal candidate points, and its performance is evaluated according to the maximum risk value discovered within it. Domains are progressively refined: the best domain is shrunk to focus the search, while the others are shifted towards it.

The setup is summarized in Table 2.

The evolution of risk values across domains is tracked at each iteration, and the highest-scoring domain drives the search. This hierarchical mechanism allows PSO-DAP to more effectively explore the space and converge towards highly critical crash configurations.

Table 2. Parameter configuration for Domain-as-Particle PSO.

Parameter	Value
Number of domains	5
Particles per domain	100
Iterations	5
Bounds on v	[0, 100] km/h
Bounds on θ	[0, $\pi/2$] rad
Bounds on o	[0, 1.0]
Domain refinement	Shrink best, shift others

8.3 Crash Scenario

The crash is modeled as a controlled frontal-to-oblique impact, where the relative velocity and stiffness of the vehicles are varied. The configuration is represented through a *characteristic vector*:

$$\mathbf{x} = [v,\ \theta,\ m_r,\ k]$$

where:

- v: relative impact velocity [km/h],
- θ: impact angle [degrees],
- m_r: mass ratio between vehicles,
- k: equivalent stiffness parameter.

8.4 Objective Function

The severity of each configuration is quantified through a composite risk metric:

$$J(\mathbf{x}) = \alpha \cdot HIC + \beta \cdot ChestDefl.$$

where HIC (Head Injury Criterion) and chest deflection are weighted with coefficients α, β calibrated from experimental crash test data. This metric ensures that the optimization targets realistic safety-relevant outcomes.

E. Algorithms Compared

Two optimization strategies were implemented and compared: the *Classical PSO* and the *Domain-as-Particle PSO (DAP-PSO)*.

Classical PSO: The algorithm was initialized with **3,000 particles** and iterated for **100 generations**. The inertia weight was set to $w = 0.7$, and the cognitive and social coefficients were $c_1 = c_2 = 1.5$. Each particle explored the three-dimensional parameter space defined by:

$$v \in [10, 100]\ km/h, \quad \theta \in [0, \pi/2]\ rad, \quad o \in [0.1, 1].$$

The optimization targeted the maximization of the risk function, with the global best updated at each iteration.

Domain-as-Particle PSO (DAP-PSO): In this hierarchical approach, the search space was initially divided into **5 domains**. Each domain contained **100 particles**, sampled uniformly within its local bounds. The process was iterated for **5 global iterations**, during which domains were dynamically updated: the best-performing domain was shrunk around its center, while the remaining domains were shifted toward it. The same risk function was used as objective, ensuring comparability with the Classical PSO.

Both algorithms were evaluated in terms of *maximum achieved risk value*, *convergence behavior*, and *computational time*. While Classical PSO relied on a large swarm and extended iterations, DAP-PSO exploited domain-level exploration to reduce stagnation and accelerate convergence.

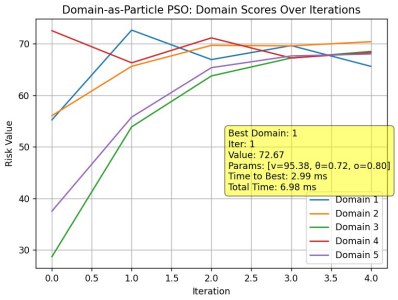

Fig. 7. Convergence of Classical PSO with 3000 particles and 100 iterations.

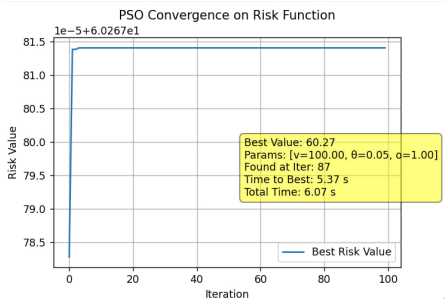

Fig. 8. Convergence of Domain-as-Particle PSO with 5 domains and 100 particles per domain.

9 Discussion and Conclusion

The results highlight the advantages of hierarchical domain exploration. By embedding subdomain dynamics, the proposed Domain-as-Particle PSO (DaP-PSO) balances exploration and exploitation more effectively than classical PSO. This hierarchical search enables faster identification of critical crash cases, while also reducing computational burden and improving safety insights.

Overall, the study confirms the superiority of DaP-PSO over the classical formulation for identifying worst-case crash scenarios. The algorithm proved capable of focusing the search on promising regions of the parameter space, leading to more consistent convergence and enhanced interpretability of results.

Future Work: Several directions remain open for further development:

– *Parallelization:* running multiple domains in parallel to significantly reduce computation time and make the approach scalable for large datasets.

- *Multi-objective optimization:* extending the framework to jointly consider competing safety objectives, such as minimizing injury risk while maximizing system robustness.
- *Real-world datasets:* validating the approach with experimental crash data to confirm its effectiveness beyond simulation.
- *Integration with active safety systems:* coupling DaP-PSO with online controllers to support adaptive safety strategies in real-time driving scenarios.
- *Hybrid surrogate models:* combining machine learning surrogates (e.g., Random Forest, Gaussian Processes) with DaP-PSO to reduce evaluation cost while maintaining accuracy.

In conclusion, DaP-PSO represents a promising extension of PSO for complex safety-related optimization tasks, with clear potential for real-world automotive applications.

Acknowledgment

The authors acknowledge that the knowledge required to develop this paper was acquired in the course *Applied Algorithms in Estimation and in Control of Technical, Economical, and Biological Dynamical Systems* at Leuphana University of Lüneburg, taught by Prof. Paolo Mercorelli.

References

1. Ieee iv 2026 - call for papers. Accessed: 2025-08-21
2. Aye, C.M., Pholdee, N., Yildiz, A.R., Bureerat, S., Sait, S.M.: Multi-surrogate-assisted metaheuristics for crashworthiness optimisation. International Journal of Vehicle Design **80**(2/3/4), 223–240 (2019). https://doi.org/10.1504/IJVD.2019.109866
3. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science. pp. 39–43. IEEE (1995). https://doi.org/10.1109/MHS.1995.494215
4. Elhafsi, A., Khattak, S., Clark, T., Stone, P., Beainy, F., et al.: Small-scale testbeds for connected and automated vehicles and robot swarms: Challenges and a roadmap. arXiv preprint arXiv:2503.05656 (2024), https://arxiv.org/abs/2503.05656
5. Eskandarian, A., Wu, C., Sun, C.: Crashworthiness optimization of vehicles using multi-objective particle swarm algorithms. International Journal of Vehicle Design **51**(1/2), 98–116 (2009). https://doi.org/10.1504/IJVD.2009.026862
6. Jin, Y.: Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation **1**(2), 61–70 (2011). https://doi.org/10.1016/j.swevo.2011.05.001
7. Kroese, D.P., Taimre, T., Botev, Z.I.: Handbook of Monte Carlo Methods. Wiley Series in Probability and Statistics, Wiley (2014). https://doi.org/10.1002/9781118014967
8. Lv, Z., Wang, L., Han, Z., Zhao, J., Wang, W.: Surrogate-assisted particle swarm optimization algorithm with pareto active learning for expensive multi-objective optimization. IEEE/CAA Journal of Automatica Sinica **6**(3), 838–849 (2019). https://doi.org/10.1109/JAS.2019.1911450

9. Zhang, W., Jiang, H., Wang, Z.: Multi-objective crashworthiness optimization of vehicle structures using a hybrid pso-ga algorithm. Structural and Multidisciplinary Optimization **50**(3), 543–556 (2014). https://doi.org/10.1007/s00158-014-1065-9