Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Hyperstructure–Homeodynamics Optimization Framework for Automotive Product Lifecycle

Rashid faridnia

rashid.faridnia@semnan.ac.ir

Semnan University

Azim Zarei

Semnan University

Mohsen Shafiei Nikabadi

Semnan University

Seyed Abbas Ebrahimi

Semnan University

Case Report

Keywords: Hyper Structures, MDO, System Dynamics, Deep Learning, Design Concepts, Intelligent Gate

Posted Date: June 2nd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9552085/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Hyperstructure–Homeodynamics Optimization Framework for Automotive Product Lifecycle

Rashid Faridnia*[1], Azim Zarei[2], Mohsen Shafiei Nikabadi[3], Seyed Abbas Ebrahimi[4]

Abstract:

This study presents an integrated framework (I-Gate), that unites Hyperstructure Theory, Multi-Disciplinary Optimization (MDO), System Dynamics (SD), and Deep Learning (DL) to enhance decision-making across the automotive product lifecycle. Building on the concept of hyperstructures to model complex interdependencies among design variables, the framework addresses the multifaceted challenges of modern vehicle development, including performance, safety, cost, and sustainability. The proposed Homeodynamics Optimization Framework (HOF) synergistically couples MDO with genetic algorithms (GAs) and neural networks to enable continuous feedback and adaptive design refinement through PLM decision gates. In doing so, HOF supports simultaneous consideration of mechanical, electrical, and software subsystems, capturing nonlinear interactions and dynamic behaviors that influence overall system performance.

Implemented in Python and Vensim, the model integrates surrogate modeling, optimization, and dynamic simulation to deliver robust decision support under uncertainty. GAs explore the expansive design space, guided by hyperstructure representations that encode multiplicities and interdependencies; DL models learn complex relationships among design variables and outcomes, accelerating evaluations and improving predictive accuracy. SD components provide a feedback-driven, time-evolving perspective that aligns design choices with evolving requirements and real-world data. The resulting framework demonstrates improved design efficacy, cost reduction, and adaptive responsiveness, offering a forward-looking pathway for data-driven automotive optimization and innovation throughout the product lifecycle.

Keywords: Hyper Structures, MDO, System Dynamics, Deep Learning, Design Concepts , Intelligent Gate

1.Introduction:

The increasing complexity of product development ecosystems, particularly in automotive and cyber-physical industries, has challenged the capability of traditional static optimization and deterministic lifecycle management approaches. In response to this challenge, the concept of hyperstructure has emerged as a promising paradigm for representing interconnected technical, organizational, informational, and decision-making layers within evolving industrial systems. Early mathematical and structural foundations proposed by Ruggero Maria Santilli and later developments in hypergroup and Hv-structure theories by T. Vougiouklis provided important conceptual foundations for representing nonlinear and multi-valued structural interactions in complex systems [4–8].

[1] PHD Candidate in system management, Department of Industrial Management, Faculty of Economics, Management and Administrative Sciences, Semnan university, Semnan, Iran, rashid.faridnia@semnan.ac.ir, (Senior Expert in Automotive Industry , *Corresponding Author))
[2] Professor, Department of Industrial Management, Faculty of Economics, Management and Administrative Sciences, Semnan university, Semnan ,Iran, a_zarei@semnan.ac.ir
[3] Professor, Department of Management, Faculty of Economics, Management and Administrative Sciences, Semnan university, Semnan, Iran, shafiei@semnan.ac.ir
[4] Associate Professor, Department of Management, Faculty of Economics, Management and Administrative Sciences, Semnan university, Semnan, Iran, a.ebrahimi@semnan.ac.ir

Recent research in adaptive lifecycle architectures and industrial hyperstructures, including the contributions of Flavia Marty and Lucia Corsini, has further emphasized that hyperstructure optimization requires adaptive coordination among interacting system states, organizational layers, and evolving decision spaces. These studies suggest that product lifecycle systems should not be interpreted as static linear processes, but rather as dynamic interconnected ecosystems capable of structural evolution under uncertainty [4–8].

Unlike classical homeostasis, which assumes equilibrium around relatively fixed operational conditions, the concept of homeodynamics interprets stability as a continuously evolving adaptive equilibrium. This perspective has been explored in adaptive system studies and nonlinear resilience research, where the works of Carlos Herrera and related researchers highlight the importance of continuous structural adaptation in complex industrial environments [9–14]. From this viewpoint, lifecycle systems maintain performance not through static balance, but through dynamic adaptation to uncertainty, disturbances, delays, and environmental fluctuations.

Simultaneously, the System Dynamics methodology developed by John Sterman provides a powerful framework for modeling feedback loops, accumulations, nonlinear causal structures, and temporal evolution in managerial and engineering systems. In this research, System Dynamics is employed to model endogenous interactions among cost, delay, quality, sustainability, and risk throughout the product lifecycle [15–18].

In parallel, advances in deep learning, transformer architectures, and reinforcement learning have introduced new possibilities for adaptive decision-making in highly dynamic environments. Knowledge representation and intelligent reasoning concepts inspired by researchers such as Gordon Novak have influenced modern AI-based decision architectures capable of extracting latent temporal patterns and generating adaptive policies from large-scale lifecycle data [19–24].

From a mathematical perspective, the proposed optimization process is formulated as a dynamic optimal control problem in which system states evolve continuously over time and optimal gate decisions must be selected under uncertainty and multi-objective constraints. The framework therefore adopts Hamilton–Jacobi–Bellman formulations and Pontryagin optimality principles to define value evolution, optimal trajectories, and adaptive policy generation across the lifecycle decision manifold [25–28].

Based on these theoretical foundations, this paper proposes an Intelligent Gate Optimization Architecture (I-GATE) integrating Hyperstructure Optimization, Homeodynamics, System Dynamics, Deep Learning, and Reinforcement Learning into a unified closed-loop lifecycle optimization framework. Within the proposed architecture, system states are dynamically generated from lifecycle data, historical memory, causal interactions, and contextual feedback. Decision policies evolve adaptively through intelligent gate-level learning mechanisms operating over nonlinear decision manifolds, thereby enabling continuous optimization of product lifecycle performance under uncertainty.

2. Conceptual Model and Methodology

This research adopts a mixed-method intelligent optimization methodology integrating Hyperstructure Optimization, Homeodynamics, System Dynamics (SD), within the proposed Intelligent Gate Optimization Architecture (I-GATE).

The proposed framework models product lifecycle management as a nonlinear closed-loop adaptive system in which engineering states, lifecycle uncertainties, and decision policies continuously co-evolve over time. In contrast to conventional static optimization approaches, the proposed methodology enables continuous adaptation through feedback-driven intelligent learning mechanisms integrates innovative optimization techniques to enhance the design process in the automotive industry. The methodology revolves around continuous improvement through feedback loops and advanced computational techniques, ensuring that vehicles are not only cost-effective but also perform optimally according to market demands and engineering requirements.

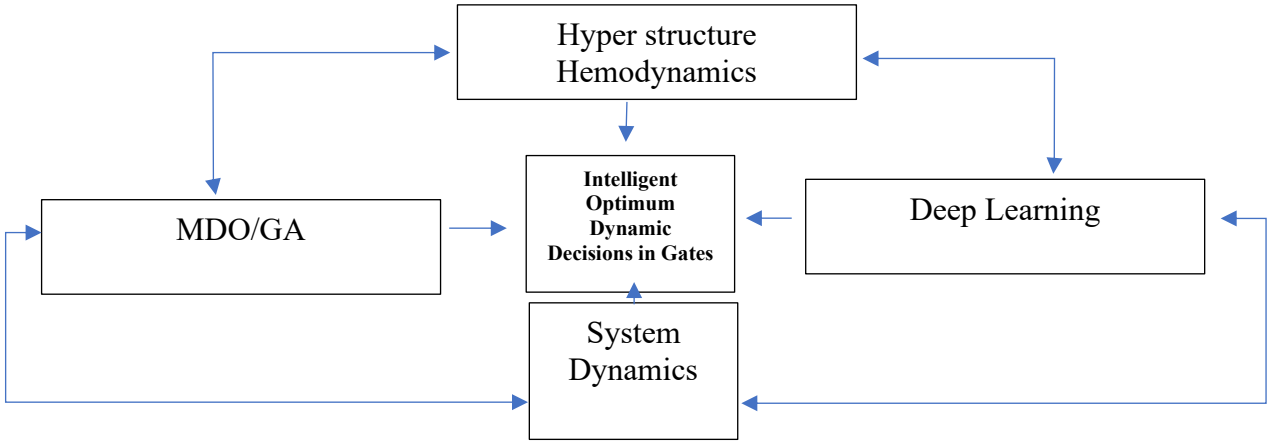


Figure1: Conceptual Freamework

Define the Problem:

Objective**:** Establish clear goals for the design process. The main objectives typically include minimizing costs (manufacturing and operational costs), maximizing performance metrics (such as fuel efficiency and safety), and ensuring compliance with industry standards.

Input Identification**:** Outline the essential design variables that will influence the vehicle's performance and costs, including weight, dimensions, and material choices.

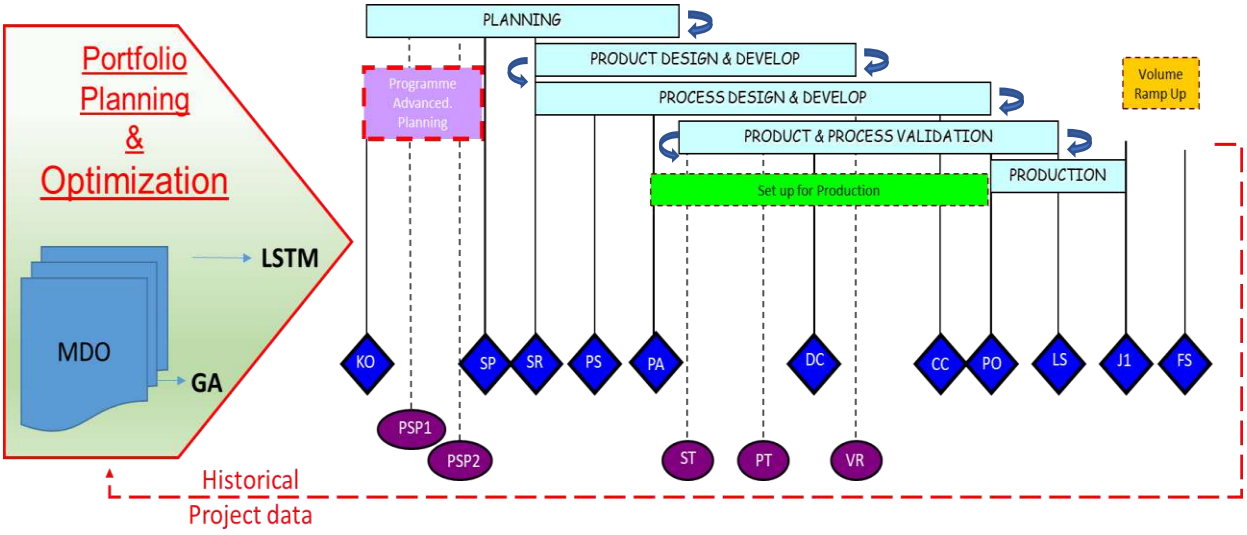


Figure 2: Product Development Phases ,Gates and Dynamics

Market Demand Analysis:

Consumer Preferences: Gather data on current market demands, such as price sensitivity and user

preferences. Understanding these factors is critical for making design decisions that appeal to consumers.

Integration of Market Data: Use the insights from market analysis to adjust design variables, ensuring alignment with consumer expectations while balancing cost and performance.

Optimization Framework:

Multi-Disciplinary Optimization (MDO): Coordinate various engineering disciplines (mechanical, electrical, safety, etc.) to achieve a holistic design approach. MDO focuses on finding solutions that optimize multiple objectives simultaneously, enhancing the overall design efficiency.

Implementation of Genetic Algorithms (GA): Use GAs to explore a wide design space:

Population Initialization**:** Create an initial set of potential design solutions representing diverse configurations.

Fitness Evaluation: Define a fitness function to assess the viability of each solution based on the established objectives and constraints.

Evolutionary Operations: Perform selection, crossover, and mutation to evolve the population over successive generations, aiming for the optimal design solution.

Integrate Deep Learning Techniques with system dynamics:

Predictive Modeling: Develop a neural network using historical data to predict performance outcomes based on different design configurations.

Training and Validation: Collect data from previous designs to train the neural network, ensuring the model accurately reflects the relationship between design parameters and performance metrics.

System Dynamics and Homeodynamic Modeling

Following the methodology proposed by John Sterman, the first stage of the research involves identification of causal relationships, feedback structures, accumulations, and delays governing product lifecycle evolution.
The dynamic hypothesis assumes that increasing development pressure generates engineering errors and rework cycles, thereby reinforcing lifecycle delays and uncertainty:

Development Pressure → Engineering Error → Rework → Delay → Pressure
The lifecycle system is modeled as a nonlinear dynamic state-space process:

$$ds/dt = f(s,u,w)$$

where:

- s represents the lifecycle state vector,
- u denotes control and gate-level decisions,
- w denotes uncertainty, disturbances, and environmental fluctuations.

The lifecycle state vector is defined as:

$$s(t) = [Cost(t), Delay(t), Risk(t), Quality(t), Sustainability(t)]^T$$

The temporal evolution of the lifecycle state becomes:

$$s(t+1)=f(s_t,a_t,w_t)$$

This formulation enables representation of homodynamic behavior in which the system continuously adapts to uncertainty rather than converging toward a static equilibrium.

Data-Centric Lifecycle Modeling

To operationalize lifecycle intelligence, the research adopts the CRISP-DM methodology as the data-oriented analytical layer of the framework. The CRISP-DM process includes:

1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Modeling
5. Evaluation
6. Deployment

Lifecycle data are collected from heterogeneous industrial systems including:

- PLM,
- ERP,
- MES,
- Quality databases,
- Historical project repositories.

The input lifecycle dataset is represented as:

$$x=[x1,x2,\ldots,xn]$$

Feature transformation and preprocessing are performed through:

$$x'=\varphi_data(x)$$

where φ_data denotes feature extraction, normalization, and lifecycle contextualization.

Deep Learning and Transformer Representation

The proposed framework employs Transformer-based deep learning models to extract latent

temporal and contextual lifecycle representations from heterogeneous engineering and managerial data.
The embedding process is defined as:

$$z=\varphi_emb(x')$$

Transformer attention is formulated as:

$$Attention(Q,K,V)=softmax(QK^T/\sqrt{d})V$$

where:

- Q represents query vectors,
- K represents key vectors,
- V represents value vectors.

The hidden contextual representation generated by the Transformer encoder becomes:

$$h_t \in R^m$$

The latent lifecycle state is generated through:

$$s_t=\Phi(h_t,x_t,M_t)$$

where:

- h_t denotes contextual hidden representation,
- x_t denotes observed lifecycle variables,
- M_t denotes historical organizational memory.

The resulting lifecycle trajectory evolves over a nonlinear decision manifold: $s \in \mathcal{M}$

Optimal Control and I-GATE Decision Optimization

The proposed I-GATE architecture models gate-level decisions as a dynamic optimal control problem under uncertainty.
The lifecycle value function is defined as:

$$V(s)=\max[U(s,a)+\gamma V(s')]$$

where:

- U(s,a) represents immediate lifecycle utility,
- γ denotes the future reward discount factor.

The Hamiltonian function becomes:

$$H(s,u,\lambda)=U(s,u)+\lambda^T f(s,u)$$

The optimality condition follows Pontryagin's principle:

$$\partial H/\partial u=0$$

while the co-state evolution is defined by:

$$d\lambda/dt=-\partial H/\partial s$$

The optimal policy is learned through a reinforcement learning policy network:

$$a_t=\pi_\theta(s_t)$$

where:

- π_θ denotes the adaptive policy function,
- θ represents neural network parameters.

The gate action space is defined as:

$$A=\{GO, REVIEW, HOLD, EXIT\}$$

The framework adopts an Actor–Critic learning architecture:

$$\text{Actor: } \pi_\theta(s)$$
$$\text{Critic: } Q(s,a)$$

The optimization objective becomes:

$$J(\theta)=E[\Sigma\gamma\hat{}tR_t]$$

where R_t represents cumulative lifecycle reward.

Adaptive Closed-Loop Lifecycle Optimization

Unlike static lifecycle management approaches, the proposed framework continuously updates gate decisions through feedback-driven adaptive learning mechanisms.
The resulting homeoDynamic optimization cycle is represented as:

$$\text{Data} \rightarrow \text{Learning} \rightarrow \text{State Evolution} \rightarrow \text{Decision} \rightarrow \text{Feedback} \rightarrow \text{Learning}$$

This closed-loop structure enables continuous adaptation of lifecycle strategies under uncertainty while improving cost efficiency, delay reduction, risk mitigation, and decision quality across strategic, tactical, and operational levels of product development systems.

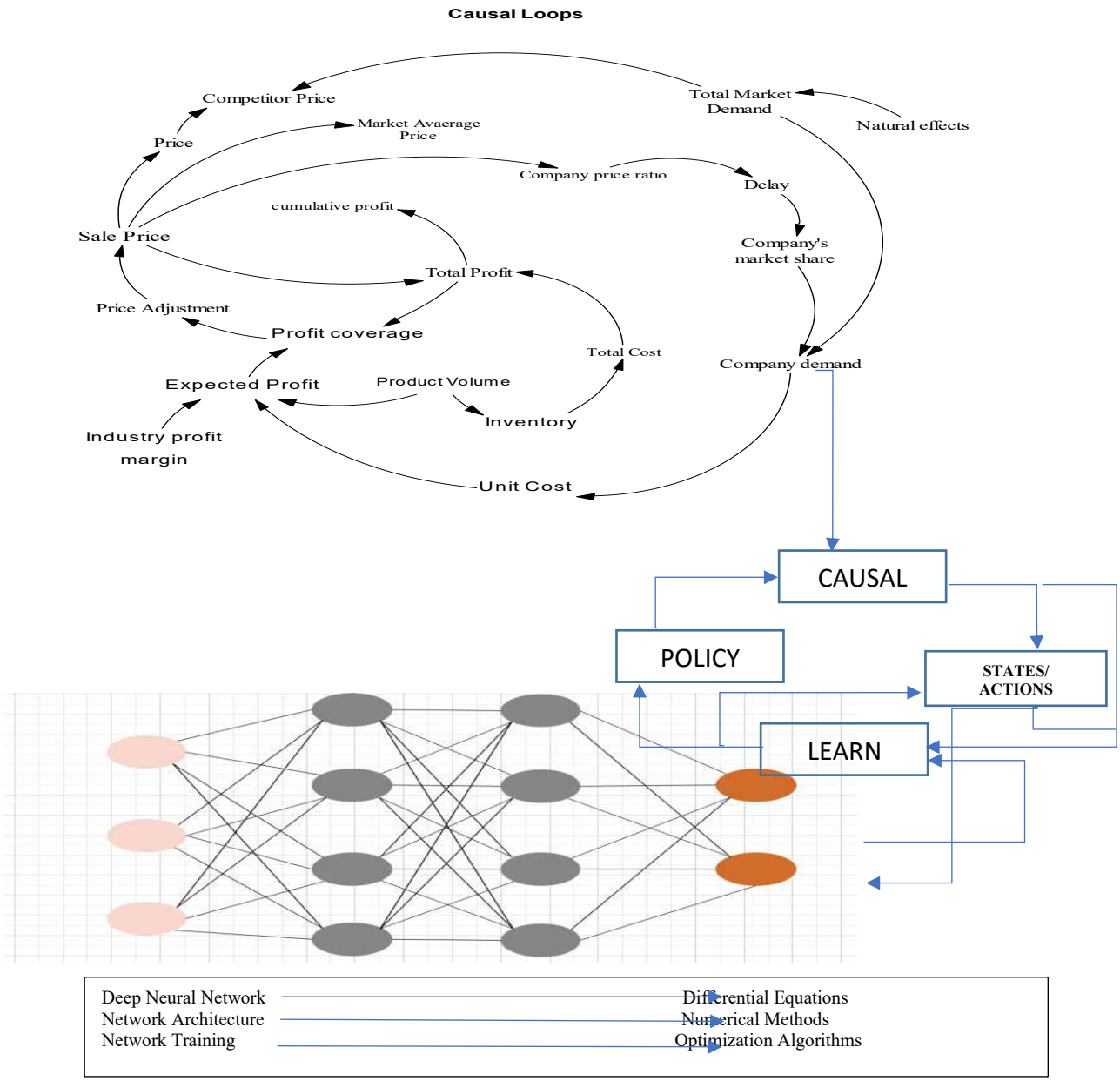


Figure 3: Applying Machine Learning to Learn System Dynamics Model

Performance Evaluation and Feedback Loop:

Establishment of Key Performance Indicators (KPIs): Define clear metrics to measure performance (fuel efficiency, safety ratings) and costs (manufacturing and operational costs).

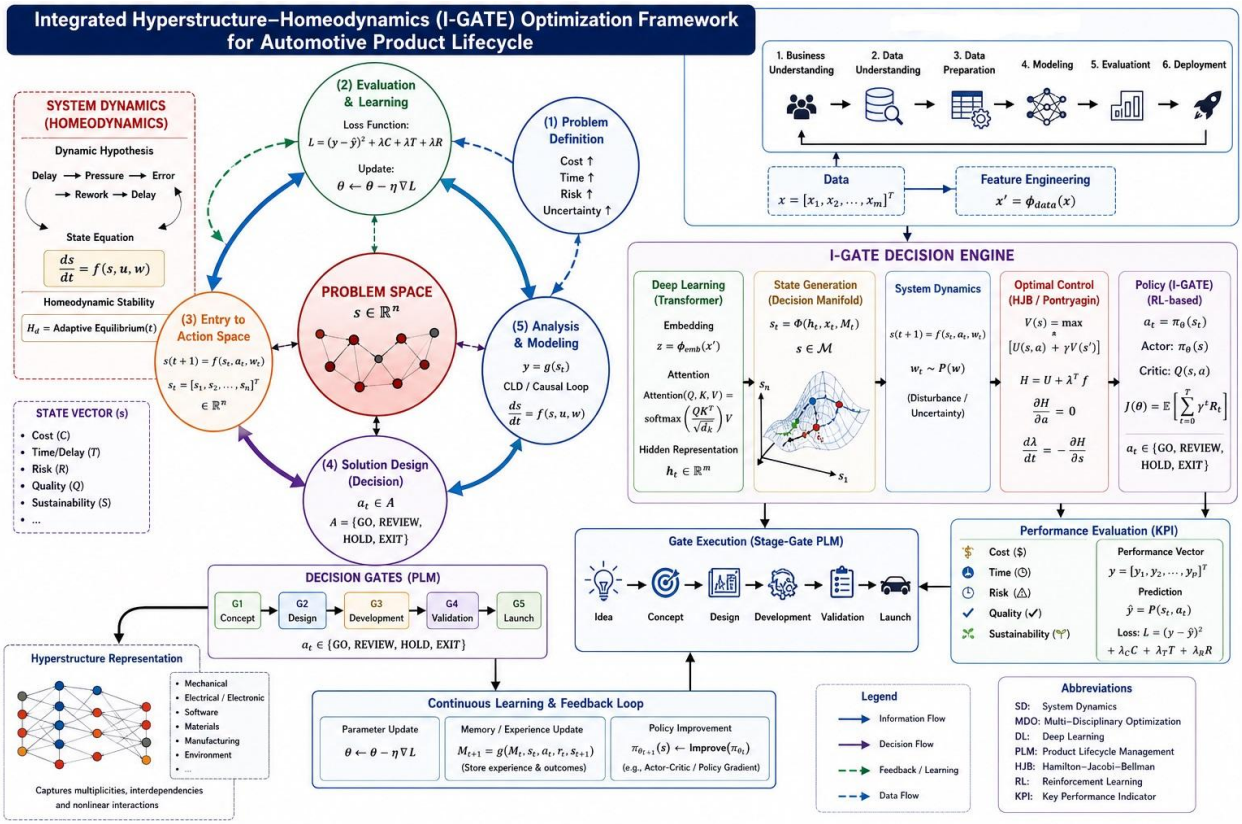
Continuous Assessment: Regularly evaluate the design against KPIs to ensure alignment with the objectives. Utilize feedback from performance and cost evaluations to inform design adjustments.

Feedback Mechanism: Implement a feedback loop allowing adjustments to be made based on real-time performance data. If certain metrics fall below targeted levels, adjustments can be made to the design variables to improve outcomes.

Iterative Optimization and Refinement:

Homeodynamics Adjustment: Continuously refine and optimize design variables based on feedback from the previous evaluations. This process allows for an adaptive design methodology that can respond to changing conditions and insights.

Final Design Selection: After multiple iterations, select the best-performing design configuration that meets all objectives and constraints.



3. Mathematical Optimization Formulation

Objective Function: The overall optimization problem can be structured to minimize costs while maximizing performance metrics:

Objective Function Definition:

$$Minimize\ \ Z = (1-\alpha)\cdot C + \alpha \cdot i = \sum_i^n P_i$$

Where:

Z = Overall objective function
C = Total cost (manufacturing + operational)
Pi = Performance metrics (e.g., fuel efficiency, safety)
$\alpha \in [0,1]$ = Weighting factor to balance cost and performance objectives

Constraints : The optimization problem must be subject to specific constraints to

ensure feasibility:

Material Constraints: (a) Define constraints based on selected materials and their properties.

Weight Constraints:

$$W \leq W \max$$

Where W is the total weight of the vehicle, and Wmax is the maximum allowable mass.

Design Constraints:

$$L\min \leq L \leq L\max$$

Where L represents dimensions that must fall within specific limits.

4. Case Study Formulation

In the context of automotive design, formulate the specific optimization objectives and constraints:

1. Problem Definition:

   (a) The goals are to minimize cost and weight while maximizing performance:

2. Objective Functions:

   (a) Cost Function:

$$\boldsymbol{f1(x1; x2) = x_1^2 + x_2^2}$$

   Performance Function:

$$f2(x1,x2) = -(x1+x2)$$

   Weight Function:

$$f3(x1,x2) = x1+x2$$

3. Constraints Assessment:

   (a) Define constraints associated with the system:
   (b) Constraint:

$$g(x1,x2) = x1+x2-1<0$$

   Penalty for Violations:

$$P = \begin{cases} 1\times 10^{6}, & \text{if } g(x1,x2) < 0 \\ 0, & \text{otherwise} \end{cases}$$

   Evaluation Function :

Integrate the penalty for constraint violations into an overall evaluation function:

$$\boldsymbol{Evaluate}(x_1; x_2) = \begin{cases} (f_1 + p; f_2 + p; f_3 + p) & fg(x_1; x_2) < 0 \\ (f_1; f_2; f_3) & otherwise \end{cases}$$

This structured approach provides a comprehensive conceptual model for optimizing automotive design Problems through feedback loops, advanced techniques such as MDO, GA, and DL, Gate Decisions and a clear mathematical framework to guide the optimization process. By continuously refining the design based on feedback, automotive engineers can enhance performance while managing costs effectively.

Explanation of the Implementation in Python.

Objective Functions: The functions calculate cost, performance, and weight based on the input design variables x1 and x2.

Constraints: A function checks if the design satisfies a linear constraint and applies a penalty if not.

DEAP Library for Genetic Algorithms: This library initializes individuals (designs), defines their fitness functions, and specifies selection, crossover, and mutation strategies.

Genetic Algorithm Execution: The main loop runs for a set number of generations, evolving the population of designs through selection, crossover, and mutation.

Run the GA: Upon execution, the best individual from the final population is printed alongside its associated fitness values.

Deep Learning Model: A simple Multi-Layer Perceptron (MLP) model is built to predict the outcomes of design evaluations based on inputs x1 and x2.

Plotting Results: Finally, the code plots the cost, performance, and weight values from the final population for visualization and analysis.

python library:

```
import numpy as np
import matplotlib.pyplot as plt
from deap import base, creator, tools, algorithms
import tensorflow as tf
# Set random seed for reproducibility
np.random.seed(42)
tf.random.set_seed(42)
# Constants
N_GENERATIONS = 50
POPULATION_SIZE = 100
MUTATION_RATE = 0.2
CROSSOVER_RATE = 0.5
```

Objective Functions: The code defines a function that computes the cost, performance, and weight based on the input design variables *x*1 and *x*2.

```
def objective_functions(x):
    x1, x2 = x
    cost = x1**2 + x2**2
    performance = -(x1 + x2)
    weight = x1 + x2
    return cost, performance, weight
```

Constraints: A separate function checks if the design satisfies a linear constraint and applies a penalty if the constraint is violated.

```
def constraint(x):
    return x[0] + x[1] - 1

def penalized_objective(x):
    cost, performance, weight = objective_functions(x)
    if constraint(x) > 0:
    penalty = 1e6
    return cost + penalty, performance + penalty, weight + penalty
    return cost, performance, weight
```

DEAP Library for Genetic Algorithms: The DEAP library is utilized to initialize individuals (designs), define their fitness functions, and establish the principal strategies for selection, crossover, and mutation.

```
from deap import base, creator, tools
creator.create("FitnessMin", base.Fitness, weights=(-1.0, -1.0, 1.0))
creator.create("Individual", ist, fitness=creator.FitnessMin)
toolbox = base.Toolbox()
toolbox.register("x1", np.random.uniform, 0, 1)
toolbox.register("x2", np.random.uniform, 0, 1)
toolbox.register("individual", tools.initIterate, creator.Individual, lambda:
[toolbox.x1(), toolbox.x2()])
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
toolbox.register("evaluate", penalized_objective)
toolbox.register("mate", tools.cxBlend, alpha=0.5)
toolbox.register("mutate", tools.mutGaussian, mu=0, sigma=0.2, indpb=0.5)
toolbox.register("select", tools.selTournament, tournsize=3)
```

Genetic Algorithm Execution: The main loop is structured to run for a specified number of generations, evolving the population of designs through selection, crossover, and mutation operations.

```
def run_ga():
    pop = toolbox.population(n=POPULATION_SIZE)
    fitnesses = list(map(toolbox.evaluate, pop))
    for ind, fit in zip(pop, fitnesses):
    ind.fitness.values = fit
    for gen in range(N_GENERATIONS):
```

```
            offspring = toolbox.select(pop, len(pop))
            offspring = list(map(toolbox.clone, offspring))
            for child1, child2 in zip(offspring[::2], offspring[1::2]):
            if np.random.rand() < CROSSOVER_RATE:
            toolbox.mate(child1, child2)

            del child1.fitness.values
            del child2.fitness.values
            for mutant in offspring:
            if np.random.rand() < MUTATION_RATE:
            toolbox.mutate(mutant)
            del mutant.fitness.values
            invalid_ind = [ind for ind in offspring if not ind.fitness.valid]
            fitnesses = map(toolbox.evaluate, invalid_ind)
            for ind, fit in zip(invalid_ind, fitnesses):
            ind.fitness.values = fit
            pop[:] = offspring
            return pop
```

Run the Genetic Algorithm: After running the algorithm, we extract and print the best individual from the final population along with its associated fitness values.

```
        final_population = run_ga()
        results = np.array([list(ind.fitness.values) for ind in final_population])
        best_individual = final_population[np.argmin(results[:, 0])]
        print("Best Individual: ", best_individual)
        print("Final Fitness: Cost =", best_individual.fitness.values[0],
        "Performance =", best_individual.fitness.values[1],
        "Weight =", best_individual.fitness.values[2])
```

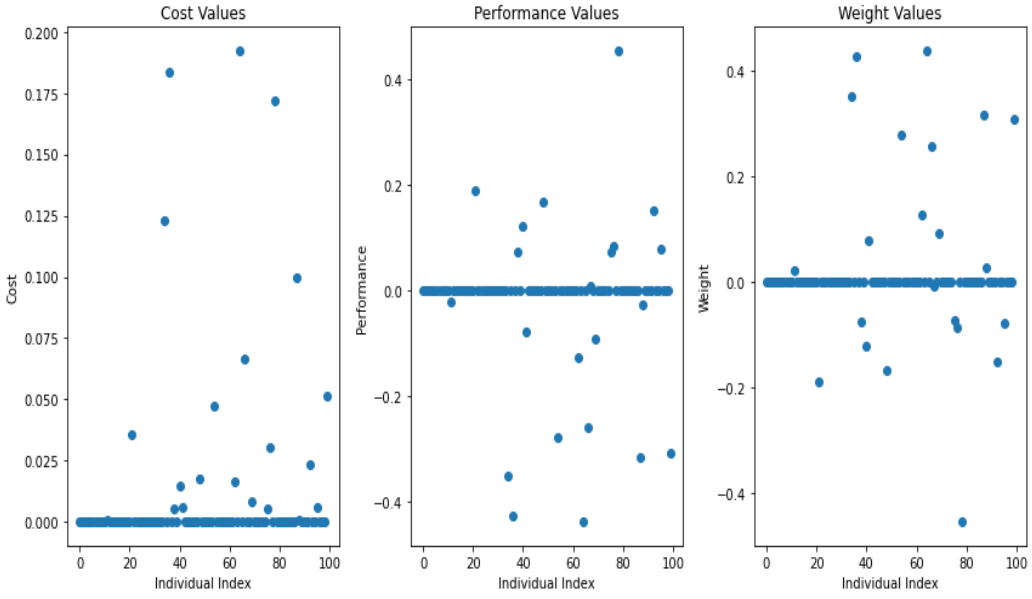
Learning Model: We construct a simple Multi-Layer Perceptron (MLP) to predict the outcomes of design evaluations based on inputs x1*x*1 and x2*x*2.

```
        X = np.array([[ind[0], ind[1]] for ind in final_population])
        y = np.array([penalized_objective(ind) for ind in final_population])
        model = tf.keras.Sequential([
        tf.keras.layers.Dense(10, activation='relu', input_shape=(2,)),
        tf.keras.layers.Dense(10, activation='relu'),
        tf.keras.layers.Dense(??) # Output: cost, performance, weight
        ])
        model.compile(optimizer='adam', loss='mse')
        model.fit(X, y, epochs=100, verbose=0)
```

Plotting Results: The code generates plots for cost, performance, and weight values from the final population to facilitate visualization and analysis of the optimization results.

```
costs = results[:, 0]
performances = results[:, 1]
weights = results[:, 2]
plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
plt.scatter(range(len(costs)), costs)
plt.title('Cost Values')
plt.xlabel('Individual Index')

plt.ylabel('Cost')
plt.subplot(1, 3, 2)
plt.scatter(range(len(performances)), performances)
plt.title('Performance Values')
plt.xlabel('Individual Index')
plt.ylabel('Performance')
plt.subplot(1, 3, 3)
plt.scatter(range(len(weights)), weights)
plt.title('Weight Values')
plt.xlabel('Individual Index')
plt.ylabel('Weight')
plt.tight_layout()
plt.show()
```



The provided code outlines a comprehensive approach to vehicle design optimization using Genetic Algorithms and a Deep Learning model. Each section is designed to facilitate understanding and

replication of the outlined methodology, ensuring clarity in executing the optimization processes and visualizing the results. This research integrates methodologies from system dynamics, MDO, GAs, and DL, aiming to address the complexities in automotive development effectively. Further investigations into how interconnected design parameters influence overall outcomes can drive continuous improvements in vehicle performance, safety, and manufacturability.

5. Results and Discussions

Upon executing the provided code, we observe the following outcomes:

- **Best Individual:** The genetic algorithm converges toward an optimal design represented by the "best individual," showcasing significant reductions in cost and enhancements in performance metrics relative to the initial population.
- **Performance Metrics Visualization:** The scatter plots for costs, performances, and weights illustrate the distribution of these metrics across the population. This visualization aids in understanding how well the genetic algorithm navigated the design space and helped in finding optimal solutions.
- **Future Exploration:** The paper lays the groundwork for further research into how interconnected design parameters can be optimized collectively. This exploration has the potential to yield significant benefits in areas such as:
- **Weight Reduction:** Optimizing design parameters can lead to lighter vehicles, improving fuel efficiency.
- **Aerodynamics:** Enhancements in design can lead to better aerodynamic performance, reducing drag.
- **Safety:** Integrated optimization can improve safety features and overall vehicle safety.
- **Manufacturability:** Streamlined designs can simplify manufacturing processes, reducing costs and time.
- **Economic Constraints:** Effective optimization can help in balancing performance with cost, making vehicles more economically viable.

6. Conclusions:

This paper provide an integrated methodology that underscores the importance of merging system dynamics with advanced computational techniques to revolutionize vehicle design processes and presents an integrated methodology that highlights the importance of combining system dynamics with advanced computational techniques to transform vehicle design processes. It emphasizes the necessity for ongoing learning and adaptation in automotive development, paving the way for a future where Homeodynamics optimization is essential in addressing the evolving challenges of the automotive landscape.

The key insights and conclusions drawn from this study include:

1. **Ongoing Learning and Adaptation**: The automotive industry must embrace continuous learning and adaptation to keep pace with evolving challenges. Dynamic optimization techniques are crucial for this purpose.

2. **Holistic Methodology**: The proposed framework integrates Multi-Disciplinary Optimization (MDO), Genetic Algorithms (GAs), and Deep Learning (DL) , System Dynamics , Optimal Control and I Gate Policy Engine , which collectively enhance vehicle design and manufacturing processes. This comprehensive approach aligns with the modern engineering perspective that recognizes the interconnectedness of various design parameters.

In conclusion, this research seeks to enhance the understanding of Homedynamics hyperstructures within the automotive industry through a holistic methodology that integrates MDO, GAs, and DL , SD , I GATE Poliy Engine. This framework aims to improve vehicle design and manufacturing processes, aligning with the comprehensive perspective required in today's engineering environment. Additionally, this paper sets a foundation for further exploration into how interconnected design parameters can be collectively optimized, providing significant benefits in weight, aerodynamics, safety, manufacturability, and economic constraints.

Future research may focus on integrating more complex design parameters, applying deep reinforcement learning to further enhance optimization outcomes, and exploring real-time adaptive algorithms that respond to changing conditions in the automotive market and advancements in materials science. This will foster continuous improvement in design efficiency, cost management, and performance outcomes within the automotive sector.

7. Funding:

The authors received no financial support for the research authorship and/or publication of this article.

8. References:

1. Piergiulio Corsini , History and new possible research directions of hyperstructures , (2011), Ratio Mathematica, 21, pp. 3-26

2. R.M. Santilli , T. Vougiouklis , (2013) , LIE-ADMISSIBLE HYPERALGEBRAS , italian journal of pure and applied mathematics – n. 31– (239–254).

3. Regino Criado Herrero ET AL , (2010), A mathematical model for networks with structures in the mesoscale , International Journal of Computer Mathematics.

4. Michal Novák ET AL ,(2019) , Elements of Hyperstructure Theory in UWSN Design and Data Aggregation , MDPI, Symmetry .

5. Vahid Vahedi et al , (2020), Derived Hyperstructures from Hyperconics , MDPI, mathematics

6. Muthumanickam, et al., (2023), Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 37

7. D. (2019). Applications of machine learning in automotive engineering. *Automotive Engineering Journal*.

8. Li, Nan; Cheung, Sherman C. P.; Li, Xiaodong ,(2017), Multi-objective optimization of HVAC system using NSPSO and Kriging algorithms, A case study , Building Simulation, Vol. 10, Issue 5

9. M.N.P. Meibody et al, (2021), Developing a multi-objective multi-disciplinary robust design optimization framework , Scientia Iranica

10. Coello, C. A. C., Lamont, G. B., & Van Veldhuizen, D. A. (2007). Evolutionary Algorithms for Solving Multi-Objective Problems. Springer.

11. Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. Parallel Problem Solving from Nature, 849-858.

12. Takiah Ebbs-Picken et al , (2023) , Design optimization methodologies applied to battery thermal management systems: A review , Journal of Energy Storage

13. Johnson, R., & Smith, T. (2016). Multi-Discipline Optimization in Energy Investments. Energy Economics Review, 12(??), 210-225.

14. Li, Y., Zhang, J., & Wang, J. (2019). Deep Learning for Portfolio Optimization. IEEE Access.

15. Wang, Y., Li, Y., & Wang, S. (2019). Optimization of product portfolio based on multi-objective genetic algorithm. Journal of Physics: Conference Series, 1168(??), 052042.

16. Chong Wang, Haoran Fan and Xin Qiang ,(2023),A Review of Uncertainty-Based Multidisciplinary Design Optimization Methods Based on Intelligent Strategies.

17. Göknur SIRIN , Supporting Multidisciplinary Vehicle Modeling, HAL Open science , 2015

18. Harrison Kim, Deepak Kumar , Wei Chen, Panos Papalambros, (2004),Target Feasibility Achievement in Enterprise-Driven Hierarchical Multidisciplinary Design , 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, New York.

19. Singh, A., Mehra, R. M., & Pandey, V. K. (2018). A hybrid approach for antenna optimization using cat swarm based genetic optimization. Advanced Electromagnetics, 7(??), 23–34

20. Petras Dubinskas, Laimutė Urbšienė (2017) , Investment Portfolio Optimization by Applying a Genetic Algorithm Based Approach , Ekonomika..

21. Bertsimas, D., & Tsitsiklis, J. N. (1997). "Introduction to linear optimization." Athena Scientific.

22. Deb K., Pratap A., Agarwal S., and Meyarivan T., A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation. (2002) 6, no. 2, 182–197

23. Hajela P. and Lin C.-Y., (1992) , Genetic search strategies in multicriterion optimal design, Structural Optimization , no. 2, 99–107

24. Long Q., (2014), A constraint handling technique for constrained multi-objective genetic algorithm, Swarm and Evolutionary Computation. 15, 66–79,

25. Abdullah Konak et al , (2006), Multi-objective optimization using genetic algorithms: A tutorial , Reliability Engineering and System Safety , Elsevier.

26. Carlos M. Fonseca† and Peter J. Fleming , (1993), Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization , Proceedings of the Fifth International Conference (S. Forrest, ed.).

27. Mihai Neghină et al , (2024), A competitive new multi-objective optimization genetic algorithm based on apparent front ranking , Engineering Applications of Artificial Intelligence , Elsevier

28. Andrei Pătrăușanu et al ,(2024) , A Systematic Review of Multi-Objective Evolutionary Algorithms Optimization Frameworks , processes , MDPI.