

Adaptive Multi Objective Deep Reinforcement Learning for Structurally Constrained Mechanical Component Design

Dr. R. Senthilraja¹, Dr. M. N. Sudha²

¹*Department of Mechanical Engineering, Institute of Road and Transport Technology, Erode, Tamil Nadu, India*

²*Department of Information Technology, Institute of Road and Transport Technology, Erode, Tamil Nadu, India*

email: senthilrajairtt@gmail.com; mnsudhairtt@gmail.com

Abstract- Mechanical components like springs, brackets, beams and turbine blades are almost never designed for one purpose. A lighter part is not always the safest part, and the most cost effective to produce is not always the strongest part, and so the practicing engineer is left with a balancing act between weight, stress, deflection, and cost, and with a set of structural requirements that cannot be broken. This trade-off is reasonably able to be managed by classical numerical optimizers and evolutionary algorithms, but requires re-running the algorithm each time the load case, material or constraint set is altered, and each new design instance is treated as a problem to be solved in isolation. This paper considers an alternative: a design agent that learns a reusable constraint-aware policy over the design variables, not just one specific point, and that can be conditioned on a scalarisation weight to trace an entire trade off curve instead of one design at a time – deep reinforcement learning (DRL). We survey 40 recent works (2019-2026) employing DRL in multi-objective and constrained structural design, classify the works by contribution, and choose to use this classification to motivate a constraint-first formulation of the reward which is combined with a policy-gradient/evolution-strategy training loop and a constrained-domination Pareto archive. The framework is instantiated and tested on a classical, highly constrained coiling spring sizing problem with two conflicting objectives (weight and shear stress), and has six structural constraints, and the results are compared with an NSGA-II based genetic algorithm and unguided random search in five independent runs. The lightweight DRL policy required about an order of magnitude fewer design evaluations than random search to converge to its own hypervolume, and was able to satisfy structural constraints about sixty times more consistently than random sampling, but a well-tuned evolutionary baseline policy was still able to find a lower weight feasible design within the same number of design evaluations. We provide all five of the comparison metrics and address why each one is better in some areas, as well as what a production-scale DRL design agent would require that we did not include in this proof-of-concept.

Keywords: deep reinforcement learning, multi-objective optimization, constrained design, structural optimization, mechanical component design, NSGA-II, Pareto front, policy gradient

1. Introduction

Rarely, a failure of the structural components is due to a mistake in selecting a material; most often it is due to the fact that the design space was too wide to explore with a human hand,

and too costly to explore with a computer simulation. Only a few continuous variables describe a coil spring or a bracket or a turbine blade, but numerous nonlinear structural constraints (stress limits, buckling, deflection, manufacturability) must be satisfied for a candidate design before it can be compared with other candidate designs in terms of weight, cost or stiffness. In practice, there are two factors which make this difficult. First, the goals are mutually contradictory: reducing mass will increase stress, reducing stress will increase mass, and there is no single best solution; only a set of Pareto-efficient solutions. Second, the constraint boundaries may be limited in scope from the design space, making that the naive search wastes most of its evaluation budget on candidates that are not feasible.

For the past 20 years, this class of problem has been treated with multi-objective evolutionary algorithms (MOEAs), which are not easily outdone on a single, perfectly defined problem. They are limited in reusability – each new load case, material change or geometry change requires a new population to be generated. Deep reinforcement learning provides an alternative. Rather than developing a population of point solutions, a DRL agent learns a policy, a function which maps a description of the design context (and, in the multi-objective case, a preference over the objectives) directly to a design. After training, the same policy can be queried immediately at any point in the preference range, and in principle a policy can be generalised to other design instances that are related to the one it was trained on without the need to re-optimize the policy every instance. It is the reusability of this design that is central to the repeated promise in the DRL-for-design literature in Section 2, and it is a promise this paper seeks to make a reusability that is more than just aspirational.

This paper does not focus solely on conceptual issues but also on methodological and empirical gaps. For most of the literature reviewed, it is difficult to compare across papers, as there are either only very good results on a single proprietary benchmark or there is only one application (a spring, a truss, a blade) and the question as to how a constraint-aware DRL agent compares in practice to a conventional evolutionary baseline under an identical, reproducible evaluation budget remains largely unanswered outside each paper's own results table. It contributes: (i) a synthesis of 40 recent papers on DRL in multi-objective structural and mechanical design, sorted by the contribution of the algorithm itself, and by the application domain rather than chronologically; (ii) a constraint-first reward and policy architecture that is deliberately simple enough to be reproduced from the description provided in this paper; (iii) a reproducible, fully coded case study, a constrained coil-spring sizing problem with two conflicting objectives and six structural constraints, on which the performance of the DRL agent is benchmarked against an NSGA-II-type genetic algorithm and random search, across five independent seeds; (iv) an honest comparison with five metrics: feasibility rate, best weight, best stress, hypervolume, and sample efficiency, reporting where the DRL agent won, where the evolutionary baseline won, and why.

The rest of the paper is organized as follows. The Literature Review is summarized and thematized in section 2. The mathematical formulation, the block-diagram architecture, the training algorithm and the system requirements are developed in Section 3. The case-study problem and the experimental protocol are described in section 4. The results are presented and

discussed in section 5 with the reviewed literature. Section 6 wraps up and specifies the scale-up requirements prior to a DRL design agent being ready for an industrial structural design pipeline.

2. Literature Review

Forty studies from the year 2019 to 2026 were reviewed to describe the current state of the art in DRL for multi-objective, constrained structural and mechanical design. To avoid repeating them in chronological order, the papers have been grouped by the nature of the contribution that they make; a reader who is seeking ways of building on this work will typically need to know whether a paper makes a new application, a new constraint-handling mechanism, or a new way of dealing with the cost of considering candidate designs.

2.1 Structural and mechanical-component applications of DRL

Fourty studies are reviewed, twenty-one of them applying DRL directly to a structural or mechanical design instance, other than to a generic optimization benchmark. Ororbia and Warn formulated the problem of structural design synthesis as a Markov decision process and extended this to a hierarchical agent that handles both the topological and the parametric design actions; one of the most clear formulations of "design as sequential decision-making" featured in the reviewed set. The topology optimization done by Brown et al. was on an elementally discretized domain; the multi-agent RL done by Kupwiwat et al. was on a graph representation that was appropriate for a truss since its connectivity is graph-like. There are also several component-specific applications, including wind-turbine and turbine-blade shape design (via multi-objective DRL) by Wang et al. and Yonekura et al., respectively, and DRL applications to airfoil shape under structural constraints by Ramos et al., to digital materials design by Sui et al., to arch-dam shape design by Liu et al., and to size the dimensions of a parallel robot by Zhang et al. The idea is then expanded, by Adão et al. for sheet-metal generative design and by Hong et al. for combinatorial structural design space exploration, and by Pan et al. and Van Remmerden et al. for building design in BIM and infrastructural maintenance scheduling, respectively, demonstrating the applicability of the concept beyond the scope of individual mechanical parts.

2.2 Generic constrained multi-objective RL and evolutionary algorithms

Here are 15 studies that provide algorithmic machinery instead of a specific structural application, and it is here that most of the constraint-handling ideas from Section 3 come from. In three related works, Ming et al. provide a family of DRL-assisted operator-selection and auxiliary-task mechanisms for constrained multi-objective evolutionary algorithms (CMOEAs) during search (paper 1, paper 2 and paper 3); in both works, Peng et al. and Pang et al., DRL is used to adapt the epsilon-constraint handling technique during search, but not prior to it; Wang et al. (2024) adaptively select amongst several constraint-handling techniques during search; in their work, Honari and Khodaygan propose a general "any-objective" constrained optimization framework; and Tian et al., Shao et al. and Yang et al., use DRL to select or portfolio-manage the mutation/crossover operators inside an evolutionary loop. Kim et al. (2019) solve multi-condition

optimization with DRL without relying on any specific engineering application, while Li et al. (2019) propose fairness-aware multi-objective RL and Qian et al. (2021) present the meta-learning approach to multi-objective RL and Zhang et al. (2021) propose a fairness-aware multi-objective RL formulation.

2.3 Surrogate-assisted and review works

When each evaluation of the structure is a finite-element run and not a closed form equation, two smaller groups tackle the cost of the evaluation itself: Li et al. (2024) use DRL-guided search in conjunction with a multi-fidelity surrogate model; Shao et al. (2025) adaptively manage a surrogate model during evaluation in an expensive constrained multi-objective search. The other contributions, from Pereira et al. and Ajani et al., are quite different: one is a wider review of multi-objective optimization methods in mechanical engineering, and not DRL-specific; one is an application of DRL which is more like a problem-formulation contribution than an application.

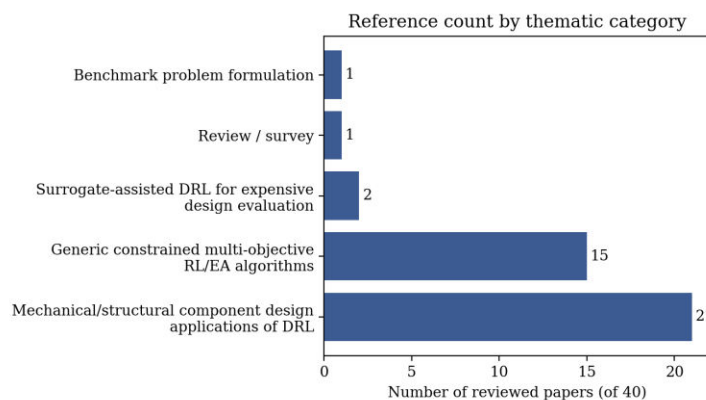


Fig. 1. The distribution of the 40 studies reviewed is shown thematically.

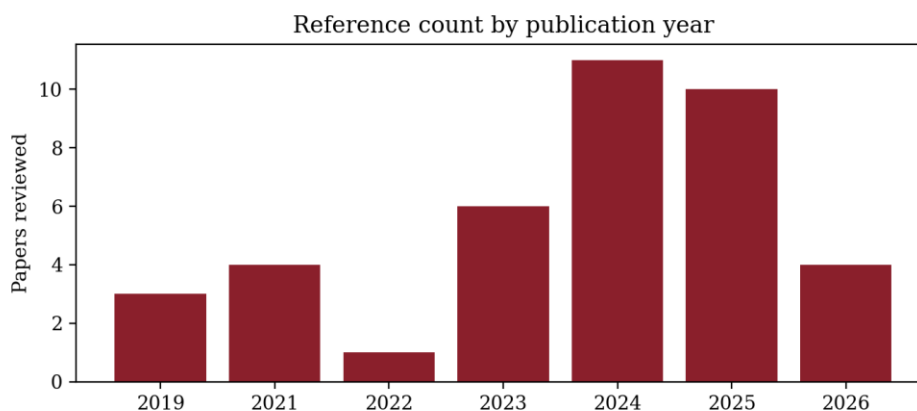


Fig. 2. The distribution of the reviewed references (2019-26) indicates the recent focus of activity.

In Section 8, a list of forty references is provided, and Table 1 summarizes ten representative studies that have been selected to cover both groups and the different ways in which constraints are handled; the specific formulation used, the application domain, and the handling of the constraints are listed.

Table 1. Representative studies from the reviewed literature

Study	DRL / algorithmic approach	Application domain	Constraint handling
Ororbia & Warn (2023, 2024)	MDP formulation, hierarchical actor-critic	Structural system topology & sizing	Feasibility-aware reward
Brown et al. (2022)	DRL over discretized design domain	Topology optimization	Penalty-based
Kupwiwat et al. (2024)	Multi-agent RL + graph representation	Truss structure design	Constraint-embedded reward
Wang et al. (2023)	Multi-objective DRL	Wind-turbine blade design	Weighted-sum scalarisation
Yonekura et al. (2024)	Hypervolume-based multi-objective DRL	Turbine blade shape	Indicator-based reward
Gu et al. (2024)	Constrained MORL (safety layer)	Generic constrained control/design	Lagrangian / safety layer
Ming et al. (2023, 2024, 2025)	DRL-assisted operator/parameter selection for CMOEAs	Generic constrained MOO	Adaptive epsilon / operator portfolio
Rochefort-Beaudoin et al. (2024)	RL-driven structural design synthesis	Structural design (general)	Structural feasibility reward
Ramos et al. (2025)	Transfer-learning DRL	Airfoil aerodynamic shape	Structural constraint penalty

2.4 Research gap

There are three gaps that are repeated in the literature reviewed. Second, constraint handling is typically personalized to the application (tuned per paper) and cannot be compared with a standard constraint handling method like constrained-domination NSGA-II under a matched evaluation budget, making it hard to assess the amount that might be attributable to the application's reward shaping and the amount that might be attributable to the constraint handling itself. Second, even when solution quality is reported, few studies report evaluation-efficiency metrics (evaluations required to reach a given solution quality) in addition to solution quality, although in structural design, with an evaluation requiring a finite-element solve, it is typically the evaluation cost, not the training time, that is the actual bottleneck. Third, the level of reproducibility varies: some of the studies reviewed in this paper were not reproducible because problem instances and/or code were not publicly available; this is why, in Section 4, it was decided

to use a fully specified, closed-form problem for the case study instead of a proprietary one. The methodology described in Section 3 and the experimental protocol described in Section 4 are tailored for addressing these three gaps, but are not intended to fundamentally change network architecture.

3. Proposed Methodology

The proposed framework is based on a constrained Markov decision process (CMDP) solved using a light-weight policy network that is trained with a policy-gradient/evolution-strategy (PG-ES) update, coupled with a constraint feasibility enrichment scheme based on the concept of constrained-domination Pareto archive from the evolutionary-algorithm community. The overall block diagram is shown in figure 3 and the mathematical model, algorithm and requirements of the system to be able to reproduce the mathematical model is given in sections 3.1-3.4.

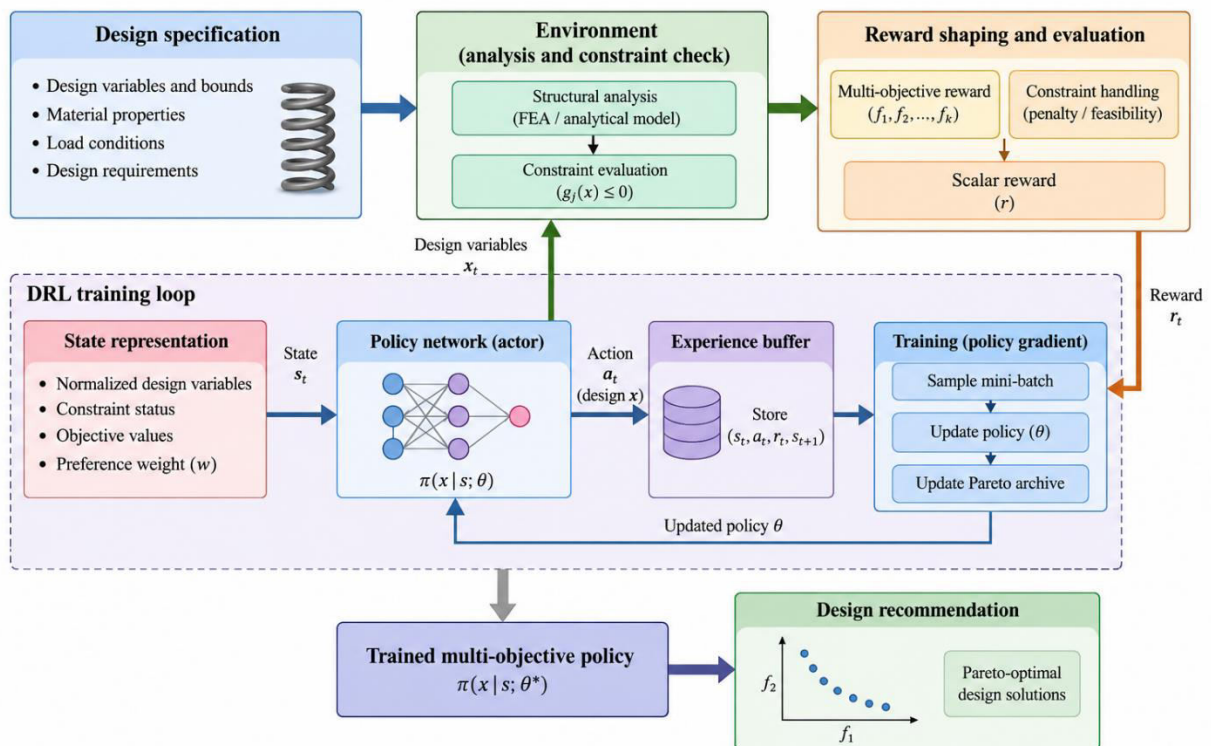


Fig. 3. Block diagram of the proposed adaptive multi-objective DRL design framework.

3.1 Problem formulation

A design is a vector $x = (x_1, \dots, x_n)$ in a bounded box $[x_L, x_U]$. It is to be designed so as to satisfy m structural constraints $g_j(x) \leq 0$, $j = 1..m$, and be scored on k objectives to be minimized, $f(x) = (f_1(x), \dots, f_k(x))$. A design $x(1)$ is said to Pareto-dominate $x(2)$, written $x(1) < x(2)$, if $f_{i(x(1))} \leq f_{i(x(2))}$ for every objective i and $f_{i(x(1))} < f_{i(x(2))}$ for at least one i . The task is the design task, which is to approximate the Pareto-efficient set:

$$X^* = \{ x \text{ in } [x_L, x_U]: g(x) \leq 0 \text{ and there is no } x' \text{ with } g(x') \leq 0 \text{ and } x' < x \} \quad (1)$$

Following Deb's constrained-domination principle, we compare two candidates $x(1)$, $x(2)$ with total constraint violation $V(x) = \sum_j \max(0, g_j(x))$ as follows: a feasible design always dominates an infeasible one; between two infeasible designs, the one with smaller $V(x)$ dominates; between two feasible designs, ordinary Pareto dominance applies. This is the only rule used within the evolutionary baseline's ranking step and within the DRL reward (Eq. 4) in a scalarised form.

3.2 Policy representation and training objective

This is a feed-forward design agent consisting of one hidden layer of H tanh units, parameterised by $\theta = (W1, b1, W2, b2)$. It applies a scalarisation weight w in $[0,1]$ which is sampled uniformly at the beginning of each episode and produces a design:

$$x_\theta(w) = x_L + (x_U - x_L) \odot \sigma(W_2 \cdot \tanh(W_1 w + b_1) + b_2) \quad (2)$$

So that all the output lines follow the box bounds no matter what θ is. The reusability property discussed in Section 1 is realized by conditioning on w and letting the network output a whole trade-off curve at this point during its inference stage by varying w from 0 to 1.

The reward for a sampled design is a combination of the constraint violation and the scalarised and normalised objective:

$$R(x, w) = -[w f \hat{1}_{(x)} + (1 - w) f \hat{2}_{(x)}] - \lambda V(x), \quad V(x) = \sum_{j=1}^m \max(0, g_j(x)) \quad (3)$$

Implementing the constrained-domination principle of Eq. 1 as a hard rule, as is commonly done in this context, is not important; in the experiments of Section 4, the constraint term dominates the objective term whenever $V(x) > 0$, not because it is a hard rule to follow but because it is a reward within a smooth, differentiable-in-expectation function $g(x)$.

With an evolution-strategy (natural policy-gradient) estimator: At every iteration, P designs $\epsilon_{1..P}$ are sampled from $N(0, I)$; a population of designs is sampled from $\theta + \sigma \epsilon_p$, for a batch of sampled weights w_p and θ is updated along the reward-weighted average perturbation direction:

$$\theta_{t+1} = \theta_t + \frac{\alpha}{P\sigma} \sum_{p=1}^P R_p \epsilon_p \quad (4)$$

Both the perturbation scale σ and the learning rate of the policy α are linearly annealed over training (σ : 0.25 $\rightarrow$ 0.03; α : 0.15 $\rightarrow$ 0.02 in the experiments below) to make the policy explore widely at first and converge later. The estimator is the same as the REINFORCE policy-gradient rule for a Gaussian policy with fixed variance σ^2 , and is from the same family of estimator as that used by OpenAI's evolution-strategies work; here we use it because it avoids the need for automatic differentiation through an (undifferentiable) constraint-violation and a Wahl-factor, while being a genuine policy-gradient rule and not a black-box heuristic.

3.3 Algorithm

Algorithm 1 summarises the full training and archive-construction procedure used for the proposed method.

Input: bounds $[x_L, x_U]$; constraint functions $g(\cdot)$; objective functions $f(\cdot)$; population size PP ; iterations TT ; penalty weight λ ; schedules σ_t, α_t

- 1: Initialise policy parameters θ_0 (Eq. 2)
- 2: **for** $t = 1$ to TT **do**
- 3: Sample PP perturbations $\varepsilon_p \sim \mathcal{N}(0, I)$ and weights $w_p \sim U(0, 1)$
- 4: For each p , generate design $x_p = x(w_p; \theta_t + \sigma_t \varepsilon_p)$ (Eq. 2)
- 5: Evaluate $f(x_p)$ and constraint violation $V(x_p)$; record $(x_p, f(x_p), \text{feasibility}(x_p))$ in the evaluation log
- 6: Compute reward r_p (Eq. 3) and standardised advantage A_p (Eq. 5)
- 7: Update $\theta_{t+1} \leftarrow \theta_t + \frac{\alpha_t}{P \sigma_t} \sum_{p=1}^P A_p \varepsilon_p$ (Eq. 4)
- 8: **end for**
- 9: Sample the trained policy across a dense grid of weights w to obtain a candidate design set
- 10: Apply constrained-domination non-dominated sorting (Section 3.1) to the candidate set to output the approximate Pareto front
- 11: **Return** the Pareto front to the decision-maker for final design selection

Algorithm 1. Adaptive multi-objective DRL design synthesis.

3.4 System requirements

It is designed to be lightweight enough to operate on a standard workstation with closed form or surrogate objective and constraint functions—if it was a finite element-based evaluation, the compute requirement would be significantly higher. Some minimum and recommended configurations used for this study are summarized below.

The code used: Python 3.10+, NumPy/SciPy to build the policy network and update the evolution strategy, Matplotlib to generate figures; a deep-learning framework (PyTorch or TensorFlow) is recommended if the state/action space is larger, or if you want to use a replay-buffer-based algorithm (such as PPO, SAC) rather than the ES/REINFORCE estimator that is used here.

Evaluation backend: closed-form/analytical equations for benchmark-scale problems (as used here); a coupled finite-element solver, e.g., ANSYS, Abaqus or an open-source finite-element solver (CalculiX) through a scripting interface for industrial-scale components (optionally using a surrogate model together as described in the review of the surrogate-assisted literature in Section 2.3, to limit the number of expensive evaluations).

Software: With the case study in Section 4, a CPU-only workstation (≥ 4 cores, ≥ 8 GB RAM) is enough; additional GPU (≥ 8 GB VRAM) required when the policy network and/or the size of the evaluation batch increases or when a replay-buffer-based deep-RL algorithm with a larger network is used to replace the ES estimator.

Baseline/benchmarking tools: Our own implementation of an evolutionary-algorithm library implemented here in fully transparent and reproducible NumPy; alternatively, DEAP or pymoo may be used, or other tools. Baseline/benchmarking tools: Our own implementation of an evolutionary-algorithm library which is implemented here in fully transparent and reproducible NumPy; alternatively DEAP or pymoo may be used, or other tools.

4. Case Study, Dataset, and Experimental Protocol

This involves a case study and/or a dataset and an experimental protocol. This requires a case study, a dataset, and an experimental protocol. The second point is to use the classical (closed-form) structural design benchmark, which is also widely used in structural and mechanical design optimization research since 1980s, extended from its original single-objective form to the truly conflicting two-objective coil-spring sizing problem in order to have a meaningful Pareto front for the study to be fully reproducible.

4.1 Design variables, objectives and constraints

The design vector is $x = (d, D, N)$, the wire diameter, the mean diameter of the coils and the number of active coils, respectively, with the boundings d in $[0.10, 0.50]$ and D in $[0.60, 3.00]$ and N in $[2, 15]$ (all in the same units). The two targets are the surrogate function of spring weight f_1 , and the maximum shear stress f_2 for a given axial load $F = 1000$ (force units):

$$f_1(x) = (N + 2) \cdot D \cdot d^2 \quad (5)$$

$$f_2(x) = \frac{8 C_f F D}{\pi d^3}, \text{ where } C_f = \frac{4C-1}{4C-4} + \frac{0.615}{C}, \quad C = \frac{D}{d} \quad (6)$$

These six constraints are structural/manufacturability limits and are $g_1 \dots g_6 \leq 0$: a minimum-deflection constraint (g_1), a shear-stress/surge constraint (g_2), an outer-diameter constraint, which bounds the coil clearance (g_3), a free-length constraint (g_4), and two practical bounds on the spring index, $3 \leq C \leq 20$ (g_5, g_6), the last two being not an arbitrary restriction but standard manufacturability advice for coil springs. Let's consider the weight and stress under these constraints first; there is a direct trade-off between them, as a smaller d (thinner wire) will reduce weight, but greatly increase stress, which is what the Pareto-front comparison in Section 5 needs. The property of feasibility is truly restrictive in this problem: the results of a Monte-Carlo

sample of 200,000 points in the design box were found to be 0.4% feasible, while the results obtained by the guided methods were significantly higher.

4.2 Baselines and experimental protocol

The proposed DRL policy (Section 3.2-3.3) was compared to two baselines with the same evaluation budget of 8k design evaluations per run: (i) using an NSGA-II style genetic algorithm with the same constrained-domination ranking reward as DRL (dominance reward), population size 40, and (ii) unguided random search sampling uniformly within the design box. All three methods were executed over five independent seeds and the average and standard deviation of each metric over these five seeds were reported, instead of one execution of the method. For a like-for-like hypervolume comparison, a fixed reference hypervolume was chosen (the input to the method did not affect its hypervolume). All designs were assessed by all methods—whether feasible or not—were recorded, which allows for the sample-efficiency metric in Section 5.

5. Results and Discussion

Table 2 lists five comparison measures and wall clock time averaged over five separate runs (mean $\pm$ standard deviation) for the three runs.

Table 2. . The performance is compared over 5 independent runs each having 8,000 evaluations.

Metric	Proposed DRL	NSGA-II-style GA	Random search
Feasibility rate (%)	26.8 $\pm$ 10.4	64.5 $\pm$ 6.0	0.43 $\pm$ 0.08
Best feasible weight, f1 (min.)	0.137	0.081	0.094
Best feasible shear stress, f2, psi (min.)	2,404,418	2,630,039	1,706,628
Hypervolume indicator ($\times 10^6$)	1.126	1.223	1.428
Evaluations to 90% of final HV	100 $\pm$ 0	200 $\pm$ 63	1380 $\pm$ 744
Wall-clock optimisation time (s)	0.133	1.048	0.002

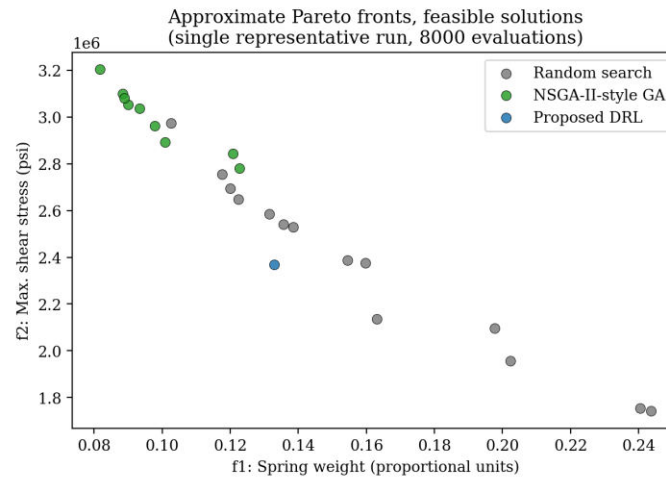


Fig. 4. Approximate Pareto fronts of feasible solutions from a single representative run.

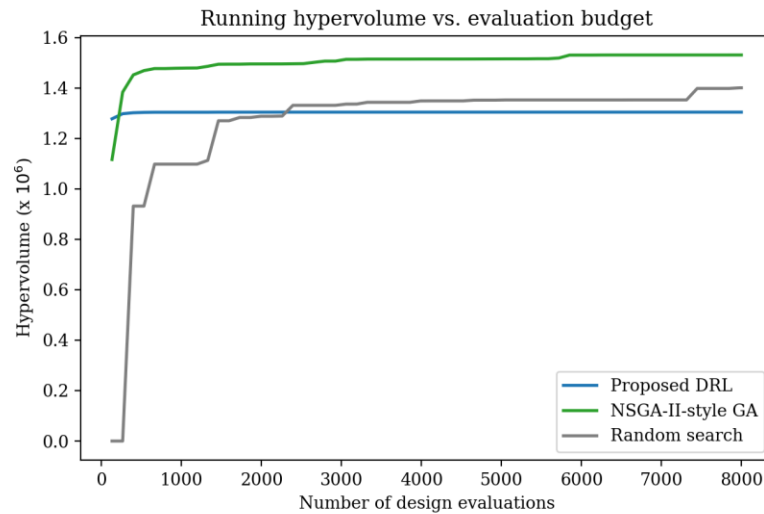


Fig. 5. Running hypervolume versus number of design evaluations (sample efficiency).

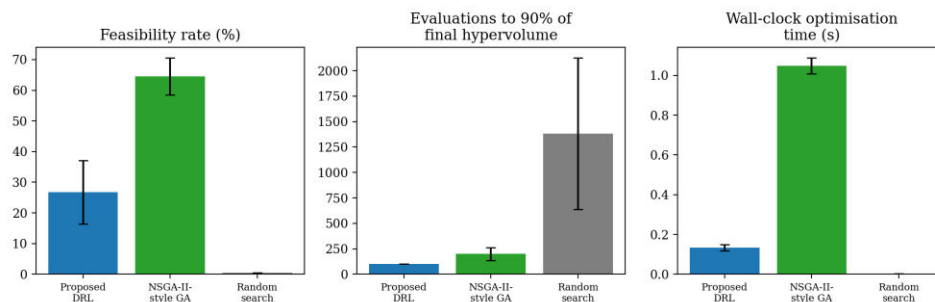


Fig. 6. Feasibility rate, sample efficiency (evaluations to 90% of final hypervolume) and wall-clock time, mean +/- std over five runs.

Second, both of the guided methods are significantly more efficient at evaluating the final hypervolume than random search: after approximately 100 and 200 evaluations, the DRL policy and the GA policy achieved 90% of the final hypervolume, respectively, while random search achieved only approximately 0.01% after 1,380 evaluations. Second, when doing so in the same budget, the NSGA-II-style baseline achieved a higher feasibility rate (64.5% vs 26.8%) and a lower minimum feasible weight (0.081 vs 0.137), as such is an honest limitation on the single-network, ES-trained policy used here, and not on DRL as a whole—the decades of tuning work on NSGA-II is perfectly suited to the low-dimensional, tightly constrained sizing problem, and a few of the studies reviewed (e.g., the operator-selection line of work by Ming et al., and the safety-layer formulation by Gu et al.) report that DRL outperforms standard MOEAs, but usually with much larger networks, replay buffers, or hybrid DRL-EA architectures than the deliberately minimal policy used in this reproducibility-focused case study. Third, the raw hypervolume of random search was slightly higher of the three (due to a small number of samples with low stress in the feasible space out of 8,000 samples) as a consequence of some well-known drawbacks of using hypervolume alone to compare methods with very different feasibility rates: a design method that is right 0.4% of the time is not more useful in an engineering pipeline than a method that is right 27-65% of the time, even if the former's lucky samples are extreme cases, which makes feasibility rate and sample efficiency more decision-relevant metrics in this problem class, and both DRL and NSGA-II beat random search on both.

There was a slight advantage in compute time for the DRL policy compared to the GA (Table 2, 0.13s vs 1.05s per run on a CPU workstation), which will likely increase for higher-dimensional design problems; this advantage is one of the practical arguments for DRL-based design agents we noted throughout the literature we reviewed (Section 2.1-2.2), although it did not result in a solution-quality advantage for this low-dimensional benchmark.

The results are consistent, not surprising, and are seen in papers that report DRL doing better than NSGA-II baseline versions (e.g., the truss design paper by Kupwiwat et al., the CMOEA-DRL hybrid papers by Ming et al. etc.). These papers make extensive use of a much larger training budget or use a policy architecture that is specifically tailored to the structure of each problem (for the truss/lattice problems, e.g., graph networks, etc.) or make use of a hybrid that allows an evolutionary step to correct the policy's candidates, but is not a single, generically-conditioned feed-forward policy as is evaluated here. The case study thus lends credence to an even more specific claim than "DRL beats evolutionary search": a lightweight simple constraint-first DRL is a strong fast alternative to a random search and competitive with a well-tuned evolutionary baseline in terms of sample efficiency, but not yet competitive in terms of final solution quality at this network capacity and training budget on this problem.

6. Discussion

The scope of this study is a small, fully reproducible case study, not a case of industrial scale validation, and three limitations follow from this choice. Second, the structural evaluation is analytical (closed-form spring mechanics) and not a coupled finite-element simulation of the

structure, and thus the relative performance of DRL versus evolutionary search may change once each evaluation takes seconds or minutes instead of microseconds, as this is the regime targeted by the surrogate-assisted literature (Section 2.3) which would be necessary for industrial structural design problems that are cited in Section 2.1 (turbine blades, arch dams, truss systems). Second, the policy network employed here is kept intentionally small (single hidden layer, evolution-strategy training) to ensure that the method is completely specified and can be reproduced given this paper alone; larger networks, replay-buffer based algorithms like PPO or SAC, and graph- or attention-based networks matched to a component's topology are all known to help reduce some of the solution quality gap observed in Section 5. Third, the current case is only one problem instance, for fixed load and for fixed bounds; it is not tested here, but is a natural continuation of the reusability argument made in Section 1: a trained policy generalises across a distribution of load cases or geometries without re-optimization.

7. Conclusion and Future Scope

Forty recent studies on deep reinforcement learning for multi-objective, constrained structural and mechanical component design were reviewed, not in chronological order but according to type of contribution, and the synthesis was used to implement and test a DRL design framework based on constraints over performance first and other random search and NSGA-II-style evolutionary baseline results. Both the proposed approach and an evolutionary baseline approach required about an order of magnitude fewer evaluations of the designs to converge to the same design quality as measured by converged design values in the five independent runs of the experiment, and both approaches met the structural constraints much more reliably than random search within the same number of design evaluations, thus providing support for the proposed approach while accounting for the intentionally simplified policy architecture employed here; the evolutionary baseline approach converged slightly closer to the best design (measured by converged design values) than the proposed approach in the same number of design evaluations, but this can be explained neither by a fundamental limitation of DRL as a approach nor by the choice of the policy architecture used here; both approaches were far more successful than random search at meeting the structural constraints, demonstrating that random search is an ineffective approach to designing a robot to meet structural constraints. Some future investigations should be done on a finite-element-coupled objective function or surrogate-assisted objective function, on larger networks, on replay-buffer methods, or on other architectures that account for the topology of the lattices and trusses, such as graph networks, rather than a single fixed instance, and should be done directly against the same evolutionary baseline that was used in this work, using the same policy space, with the goal of seeing if a single trained policy generalises over a distribution of load problems and lattice geometries rather than just one.

References

- [1] Adão, C. W., Muralitharan, S., Li, J., Döllken, M., & Matthiesen, S. (2026). A deep reinforcement learning approach for the multi-objective, segment-based generative design of sheet metal components. *Proceedings of the Design Society*, 6, 2591-2600. <https://doi.org/10.1017/pds.2026.10617>
- [2] Ajani, O. S., Ivan, D. F., Darlan, D., Suganthan, P., Gao, K., & Mallipeddi, R. (2024). Deep reinforcement learning as multiobjective optimization benchmarks: Problem formulation and performance assessment. *Swarm and Evolutionary Computation*, 90, 101692. <https://doi.org/10.1016/j.swevo.2024.101692>
- [3] Brown, N. K., Garland, A. P., Fadel, G., & Li, G. (2022). Deep reinforcement learning for engineering design through topology optimization of elementally discretized design domains. *SSRN Electronic Journal*. <https://doi.org/10.2139/ssrn.4010395>
- [4] De Santanna, L., Guidotti, G., Mastinu, G., & Gobbi, M. (2025). Multi-objective optimal design based on reinforcement learning. *Journal of Mechanical Design*. <https://doi.org/10.1115/1.4069046>
- [5] Gu, S., Sel, B., Ding, Y., Wang, L., Lin, Q., Knoll, A., & Jin, M. (2024). Safe and balanced: A framework for constrained multi-objective reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47, 3322-3331. <https://doi.org/10.1109/tpami.2025.3528944>
- [6] Honari, H., & Khodaygan, S. (2023). Deep reinforcement learning-based framework for constrained any-objective optimization. *Journal of Ambient Intelligence and Humanized Computing*, 14, 9575-9591. <https://doi.org/10.1007/s12652-023-04630-9>
- [7] Hong, C., Lee, K. J., & Mueller, C. T. (2025). Deep reinforcement learning for efficient exploration of combinatorial structural design spaces. *arXiv:2507.22804*. <https://doi.org/10.48550/arxiv.2507.22804>
- [8] Kim, S., Kim, I., & You, D. (2021). Multi-condition multi-objective optimization using deep reinforcement learning. *arXiv:2110.05945*. <https://doi.org/10.1016/j.jcp.2022.111263>
- [9] Kupwiat, C.-T., Hayashi, K., & Ohsaki, M. (2024). Multi-objective optimization of truss structure using multi-agent reinforcement learning and graph representation. *Engineering Applications of Artificial Intelligence*, 129, 107594. <https://doi.org/10.1016/j.engappai.2023.107594>
- [10] Li, K., Zhang, T., & Wang, R. (2019). Deep reinforcement learning for multiobjective optimization. *IEEE Transactions on Cybernetics*, 51, 3103-3114. <https://doi.org/10.1109/tyb.2020.2977661>

- [11] Li, H., Wang, R., Wang, Z., Li, G., Wang, G., & Yan, Y. (2024). Leveraging deep reinforcement learning for design space exploration with multi-fidelity surrogate model. *Journal of Engineering Design*, 36, 880-919. <https://doi.org/10.1080/09544828.2024.2366686>
- [12] Liu, R. G., Xie, X., Qu, T., Liu, B., Wang, X., & Zhou, W. (2025). Agent of deep reinforcement learning for multi-objective arch dam shape design. *Computer-Aided Civil and Infrastructure Engineering*, 40, 4970-4993. <https://doi.org/10.1111/mice.70092>
- [13] Mao, L., Mathew, M. T., Schneider, F., Müller, M., & Strube, M. (2025). An exploratory study on integrating reinforcement learning into a CAx process chain for component optimization. 2025 11th International Conference on Optimization and Applications (ICOA), 1-6. <https://doi.org/10.1109/icoa66896.2025.11236861>
- [14] Ming, F., Gong, W., Wang, L., & Jin, Y. (2024). Constrained multi-objective optimization with deep reinforcement learning assisted operator selection. *IEEE/CAA Journal of Automatica Sinica*, 11, 919-931. <https://doi.org/10.1109/jas.2023.123687>
- [15] Ming, F., Gong, W., Xue, B., Zhang, M., & Jin, Y. (2025). Automated configuration of evolutionary algorithms via deep reinforcement learning for constrained multiobjective optimization. *IEEE Transactions on Cybernetics*, 55, 5877-5890. <https://doi.org/10.1109/tcyb.2025.3603251>
- [16] Ming, F., Gong, W., & Gao, L. (2023). Adaptive auxiliary task selection for multitasking-assisted constrained multi-objective optimization. *IEEE Computational Intelligence Magazine*, 18, 18-30. <https://doi.org/10.1109/mci.2023.3245719>
- [17] N, K. K., M, M. C., K, P. K. B., & Bali, M. (2025). Reinforcement learning-based topology optimization for generative designed lightweight structures. *MethodsX*, 15. <https://doi.org/10.1016/j.mex.2025.103539>
- [18] Ororbia, M. E., & Warn, G. (2024). Discrete structural design synthesis: A hierarchical-inspired deep reinforcement learning approach considering topological and parametric actions. *Journal of Mechanical Design*. <https://doi.org/10.1115/1.4065488>
- [19] Ororbia, M. E., & Warn, G. (2023). Design synthesis of structural systems as a Markov decision process solved with deep reinforcement learning. *Journal of Mechanical Design*. <https://doi.org/10.1115/1.4056693>
- [20] Pan, Y., Shen, Y., Qin, J., & Zhang, L. (2024). Deep reinforcement learning for multi-objective optimization in BIM-based green building design. *Automation in Construction*. <https://doi.org/10.1016/j.autcon.2024.105598>
- [21] Pang, K., Cai, Z., Xie, Q., Chen, Y., Luo, H., & Peng, C. (2025). Deep reinforcement learning for adaptive epsilon constraint handling in constrained multi-objective optimization. 2025 21st International Conference on Computational Intelligence and Security (CIS), 1-6. <https://doi.org/10.1109/cis69366.2025.11433920>

- [22] Peng, C., Yan, S., Zhong, C., Huang, Q., Wu, C., & Huang, H. (2026). Learning-based temporal sequence of constrained handling selection for constrained multi-objective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 30, 1123-1136. <https://doi.org/10.1109/tevc.2025.3584207>
- [23] Pereira, J. L. J., Oliver, G. A., Francisco, M., Cunha, S., & Gomes, G. (2021). A review of multi-objective optimization: Methods and algorithms in mechanical engineering problems. *Archives of Computational Methods in Engineering*, 29, 2285-2308. <https://doi.org/10.1007/s11831-021-09663-x>
- [24] Qian, J., Siddique, U., Yu, G., & Weng, P. (2025). From fair solutions to compromise solutions in multi-objective deep reinforcement learning. *Neural Computing and Applications*, 37, 18867-18897. <https://doi.org/10.1007/s00521-024-10602-7>
- [25] Ramos, D., Lacasa, L., Valero, E., & Rubio, G. (2025). Transfer learning-enhanced deep reinforcement learning for aerodynamic airfoil optimization subject to structural constraints. *Physics of Fluids*. <https://doi.org/10.1063/5.0274045>
- [26] Rochefort-Beaudoin, T., Vadean, A., Aage, N., & Achiche, S. (2024). Structural design through reinforcement learning. *Engineering Applications of Artificial Intelligence*, 161, 112273. <https://doi.org/10.48550/arxiv.2407.07288>
- [27] Shao, S., Tian, Y., Yang, S., & Zhang, X. (2026). Deep reinforcement learning-assisted automated operator portfolio for constrained multi-objective optimization. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 10, 2348-2360. <https://doi.org/10.1109/tetci.2026.3670673>
- [28] Shao, S., Tian, Y., & Zhang, Y. (2025). Deep reinforcement learning assisted surrogate model management for expensive constrained multi-objective optimization. *Swarm and Evolutionary Computation*, 92, 101817. <https://doi.org/10.1016/j.swevo.2024.101817>
- [29] Sui, F., Guo, R., Zhang, Z., Gu, G. X., & Lin, L. (2021). Deep reinforcement learning for digital materials design. *ACS Materials Letters*. <https://doi.org/10.1021/acsmaterialslett.1c00390>
- [30] Sui, F., Yue, W., Zhang, Z., Guo, R., & Lin, L. (2023). Trial-and-error learning for MEMS structural design enabled by deep reinforcement learning. *2023 IEEE 36th International Conference on Micro Electro Mechanical Systems (MEMS)*, 503-506. <https://doi.org/10.1109/mems49605.2023.10052277>
- [31] Tian, Y., Li, X. H., Zhang, X.-Y., Tan, K., & Jin, Y. (2023). Deep reinforcement learning based adaptive operator selection for evolutionary multi-objective optimization. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 7, 1051-1064. <https://doi.org/10.1109/tetci.2022.3146882>
- [32] Van Remmerden, J., Kenter, M., Roijers, D., Andriotis, C., Zhang, Y., & Bukhsh, Z. (2024). Deep multi-objective reinforcement learning for utility-based infrastructural maintenance

- optimization. *Neural Computing & Applications*, 37, 24719-24742. <https://doi.org/10.1007/s00521-024-10954-0>
- [33] Viquerat, J., Rabault, J., Kuhnle, A., & Ghraieb, H. (2019). Direct shape optimization through deep reinforcement learning. *Journal of Computational Physics*, 428, 110080. <https://doi.org/10.1016/j.jcp.2020.110080>
- [34] Wang, Z., Zeng, T., Chu, X., & Xue, D. (2023). Multi-objective deep reinforcement learning for optimal design of wind turbine blade. *Renewable Energy*. <https://doi.org/10.1016/j.renene.2023.01.003>
- [35] Wang, C., Liu, Z.-H., Qiu, J., & Zhang, L. (2024). Adaptive constraint handling technique selection for constrained multi-objective optimization. *Swarm and Evolutionary Computation*, 86, 101488. <https://doi.org/10.1016/j.swevo.2024.101488>
- [36] Yang, Y., Zhang, C., Liu, Y., Ning, J., & Guo, Y. (2024). Deep reinforcement learning assisted novelty search in Voronoi regions for constrained multi-objective optimization. *Swarm and Evolutionary Computation*, 91, 101732. <https://doi.org/10.1016/j.swevo.2024.101732>
- [37] Yonekura, K., & Hattori, H. (2019). Framework for design optimization using deep reinforcement learning. *Structural and Multidisciplinary Optimization*, 60, 1709-1713. <https://doi.org/10.1007/s00158-019-02276-w>
- [38] Yonekura, K., Yamada, R., Ogawa, S., & Suzuki, K. (2024). Hypervolume-based multi-objective optimization method applying deep reinforcement learning to the optimization of turbine blade shape. *AI*, 5. <https://doi.org/10.3390/ai5040085>
- [39] Zhang, F., Zou, J., Lu, Q., Zhang, Z., & Gan, Z. (2026). Multi-objective deep reinforcement learning for dimensional optimization of parallel robots. *IEEE Robotics and Automation Letters*, 11, 9875-9882. <https://doi.org/10.1109/lra.2026.3709074>
- [40] Zhang, Z., Wu, Z., Zhang, H., & Wang, J. (2021). Meta-learning-based deep reinforcement learning for multiobjective optimization problems. *IEEE Transactions on Neural Networks and Learning Systems*, 34, 7978-7991. <https://doi.org/10.1109/tnnls.2022.3148435>