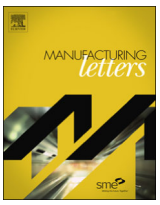
Available online at www.sciencedirect.com

Manufacturing Letters

Manufacturing Letters 35 (2023) 1203–1213



51st SME North American Manufacturing Research Conference (NAMRC 51, 2023)

Deep reinforcement learning for stacking sequence optimization of composite laminates

Sara Shonkwiler[a], Xiang Li[a], Richard Fenrich[b], Sara McMains[a,*]

[a] *University of California, Berkeley, United States*
[b] *Arevo Inc., Milpitas, United States*

Abstract

Fiber reinforced polymer (FRP) composite laminates are increasingly used in a wide range of safety-critical products due to their excellent material properties. The stacking sequence of FRP composite laminates plays a critical role in the resulting part's mechanical properties. Despite this, composite engineers still commonly fabricate parts with "classic" ply layups (e.g. four ply laminate with fiber orientation angles: [-45/0/45/90°]), which may have sub-optimal mechanical performance in the expected loading/use cases. Finding the composite stacking sequence that achieves the best material and mechanical properties possible is a challenging optimization problem characterized by the large domain space involved in solving an inverse design problem. This paper introduces a novel approach to optimizing stacking sequence for composite plate stiffness by applying off-policy deep reinforcement learning (DRL). We formulate the problem as a sequential decision making process. The state of the system is based on the stiffness of the composite for the currently selected stacking sequence and the action is to select a new stacking sequence using our reward function, formulated as the offset, normalized stiffness modulus. We compare our DRL model to two classical stacking sequences, a randomized baseline model, and a competitive genetic algorithm (NSGA-II). For maximizing longitudinal composite plate stiffness, the DRL model finds the optimum solution for all ply thicknesses and the genetic algorithm comes within 0.5% of the optimum. The DRL model determines the optimum stacking sequence significantly faster and is less sensitive to parameter tuning. The DRL model and the genetic algorithm both outperform the random baseline algorithm by over 5.7 standard deviations in the most conservative case. This research demonstrates the ability of DRL to effectively and efficiently optimize composite laminate stacking sequence.

Keywords: Fiber-reinforced polymer composite; Composite stacking sequence; Deep reinforcement learning

1. Introduction

Fiber reinforced polymer (FRP) composites are increasingly used, especially in high performance manufacturing applications (e.g. aerospace, automotive, sports equipment), due to their high strength-to-weight ratio, corrosion resistance, and other favorable material properties [45, 36].

Fiber Reinforced Polymer (FRP) composites consist of (1) fiber reinforcements and (2) polymer resin matrix. The fibers, commonly made of carbon or glass, support the majority of the loads applied to FRP parts. The resin matrix connects the fibers and transfers the stress among the fibers. Composite laminates are made by stacking a series of single orientation fiber layers or lamina. Composite laminates generally have orthotropic mechanical properties that can be altered or tuned. Lamina stacking sequence is vital to determining FRP composite mechanical properties [22].

The choice of lamina fiber orientations, called the composite stacking sequence, impacts the mechanical properties of the resulting composite part [31, 11, 34]. As manufacturing processes for FRP composites improve, there is increasing control over stacking sequence. Increased control opens an opportunity to efficiently optimize stacking sequence to optimize mechanical properties for anticipated loading conditions. Currently, engineers in industry often treat FRP composites as "black alu-

* Corresponding author. Tel.: +0-000-000-0000 ; fax: +0-000-000-0000.
E-mail address: mcmains@berkeley.edu (Sara McMains).

minum," using default stacking sequences (e.g. [0,45,-45,90°]) with "fixed" mechanical properties. Not utilizing the ability to fully tune composite properties is a loss of potential weight and cost savings. FRP composites' widespread use, especially in safety and weight critical applications, coupled with few industry-ready tools to optimize FRP composite stacking sequence, makes it an important material to optimize or tune individual mechanical properties for specific use case needs. For the purpose of this paper, we will focus on maximizing composite plate stiffness (E_x, E_y) as two examples of mechanical properties that can be optimized.

A plethora of optimization algorithms have been implemented to try to optimize composite stacking sequence for given use cases and desired properties; however, according to recent reviews and analysis [44, 41, 4], these algorithms have significant limitations, such as only guaranteeing local optimal results or using outdated evolutionary optimization algorithms.

In this research, a Deep Reinforcement Learning (DRL) framework is developed to design mechanically stiff FRP composites by optimizing over the stacking sequence. Our main contributions include:

- A novel approach to composite stacking sequence optimization using deep reinforcement learning.
  - Performs well: Optimizes E_x with results 5.7, 6.5, 10.7, and 15.5 standard deviations above the mean results for random stacking sequences for 3, 9, 32, and 70 ply respectively.
  - Fast: For continuous stacking sequence [0,180°) 3, 9, 32, and 70 ply laminates, calculates E_x in ∼1, 3, 6, 11 seconds respectively.
- A comparative analysis of deep reinforcement learning to a genetic algorithm and two baseline algorithms.
- Discussion of design choices, parameter tuning details, and sensitivity analysis that other researchers can build upon.

2. Background

FRP composite mechanical properties depend on their microstructure, such as the spatial arrangement of the fibers. For composite laminates the most easily changed microstructure property is the orientation or stacking sequence of fibers in the matrix [5]. As early as 2010, Bloomfield [4] pointed out some limitations of existing composite stacking sequence optimization methods (primarily lamination parameters and genetic algorithms), yet as of 2020, Wang and Sobey [44] state that published work still uses older evolutionary optimization algorithms that only guarantee locally optimal results. Additionally, in industry, engineers rarely use existing optimization techniques, often due to time constraints.

One method to optimize composite laminate stacking sequence is using gradient based approaches [38, 24, 30, 12]. These methods require being able to calculate the gradient, can only guarantee that solutions are a local optima for the stacking sequence, and are unable to handle large numbers of design variables (which many modern problems have) [4].

The use of lamination parameters as an intermediate design variable is another approach to composite stacking sequence optimization [43, 29, 10, 14, 15, 32, 8, 18, 23, 40]. As stated in Tsai & Hahn [43], researchers use material invariants and twelve lamination parameters to define stiffness properties of composite laminates. Miki [29] uses lamination parameters to maximize buckling load by selecting the optimal stacking sequence. Fukunaga & Vanderplaats [10] use lamination parameters and mathematical programming techniques to optimize the stiffness of orthotropic laminates. Haftka & Walsh [15] and Nagendra et al. [32] use lamination parameters and integer programming techniques to optimize stacking sequence subject to buckling constraints (and strength constraints in the case of Nagendra et al.). Unfortunately, integer programming techniques are computationally expensive. Diaconu & Sekine use lamination parameters to maximize the buckling load of laminated composite shells [8]. Grenestedt & Gudmundson [14] derives the possible region of lamination parameters for symmetric orthotropic laminates. One of the main limitations of lamination parameters is the full feasible region of lamination parameters is unknown [4].

Another stacking sequence optimization approach is using simulated annealing as in Erdal & Sonmez [9]. These approaches have several downsides such as points may converge to a non-optimal solution and they are not good at optimizing for specific use cases (they are better for general stacking sequence optimization) [13].

Genetic algorithms are the most popular type of algorithm used to optimize composite stacking sequence [28, 16, 39, 6, 1, 27, 35, 42]; however, genetic algorithms may only yield local optimums [41] and as described in Wang & Sobey [44], many of the genetic algorithms used in composite stacking sequence optimization research are lacking in documentation of parameter choices. The genetic algorithm used as a comparison in this research is described in detail in subsection 3.4. Researchers have also investigated using other optimization algorithms, such as multi-objective evolutionary algorithms [4, 2, 21, 20, 19] for composite stacking sequence optimization.

Reinforcement learning (RL) is the branch of machine learning (ML) where an agent takes a series of actions based on interactions with an environment attempting to maximize expected rewards, $r(s, a)$. The strategy the algorithm uses to pick successive actions is called the policy. Unlike traditional ML or deep learning (DL) models, RL models do not generally use pre-existing data to learn a model, but rather RL models generate data as they learn which is a key advantage to RL. Further, RL does not require "successful" data in order to learn. Reinforcement learning occurs in a simulated world with policies, states and/or observations, actions, buffers, and rewards. The goal of a RL algorithm is to learn a policy, π_θ, that maps the current state, s_t, to an action, a_t. In the case of Deep Reinforcement Learning (DRL), the policy is generally a neural network and θ represents the parameters (weights, biases) of that deep neural network. Thus for DRL, the goal is to find parameters, θ, that define the policy so as to maximize the expected value of the sum of the rewards over the trajectory (τ):

$$\theta^* = \arg\max_{\theta} E_{\tau\sim\pi_\theta(\tau)}[\sum_t r(s_t, a_t)] \quad (1)$$

Deep reinforcement learning has been used in other material design and manufacturing applications [25]. For example, RL is used in Zheng et al. [46] to optimize the distribution of functional groups in graphene oxide. Sui et al. designed a DRL framework and used a collaborative deep Q network (DQN) architecture to optimize average tensile strength along the primary axes for small 2D grids representing composite parts. This research toggles which areas of a 2D composite part are the soft versus stiff material to find the optimal location of soft and stiff materials [41]. This research is limited to small, not necessarily continuous fiber, 2D square cross sections. To the best of our knowledge, reinforcement learning has never been used to optimize composite stacking sequence. In the following section, we detail our approach to creating a DRL model targeted at optimizing composite stacking sequence, as well as the underlying composite plate model, and two baseline models, and a genetic algorithm for comparison.

3. Methods

3.1. Overview of optimization models

This paper compares the performance of four models — our DRL model, an NSGA-II genetic algorithm, and two baseline models — at optimizing composite stacking sequence for stiffness. The models are summarized in Table 1 below. Detailed descriptions of each algorithm and the motivation behind using them can be found in the following subsections.

Table 1: Models

Model Type	Model Description
Baseline Model 1	Randomized model (1000 runs)
Baseline Model 2	Classic stacking sequences (e.g. $[0/45/90°]_3$)
Genetic Algorithm	NSGA-II
DRL Algorithm	Deep deterministic policy gradient model

3.2. Composite plate stiffness model

We calculate composite plate stiffness using standard plate homogenization from basic composite laminate theory based on [33]. The material used throughout this paper is based on a composite laminate made by the company Arevo Inc. with the following properties:

- $Volume fraction, v_f = 0.50$
- $E_1 = 115GPa$
- $E_2 = 7.1GPa$

based on the material used for testing in [37]. The model input is the stacking sequence, expressed as a 1D array, $nx1$ (where n is number of plies), to represent a single laminate plate. The simulation calculates resulting composite mechanical properties including longitudinal and transverse Young's Modulus (E_x, E_y). For this research, four different model/example composites were optimized: 3, 9, 32, and 70 ply composite plates. Realistic use cases for each composite model level are described in Table 2.

Table 2: Composite Laminate Model Parts

Number of plies	Use Case
3	Simple test case of laminate over core material, allows 3D design space display
9	Thin solid laminate
32	Medium solid laminate, similar to [19]
70	Thick solid laminate, reasonable upper bound due to manufacturing process and time constraints

3.3. Deep reinforcement learning model

In DRL models, the state s_t of the system is an input to the policy and represents where the model is at a given point in time, t. The action, a_t, represents the output of the policy or the way the model changes states. The reward function, $r(s, a)$, encourages actions that lead to desirable states and discourages actions that lead to undesirable states.

In our DRL model, each state is a representation of the stiffness of the composite for a given stacking sequence ($s_t = normalize\_factor * E_{x/y}$). Each action is a one dimensional array of length n (the number of plies) of numbers between $[0, 180)$, where we define zero degrees to be the direction of longitudinal pull. The n-dimensional action array represents the angles, in degrees, of the fibers in each layer (i.e. the stacking sequence). Since our model picks as each action an entirely new stacking sequence, it is able to update/change quickly and potentially find the optimum in a short amount of time. The reward function is determined from the longitudinal or transverse Young's Modulus (E_x or E_y), the stiffness of the laminate in the longitudinal or transverse direction. Normalizing rewards to $[-1, 1]$ roughly balances half the rewards as "good" (positive rewards) and half the rewards as "bad" (negative rewards). Normalizing rewards can also be thought of as controlling the variance of the policy gradient estimator (i.e. avoiding extreme values in the neural network weights). Practically this often results in faster learning than the same model and parameters without normalization. Thus, to achieve normalized rewards, we offset the current stiffness ($E_x(s_t)$ or $E_y(s_t)$) by the mean stiffness for that direction (μ_{E_x} or μ_{E_y}) so that the reward has zero mean. Then we divide by six times the standard deviation (σ_{E_x} or σ_{E_y}) to cap the bounds to approximately $[-1, 1]$. Our reward function is defined as follows:

$$r(s_t) = \frac{E_{x/y}(s_t) - \mu_{E_{x/y}}}{6\sigma_{E_{x/y}}} \quad (2)$$

where $E_{x/y}(s_t)$ is the longitudinal/transverse composite plate stiffness at the current state, $\mu_{E_x/y}$ is the mean composite plate stiffness in that direction, and $\sigma_{E_x/y}$ is the standard deviation of composite plate stiffness in that direction. Our DRL model variables are summarized in Table 3, along with an example. An overview of our DRL model is shown in Figure 1.

Table 3: DRL Model Variables

Variable	Description	Example
State (s_t)	Normalized stiffness modulus	$s_t = 2.23$
Action (a_t)	Selecting a new stacking sequence	$a_t = [23.34, 4.53, ..., 130.49]$
Reward (r_t)	Normalized, offset stiffness modulus	$r_t = -0.25$

The goal of the reinforcement learning algorithm is to learn the policy, π_θ, that maps the current state to an action. Similar to Sui et al. [41], we use a fully connected neural network to approximate the policy. After conducting experiments to test different numbers of neural network layers and nodes, we determined that a two layer, 64 node network is sufficient for this optimization problem (more layers or nodes only increased the run times). Thus we use a two layer, 64 node multi-layer perceptron (MLP) neural network to model the Q-function and policy at each time step. (Note that a deeper neural network might be called for to make more complicated optimization problems feasible.)

There are numerous RL algorithms that generally fall into three categories: on-policy RL, off-policy RL, and offline RL. On-policy RL updates the policy with data collected by the current policy. Off-policy RL updates the policy with data from all policies the RL algorithm has generated, sampled from a buffer. Offline RL uses data collected prior to running the algorithm with no online data collection. We chose an off-policy approach to take advantage of reusing already collected data from previous policies. This saves time because accurately modeling many composite stacking sequences takes a non-trivial amount of time. Off-policy RL stores data from previous policies in a buffer and samples from this buffer to update the policy. This is advantageous because it means the model learns from a wide range of previous experiences. It is also advantageous to be able to reuse previously collected data instead of having to collect new data at every time step. The advantage of off-policy would be even higher for more computationally expensive modeling, such as FEM.

We implemented the deep deterministic policy gradient (DDPG) algorithm. DDPG is an actor-critic, off-policy, model-free algorithm that extends deep Q-learning to the continuous action space. Q-learning uses a Q-table to map state, action pairs to expected future rewards. The Q-value is the maximum expected reward an agent can get if it takes a particular action from state s_t. Deep Q-learning uses a neural network to approximate the relationship between states, actions, and expected future rewards.

Using DDPG, the model can predict any ply orientation instead of only fixed orientations. This makes DDPG ideal for high precision stacking sequence tuning. Additionally, it avoids unnecessary action space discretization that reduces the algorithm's ability to learn relationships between nearby actions.

DDPG uses off-policy data and the mean squared Bellman error to learn the Q-function. In turn, the Q-function learns the updated policy, π_θ.

The Bellman equation is the optimal action-value function:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}\Big[r(s_t, a_t) + \gamma\, \mathbb{E}_{a_{t+1} \sim \pi}[Q^\pi(s_{t+1}, a_{t+1})]\Big] \quad (3)$$

where $r(s_t, a_t)$ is the reward, γ is the discount factor (how much to prioritize near term versus long term rewards), and $Q^\pi(s_{t+1}, a_{t+1})$ is the ideal Q-value for the following state, action pair.

To learn the Q-function, DDPG minimizes the mean squared Bellman error:

$$L(\phi, D) = \mathbb{E}_{s_t, a_t, r_t, s_{t+1} \sim D}\Big[\Big(Q_\phi(s_t, a_t) - \big(r(s_t, a_t) + \gamma Q_{\phi_{targ}}(s_{t+1}, \pi_{\theta_{targ}}(s_{t+1}))\big)\Big)^2\Big] \quad (4)$$

which represents how close our approximator, $Q_\phi(s, a)$, is to $Q^\pi(s, a)$. The DDPG algorithm approximates $Q^\pi(s, a)$ using a neural network, $Q_\phi(s, a)$, in our case the two layer MLP neural network.

Choosing an appropriate ML architecture based on process-related knowledge is an important factor for RL optimization. In the case of this optimization, our model is the first RL model to optimize composite stacking sequence; accordingly, we selected a relatively straightforward optimization task (i.e. stiffness). Since the optimization task is straightforward, a "traditional" two-layer MLP architecture is sufficient. We anticipate that the ML model for more complicated optimization cases will need to be carefully fine-tuned based on process related knowledge.

To learn the actor policy, π_θ, from the Q-function, we take the gradient ascent of the Q-function with respect to the policy parameters.

DDPG creates target networks, θ_{targ} and ϕ_{targ}, for the policy network and Q-function network respectively, which are the same neural networks, but the weights and biases lag the original neural networks because the target networks depends on the same parameters we are trying to learn; without this difference learning would be unstable. The target networks are periodically updated.

The DDPG algorithm performs the following operations for each step while learning: (1) select an action using the current policy and state s_t, (2) use that action to calculate reward r_t

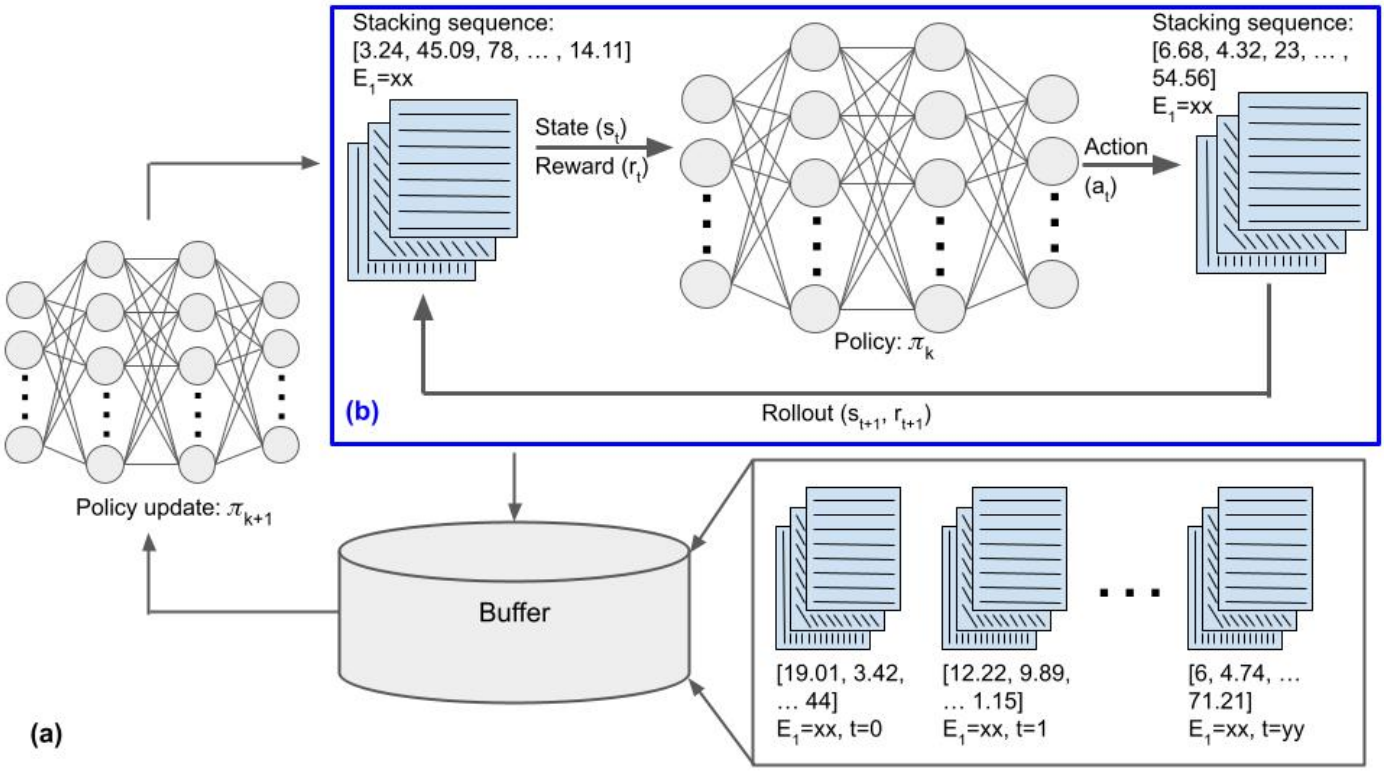


Fig. 1: DRL state transition and model for composite plate stiffness maximization. (a) Our off-policy DRL model. The buffer stores data (transitions) from previous policies. The neural network policy is updated periodically. (b) (upper right) The portion of our off-policy DRL model that shows one state transition. Our neural network models the DRL policy, π_k, and maps states and rewards to actions. After an action is taken, the state and the reward are updated.

and update state s_{t+1}, (3) store transition s_t, a_t, r_t, s_{t+1} in buffer, (4) randomly sample transitions from buffer (mini-batch), (5) update critic network/Q-function, $Q_\phi(s, a)$, to minimize mean squared Bellman error using stochastic gradient descent, using target networks (6) update actor policy network, π_θ, using the sampled policy gradient ascent, and (7) update target networks, θ_{targ} and ϕ_{targ}. At the end of each step, the algorithm returns to operation (1) and repeats. At the end of each episode, the state is reset before returning to operation (1) [26].

Deep reinforcement learning models, and ML models in general, involve multiple user-defined parameters that generally need to be tuned or varied for optimal results. We tune the following DRL parameters:

- Learning rate (α): value between 0 and 1 that determines how quickly the parameters, θ, in the policy are updated (tuned over $[0.0005, 0.001, 0.003, 0.005, 0.01, 0.05, 0.1]$)
- Learning starts: time step at which learning process begins (tuned over 0–400)
- Time steps: number of discrete time steps the DRL algorithm executes. An example time step can be seen in Figure 1b (tuned over 20–1000)
- Episode length: number of time steps before the DRL environment resets to the initial state and reward is reset to zero (tuned over 1–10)

3.4. Genetic algorithm (NSGA-II) model

The genetic algorithm implemented in this paper is the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [7], which is the most frequently used in composite stacking sequence optimization research [44]. The algorithm is implemented with an open-source optimization framework, Pymoo [3].

Genetic algorithms, a subclass of evolutionary algorithms, are a type of search heuristic based loosely on the process of natural selection. In general, genetic algorithms consist of the following phases: (1) initial population generation, (2) fitness function calculation, (3) selection of fittest individuals (parents), and (4) crossover and mutation operations. The crossover operation is the exchanging of parent genes to create entirely new individuals (offspring). The mutation operation is a type of noise created by inserting of random genes in the offspring to promote diversity and reduce the odds of premature convergence.

NSGA-II is a non-dominated sorting genetic algorithm, which means each solution in the solution set is only better than every other solution in the solution set in one or more objectives if it is worse at other objectives. The non-dominated set of solutions represents the Pareto optimal set. The NSGA-II algorithm (1) creates an initial population; (2) executes non-dominated sorting and classifies non-dominating fronts; (3) uses crowding distance to further sort the population, if necessary; (4) selects fittest individuals using tournament selection; (5) performs

crossover and mutation operations; and (6) combines offspring with previous generation; and (7) repeats (2)-(6) until termination.

In this paper, the initial population is a random set of ply orientations (a random stacking sequence). Non-dominated sorting and front classification is determined by running the composite plate stiffness model to calculate E_x and/or E_y (E_x or E_y for single-objective optimization, E_x and E_y for multi-objective optimization).

The size of the population and termination condition used in genetic algorithms are user-determined parameters. We tune population size over $[10, 20, 50, 75, 100, 200]$. Three termination conditions were tested: (1) a tolerance of one degree change in orientation on average over the stacking sequence for a period of 20, (2) a tolerance of 100 Pa change in stiffness for a period of 20, and (3) a timed termination to match the DRL model for comparison. The period represents the number of generations over which the termination condition is checked.

3.5. Baseline models

The first baseline model for comparison calculates composite plate stiffness for several "classic" stacking sequences (e.g. [0/45/90°], [-45/0/45/90/-45/0/45/90/0°]). These are the types of stacking sequences commonly used by engineers in industry instead of tuning for the optimal stacking sequence [33, 17]. As such, comparing to these stacking sequences gives a reasonable comparison of how much better our algorithm is compared to what many engineers in industry are using. Including this comparison demonstrates the potential for sub-par mechanical properties when using "classic" stacking sequences.

The second baseline model is a randomized stacking sequence. We create 1,000 $nx1$ dimensional arrays for each ply thickness where n represents the number of plies in the laminate. Each array element is a random number $[0, 180)$. From this data set, mean, standard deviation, and maximum composite plate stiffness are calculated. These are the types of stacking sequences that a computer would generate without human knowledge of composite behavior. The best random stacking sequence outperforms the industry standard "classic" stacking sequences as can be seen in Figure 2. Including this second baseline emphasizes how much room there is for improvement over using "classic" stacking sequences — even our random baseline model will improve performance.

4. Results and Discussion

4.1. Longitudinal composite plate stiffness maximization

We use our DRL model to maximize longitudinal or tensile composite plate stiffness (which we define as E_x). We compare the DRL results to the results from the genetic algorithm (NSGA-II), as well as the two baseline algorithms ("classical" and 1,000 random orientation layups). Composite plate stiffness is maximized in this paper, but this is just one example of a mechanical property that can be optimized using DRL. Engineers, designers, and researchers can customize DRL to optimize for different mechanical properties.

Our DRL model successfully finds the maximum possible longitudinal stiffness , $115GPa$, for all composites tested (3, 9, 32, and 70 ply composites). The genetic algorithm (NSGA-II), using termination conditions (1) and (2), comes within ~0.5% of the maximum possible longitudinal stiffness.These results are displayed in Figure 2.

The DRL model executes significantly faster than the genetic algorithm with termination conditions (1) and (2). The time differential increases to the second power with number of plies, as shown in Figure 3.

NSGA-II termination condition (3) is set to take roughly the same amount of time as the DRL model for direct performance comparison. NSGA-II termination condition (3) and the 1,000 random runs maximum perform decently at optimizing stiffness for 3 plies, but their performance drops off significantly for parts with more plies. Additionally, 1,000 random runs takes longer than the DRL model, once again with the time differential increasing with number of plies (Figure 3).

The baseline models (with the exception of 1,000 random runs maximum) perform poorly at maximizing stiffness. The single run random baseline, which represents selecting a random stacking sequence, performs poorly and has considerable variation in performance. The gap in performance between the DRL model, genetic algorithm, and baseline models grows as the number of plies increases, which highlights the importance of using an appropriate model rather than simply using trial and error.

DRL algorithms learn by maximizing expected return or the expected cumulative reward. Plots of our DRL model rewards over the different ply experiments are shown below in Figure 4. Our DRL model hits the reward upper bound efficiently. The model quickly learns that taking actions moving towards the action space bounds (0 or 180) improves model performance, measured by the reward function. The model updates the entire stacking sequence at each time step. This means the model is nimble and can change very quickly — allowing it to optimize straightforward objectives, such as E_x, relatively quickly. Additionally, the model cannot overshoot the optimum stacking sequence, which simplifies the optimization problem (for both the genetic algorithm and our DRL model), which further improves the model efficiency.

4.2. Transverse composite plate stiffness maximization

The focus of this paper is on optimizing longitudinal or tensile composite plate stiffness, but to demonstrate that this model can be adapted for other mechanical properties we highlight optimizing transverse composite plate stiffness. Maximizing longitudinal composite plate stiffness is relatively straightforward (for both the genetic algorithm and DRL model) because the solution set ($[0, 0, ..., 0]$, $[180, 180, ...180]$ and combinations of angle 0 and 180) lie on the bounds of the action space. The models cannot overshoot the optimal action and can therefore converge much faster. We could easily optimize transverse or transverse tensile composite plate stiffness by making the ac-

tion space bounds [-90, 90) so that the "best" stacking sequence would once again lie on the bounds of the action space. Instead, we keep the action space as [0, 180) and use E_y as a proxy variable to demonstrate that both our DRL model and the genetic algorithm can optimize for arbitrary mechanical properties where the solution set does not lie on the bounds of the action space. Maximizing transverse composite plate stiffness, E_y, is more challenging (both for our DRL model and the genetic algorithm) than maximizing longitudinal composite plate stiffness. The DRL model and genetic algorithm successfully learn to move to ~90 degrees in each layer, which is not on the bounds of the action space. Since the optimal stacking sequence the RL model learns is not on the bound of the action space, it takes longer to optimize rewards and learn the best stacking sequence (as can be seen in Figures 5 and 6).

Figures 5 and 6 show the reward graphs for 3 and 9 ply E_y maximization; the longitudinal stiffness (in red) optimizes almost immediately, well under 100 steps, while the transverse stiffness (in blue) optimizes within ~300 steps, including 200 steps in which it does not learn but just explores the action space. Our DRL model and the genetic algorithm (NSGA-II) termination condition (3), with the time set to match the DRL model time, both optimize E_y with performance comparisons shown in Table 4. The DRL model is still the better model, but the gap in performance is appreciably reduced.

To compare the DRL and NSGA-II results for E_y, we run both models for 7 seconds for the 3 ply laminate case and 16

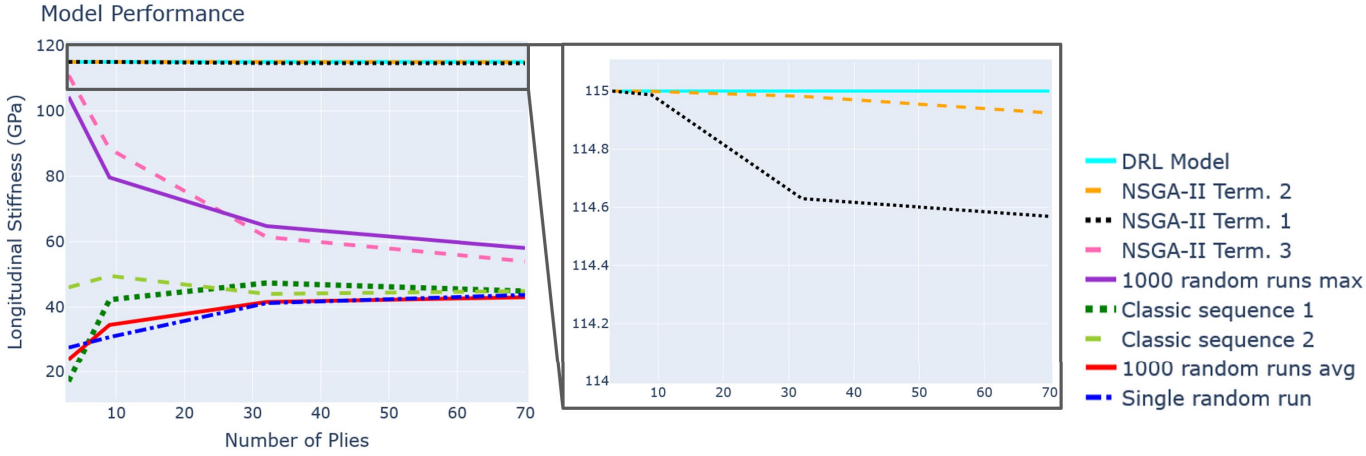


Fig. 2: Left: Graph of longitudinal stiffness of the DRL model, genetic algorithm (NSGA-II) and baseline models. Our DRL model has the best performance, with NSGA-II termination condition 2 (stiffness-based termination condition) and NSGA-II termination condition 1 (orientation angle-based termination condition) having the next best performances. The best stacking sequence of 1,000 random runs and NSGA-II termination condition 3 (timed termination set to match how long the deep reinforcement learning model takes) perform in the middle. The "classic" stacking sequences, average random stacking sequence, and single random run perform poorly. Right: Zoomed-in portion of graph showing performance comparison between the best models: our DRL model and NSGA-II termination conditions (1) and (2).

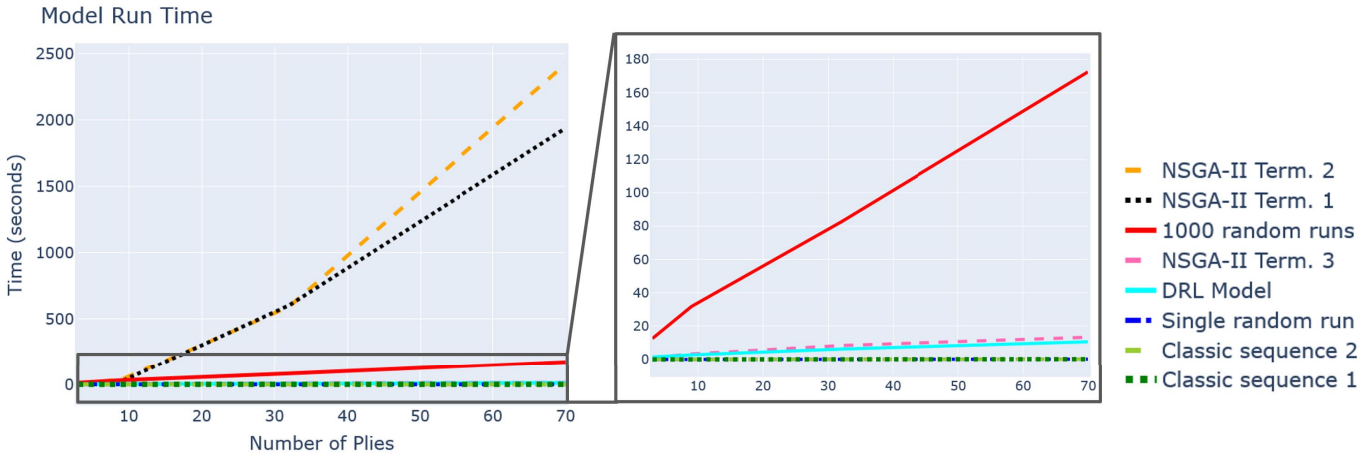


Fig. 3: Left: Time comparison of the DRL model, genetic algorithm (NSGA-II) and baseline models. Our DRL model runs significantly faster than the genetic algorithms with performance-based termination conditions (NSGA-II orientation angle-based termination & NSGA-II stiffness-based termination), as well as 1000 random runs. Right: Zoomed-in portion of graph showing time comparison between 1,000 random runs, DRL model, single random run, and "classic" stacking sequences on right. The termination condition for NSGA-II Term. 3 is the amount of time it takes our DRL model to run in order to provide the best possible direct comparison. The "classic" stacking sequences and single random run are the only models that take less time than our DRL model and NSGA-II Term. 3.

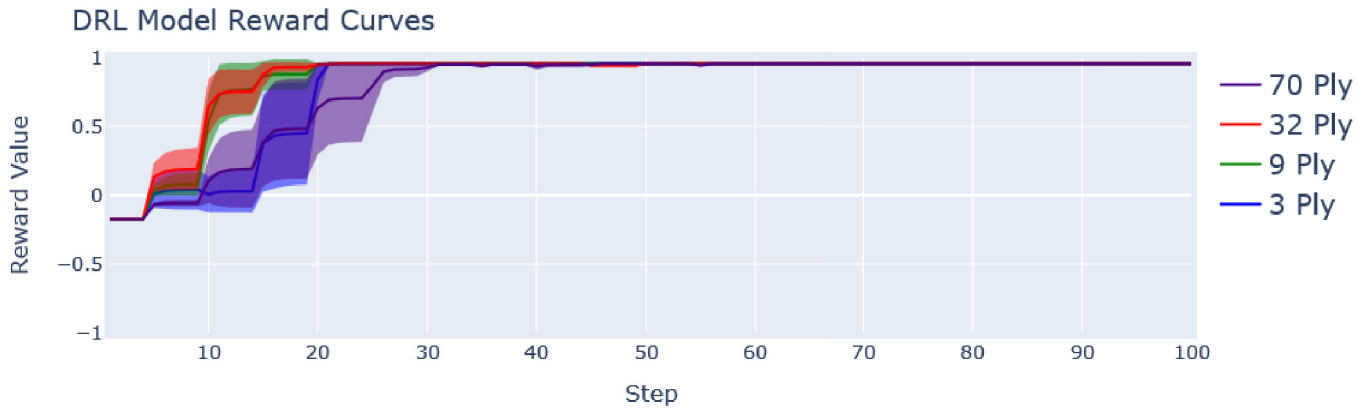


Fig. 4: DRL Model reward curve with all learning starting at time step zero. Curves converge to maximum possible reward ~0.96, which corresponds to $E_x = 115GPa$. Error bounds calculated from running each model five times and plotting one standard deviation around mean.

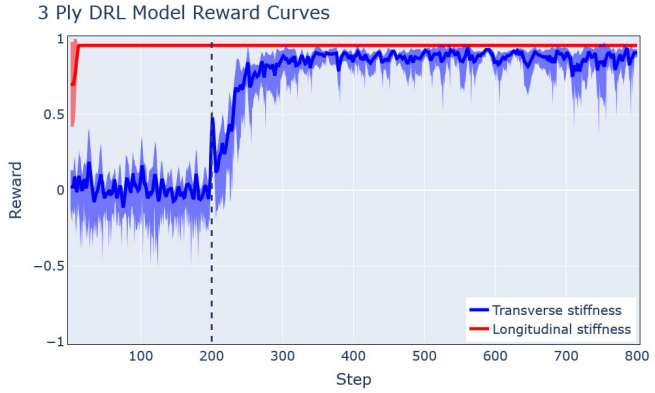


Fig. 5: Comparison of E_x (longitudinal) reward curve to E_y (transverse) reward curve on the 3 ply case showing that while optimizing E_y takes longer because the optimum is not on the bound of the action space it can achieve similar results to E_x. Black vertical line denotes where E_y starts learning.

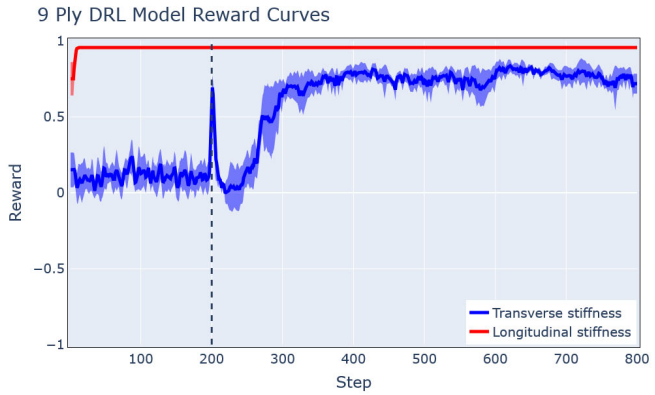


Fig. 6: Comparison of E_x (longitudinal) reward curve to E_y (transverse) reward curve on the 9 ply case. Black vertical line denotes where E_y starts learning.

seconds for the 9 ply laminate case. We determine these run times by the time it takes the DRL model to learn a significant amount, measured by the reward leveling off after its initial increase. We report these results in Table 4. The DRL model still performs better than the NSGA-II model.

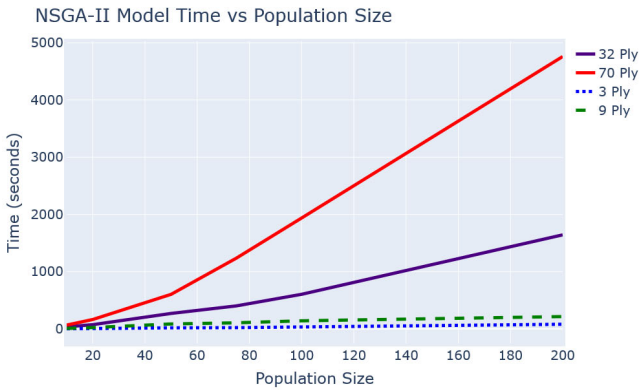
Table 4: Transverse Stiffness Results Comparison with fixed runtime

Number of Plies (n)	3	3	9	9
Variable	NSGA-II	DRL	NSGA-II	DRL
Transverse stiffness ($E_y(GPa)$)	112.90	114.31	103.32	108.80

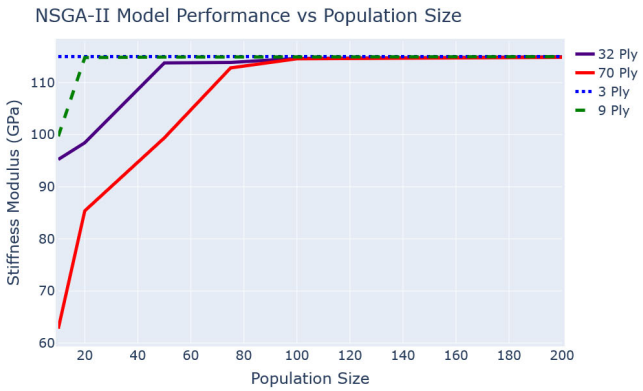
4.3. Model parameter tuning

Model parameter tuning is an important aspect of DRL and genetic algorithm performance — without proper tuning model performance may be much lower and/or run time may be much higher than under ideal parameters. Without proper tuning it is impossible to accurately compare models. For models where performance and/or time is sensitive to model parameters, tuning for the best parameters can be time consuming. This is a deterrent for engineers hoping to use optimization tools. We tune our DRL model and the genetic algorithm both to optimize model performance and also to determine model sensitivity to tuning parameters. We tune the NSGA-II genetic algorithm over population size, as shown in Figure 7. As expected, performance increases with increasing population size, but so does run time. All plots and discussion of NSGA-II results use the ~optimum population size, which we determine from tuning the parameters (e.g. for NSGA-II Term. 1: 3 plies = population size of 10, 9 plies = population size of 20, 32 & 70 plies = population size of 100, see Figure 7) to ensure a fair comparison between algorithms.

Although DRL models can be tuned for total time steps, learning start time, learning rate, and episode length, among other parameters, for longitudinal stiffness the results do not change significantly with tuning parameters. This means that engineers do not need to waste a significant amount of time tuning the DRL model for E_x. The number of time steps needs to be large enough for the model to converge. For E_x with most learning rates, 50 time steps is enough for the model to converge, as can be seen in Figure 4. Increasing the learning start time, the



(a) NSGA-II Term. 1 population size tuning time results.



(b) NSGA-II Term. 1 population size tuning stiffness results.

Fig. 7: Optimizing NSGA-II model Term. 1 performance maximizing E_x.

time step that learning starts at, does not increase model performance, but it does increase the total model run time because the model must wait longer to begin learning from the data. Otherwise the model performs well with many learning rates from 0.0005–0.1 and episode lengths 1–10. Table 5 details model performance over a range of learning rates.

For E_y, the model requires more tuning, but the default parameters still work well. We use a low learning rate, 0.0005 in the results. The model has some flexibility with learning rate, but above 0.001 model performance decreases (as can be seen in Table 6). We begin learning at time step 200, although the model is somewhat insensitive to what time step learning begins (as can be seen in Table 6). We use an episode length of 5, but the model also works well for other episode lengths. For E_y, the model takes closer to 400 time steps to converge. Table 6 summarizes the model performance for a three ply laminate over a range of learning rates and learning start times. All plots of the DRL models use the best range of parameters specified in this section.

4.4. Implementation details

Code was written in Python and run using Google Colab Pro+ using the GPU hardware accelerator, standard GPU class, and standard Runtime shape.

5. Future Work

This research is the first to apply DRL to optimize composite laminate stacking sequence. Our work opens many new directions for research. Our DRL model is set up as follows: the state is the current normalized stiffness modulus, the action is to select a new stacking sequence, and state-action pairs are evaluated with an offset, normalized stiffness as the reward function. This model works well due to its ability to change stacking sequence quickly. This is one way to define the DRL system. Future research could explore alternative definitions, for example, an RL algorithm where the state is the current stacking sequence, the action is to change one ply layer fiber orientation, and the reward is normalized stiffness. Another direction for future research is designing a custom DRL model specifically made for optimizing composite stacking sequence.

Since this is the first DRL research in this direction, we limited the optimization properties to two single-objective cases. This work could be extended to optimizing for a greater selection of mechanical and non-mechanical properties, as well as optimizing for multiple properties simultaneously. In addition to optimizing for multiple properties, another future direction is to optimize for multiple independent loading cases, as well as uncertainty in the loading cases.

Finally, the input could be designed to more fully represent complex composite parts, for instance, representing specific part properties and including additional composite part attributes.

6. Conclusion

In this work, we present a method to use deep reinforcement learning to model the optimal stacking sequence for fiber reinforced polymer composite laminates. The major contributions of this work are a DRL algorithm that successfully optimizes stacking sequence for composite plate stiffness, as well as a comparison to a commonly used genetic algorithm (NSGA-II) and two baseline models.

We show that optimal stacking sequence can be predicted using DRL to maximize longitudinal Young's modulus. Our DRL model successfully maximizes the stiffness modulus to $115GPa$ and achieves stiffness values 5.7, 6.5, 10.7, and 15.5 standard deviations above the average random stacking sequence for 3, 9, 32, and 70 plies respectively. Our DRL model performs significantly better than the maximum of 1,000 random runs. The NSGA-II algorithm optimizes the longitudinal stiffness modulus to within 0.5%, while our DRL algorithm 100% optimizes the longitudinal stiffness modulus. Although this performance advantage of our DRL model over the NSGA-II model on this relatively simple objective is small, as the optimization objective complexity increases, one would expect the performance advantage to increase as well. The running time of our DRL

Table 5: Summary of longitudinal stiffness tuning results for determining optimum learning rate (LR). The table reports the number of steps to 100% convergence. All reported values are for learning starts (LS) at time zero, because although the results converge with larger LS values, the algorithm takes longer to converge. We cap the model at 400 total time steps. We use an episode length of five. Learning rate (LR) varied in table. Results show the model is relatively insensitive to hyperparameter tuning for the case of longitudinal stiffness.

Plies (n)	$LR = 0.1$	$LR = 0.05$	$LR = 0.01$	$LR = 0.005$	$LR = 0.003$	$LR = 0.001$	$LR = 0.0005$
3	< 50	< 50	< 50	< 50	< 50	< 50	< 50
9	< 50	< 50	< 50	< 50	< 50	< 50	< 50
32	< 50	< 50	< 50	< 50	< 50	$\sim 50 - 400$	$\sim 50 - 400$
70	< 50	< 50	< 50	< 50	< 50	$\sim 50 - 400+$	$\sim 50 - 400+$

Table 6: Summary of transverse stiffness tuning results for determining optimum learning start time (LS) and learning rate (LR). The table reports the average and maximum reward out of five runs (maximum possible reward 0.96). Learning starts (LS) and learning rate (LR) varied in table. We cap the model at 400 total time steps. We use an episode length of five. Only in the best cases does transverse stiffness achieve 100% convergence, but it achieves relatively high convergence in many cases. $LS = 200$ & $LR = 0.001$ or 0.0005 get the best results with $r_{avg} = 0.95$, 99% convergence, and $r_{max} = 0.96$, 100% convergence.

LS	$LR = 0.01$		$LR = 0.005$		$LR = 0.003$		$LR = 0.001$		$LR = 0.0005$	
	r_{avg}	r_{max}	r_{avg}	r_{max}	r_{avg}	r_{max}	r_{avg}	r_{max}	r_{avg}	r_{max}
100	0.66	0.73	0.68	0.89	0.94	0.96	0.93	0.95	0.94	0.95
200	0.73	0.78	0.81	0.95	0.85	0.95	0.95	0.96	0.95	0.96
300	0.95	0.95	0.95	0.95	0.86	0.95	0.77	0.84	0.80	0.85

model is far faster than the NSGA-II model, up to orders of magnitude faster (e.g. on the largest number of plies).

Preliminary results show that our DRL model can also optimize more complex objective variables. In this work, we demonstrate optimizing transverse stiffness as one example. Although it takes longer for our DRL model and genetic algorithm to optimize transverse stiffness, the DRL model still optimizes transverse stiffness faster than the genetic algorithm.

Acknowledgements

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. We thank Prof. Grace Gu and Prof. Sergey Levine for the useful background on modeling composites and deep reinforcement learning in their respective courses. We thank Justin Meserve and Bowen Zheng for useful discussions.

References

[1] Almeida Jr, J.H.S., Ribeiro, M.L., Tita, V., Amico, S.C., 2017. Stacking sequence optimization in composite tubes under internal pressure based on genetic algorithm accounting for progressive damage. Composite Structures 178, 20–26.

[2] An, H., Chen, S., Huang, H., 2019. Stacking sequence optimization and blending design of laminated composite structures. Structural and Multidisciplinary Optimization 59, 1–19.

[3] Blank, J., Deb, K., 2020. pymoo: Multi-objective optimization in python. IEEE Access 8, 89497–89509.

[4] Bloomfield, M.W., 2010. Efficient optimization of laminated composites. Ph.D. thesis. University of Bristol.

[5] Cai, R., Jin, T., 2018. The effect of microstructure of unidirectional fibre-reinforced composites on mechanical properties under transverse loading: A review. Journal of Reinforced Plastics and Composites 37, 1360–1377.

[6] Cardozo, S.D., Gomes, H., Awruch, A., et al., 2011. Optimization of laminated composite plates and shells using genetic algorithms, neural networks and finite elements. Latin American Journal of Solids and Structures 8, 413–427.

[7] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE transactions on evolutionary computation 6, 182–197.

[8] Diaconu, C.G., Sekine, H., 2004. Layup optimization for buckling of laminated composite shells with restricted layer angles. AIAA journal 42, 2153–2163.

[9] Erdal, O., Sonmez, F.O., 2005. Optimum design of composite laminates for maximum buckling load capacity using simulated annealing. Composite Structures 71, 45–52.

[10] Fukunaga, H., Vanderplaats, G.N., 1991. Stiffness optimization of orthotropic laminated composites using lamination parameters. AIAA journal 29, 641–646.

[11] Gemi, L., 2018. Investigation of the effect of stacking sequence on low velocity impact response and damage formation in hybrid composite pipes under internal pressure. a comparative study. Composites Part B: Engineering 153, 217–232.

[12] Ghiasi, H., Fayazbakhsh, K., Pasini, D., Lessard, L., 2010. Optimum stacking sequence design of composite materials part ii: Variable stiffness design. Composite structures 93, 1–13.

[13] Ghiasi, H., Pasini, D., Lessard, L., 2009. Optimum stacking sequence design of composite materials part i: Constant stiffness design. Composite Structures 90, 1–11. doi:10.1016/j.compstruct.2009.01.006.

[14] Grenestedt, J., Gudmundson, P., 1993. Layup optimization of composite material structures. Optimal design with advanced materials , 311–336.

[15] Haftka, R.T., Walsh, J.L., 1992. Stacking-sequence optimization for buckling of laminated plates by integer programming. AIAA journal 30, 814–819.

[16] Hajmohammad, M., Salari, M., Hashemi, S., Esfe, M.H., 2013. Optimization of stacking sequence of composite laminates for optimizing buckling load by neural network and genetic algorithm. Indian Journal of Science and Technology 6, 5070–7.

[17] Harris, C.E., Morris, D.H., 1985. An evaluation of the effects of stacking sequence and thickness on the fatigue life of quasi-isotropic graphite/epoxy laminates. ASTM International.

[18] Herencia, J., Weaver, P., Friswell, M., 2007. Local optimisation of anisotropic composite panels with t shape stiffeners, in: 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, p. 2217.

[19] Irisarri, F.X., Bassir, D.H., Carrere, N., Maire, J.F., 2009. Multiobjective stacking sequence optimization for laminated composite structures. Composites Science and Technology 69, 983–990.
[20] Jing, Z., Fan, X., Sun, Q., 2015a. Global shared-layer blending method for stacking sequence optimization design and blending of composite structures. Composites Part B: Engineering 69, 181–190.
[21] Jing, Z., Fan, X., Sun, Q., 2015b. Stacking sequence optimization of composite laminates for maximum buckling load using permutation search algorithm. Composite Structures 121, 225–236.
[22] Jones, R.M., 2018. Mechanics of composite materials. CRC press.
[23] Kameyama, M., Fukunaga, H., 2007. Optimum design of composite plate wings for aeroelastic characteristics using lamination parameters. Computers & structures 85, 213–224.
[24] Lansing, W., Dwyer, W., Emerton, R., Ranalli, E., 1971. Application of fully stressed design procedures to wing and empennage structures. Journal of Aircraft 8, 683–688.
[25] Li, C., Zheng, P., Yin, Y., Wang, B., Wang, L., 2023. Deep reinforcement learning in smart manufacturing: A review and prospects. CIRP Journal of Manufacturing Science and Technology 40, 75–101.
[26] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 .
[27] Lin, C.C., Lee, Y.J., 2004. Stacking sequence optimization of laminated composite structures using genetic algorithm with local improvement. Composite structures 63, 339–345.
[28] Marques, F., Natarajan, S., Ferreira, A., 2017. Evolutionary-based aeroelastic tailoring of stiffened laminate composite panels in supersonic flow regime. Composite Structures 167, 30–37.
[29] Miki, M., 1982. Material design of composite laminates with required in-plane elastic properties. Progress in science and engineering of composites 2, 1725–1731.
[30] Moh, J.S., Hwu, C., 1997. Optimization for buckling of composite sandwich plates. AIAA journal 35, 863–868.
[31] Mohanavel, V., Suresh Kumar, S., Vairamuthu, J., Ganeshan, P., NagarajaGanesh, B., 2021. Influence of stacking sequence and fiber content on the mechanical properties of natural and synthetic fibers reinforced pentalayered hybrid composites. Journal of Natural Fibers , 1–13.
[32] Nagendra, S., Haftka, R.T., Gurdal, Z., 1992. Stacking sequence optimization of simply supported laminates with stability and strain constraints. AIAA journal 30, 2132–2137.
[33] Nettles, A., 1994. Mechanics of laminated composites, in: NASA Ref. Publ. Basic Mech. Laminated Compos. Plates. National Aeronautics and Space Administration Washington, DC, pp. 11–23.
[34] Pagano, N., Pipes, R.B., 1971. The influence of stacking sequence on laminate strength. Journal of composite materials 5, 50–57.
[35] Park, J., Hwang, J., Lee, C., Hwang, W., 2001. Stacking sequence design of composite laminates for maximum strength using genetic algorithms. Composite Structures 52, 217–231.
[36] Rajak, D.K., Pagar, D.D., Menezes, P.L., Linul, E., 2019. Fiber-reinforced polymer composites: Manufacturing, properties, and applications. Polymers 11, 1667.
[37] del Rosario, Z., Fenrich, R.W., Iaccarino, G., 2019. Margin as model: Some answers to" how many tests should i perform?", in: AIAA Aviation 2019 Forum, p. 3554.
[38] Sandhu, R., 1971. Parametric study of optimum fiber orientation for filamentary sheet. Technical Report. AIR FORCE FLIGHT DYNAMICS LAB WRIGHT-PATTERSON AFB OH.
[39] Şerban, A., 2016. Fast and robust matlab-based finite element model used in the layup optimization of composite laminates, in: IOP Conference Series: Materials Science and Engineering, IOP Publishing. p. 012103.
[40] Setoodeh, S., Abdalla, M.M., Gürdal, Z., 2006. Design of variable–stiffness laminates using lamination parameters. Composites Part B: Engineering 37, 301–309.
[41] Sui, F., Guo, R., Zhang, Z., Gu, G.X., Lin, L., 2021. Deep reinforcement learning for digital materials design. ACS Materials Letters 3, 1433–1439.
[42] Todoroki, A., Haftka, R.T., 1998. Stacking sequence optimization by a genetic algorithm with a new recessive gene like repair strategy. Composites Part B: Engineering 29, 277–285.
[43] Tsai, S.W., Hahn, H.T., 2018. Introduction to composite materials. Routledge.
[44] Wang, Z., Sobey, A., 2020. A comparative review between genetic algorithm use in composite optimisation and the state-of-the-art in evolutionary computation. Composite Structures 233, 111739.
[45] van de Werken, N., Tekinalp, H., Khanbolouki, P., Ozcan, S., Williams, A., Tehrani, M., 2020. Additively manufactured carbon fiber-reinforced composites: State of the art and perspective. Additive Manufacturing 31, 100962.
[46] Zheng, B., Zheng, Z., Gu, G.X., 2022. Designing mechanically tough graphene oxide materials using deep reinforcement learning. npj Computational Materials 8, 225.