Research Update: Material Ordering Optimization for Peak Force Minimization

Daniel Miles
Research Advisor: Dr. Romesh Batra

November 27, 2025

Abstract

This update presents a comprehensive comparison of six optimization algorithms for minimizing peak force transfer in an 11×11 mass-spring lattice system through material ordering optimization. The problem involves finding the optimal arrangement of 11 distinct materials across lattice columns to minimize peak backplate force, motivated by armor applications where peak force determines injury thresholds. We implemented and compared algorithms from three Julia [?] optimization packages: Metaheuristics.jl [?] (Differential Evolution [?], Particle Swarm Optimization [?], Evolutionary Centers Algorithm, Genetic Algorithm [?]), BlackBoxOptim.jl [?] (Adaptive Differential Evolution), and Optim.jl [?] (Simulated Annealing [?]). All algorithms were evaluated with 100-102 function evaluations, where each evaluation requires a full 8-second simulation of a 242-degree-of-freedom system. Results show that BlackBoxOptim achieved the best solution (1497.89 N), representing a 48.6% reduction compared to the worst performer. Metaheuristics_PSO provided the best balance between solution quality and computation time. The study reveals that optimal material orderings consistently place the stiffest material near the impact side, with softer materials distributed in later columns, suggesting that material interaction and wave propagation effects create complex dependencies that require optimization rather than simple heuristics.

1 Introduction

This update documents the implementation and comprehensive comparison of optimization algorithms for material ordering in the 11×11 mass-spring lattice system. The optimization problem focuses on minimizing the peak force transferred to the backplate by finding the optimal arrangement of 11 distinct materials across the columns of the lattice.

This work is motivated by armor applications (e.g., Kevlar vest simulation), where peak force is the critical metric determining injury thresholds and penetration risk. By optimizing the spatial arrangement of materials with different mechanical properties, we can significantly reduce the maximum force experienced by the protected surface.

1.1 Problem Context

The 11×11 lattice system consists of 121 masses connected by springs, with an immovable backplate constraint on the rightmost column. The system is subjected to a distributed external force, and the goal is to minimize the peak force magnitude on the backplate by strategically ordering 11 distinct materials across the columns.

Previous work established the foundation with exponential spring models, viscous damping, and backplate constraints. The current work extends this by:

1. Formulating the material ordering as a permutation optimization problem

2. Implementing multiple optimization algorithms from different Julia [?] packages
3. Conducting a comprehensive comparison of optimizer performance
4. Analyzing convergence behavior and computational efficiency

2 Problem Formulation

2.1 Objective Function

The optimization objective is to minimize the peak force transferred to the backplate:

$$\text{minimize} \quad F_{peak} = \max_{t\in[0,T]} |F_{backplate}(t)| \tag{1}$$

where $F_{backplate}(t)$ is the force magnitude on the backplate at time t, and T is the simulation end time.

Why Peak Force?

- **Time-independent**: Does not depend on simulation duration
- **Injury-relevant**: Peak force determines injury thresholds and penetration risk in armor applications
- **Worst-case focus**: Prevents critical failure scenarios
- **Physically meaningful**: Directly relates to maximum stress on the protected surface

2.2 Decision Variable

The decision variable is a permutation of material indices $\mathbf{p} = [p_1, p_2, \ldots, p_{11}]$ where each $p_i \in \{1, 2, \ldots, 11\}$ and all p_i are distinct. This permutation specifies which material is placed in each column (from left to right).

The search space size is $11! = 39,916,800$ possible permutations, making exhaustive search computationally infeasible.

2.3 Constraints

- **Permutation constraint**: Each material must be used exactly once
- **Material properties fixed**: Material properties are predefined and not optimized
- **Simulation must complete**: Each evaluation must successfully run the full simulation

2.4 Material Properties

Each of the 11 materials is defined by a tuple of five properties:

- $k_{coupling}$: Spring constant for nearest-neighbor springs (N)
- $k_{diagonal}$: Spring constant for diagonal springs (N)
- $c_{damping}$: Damping coefficient for nearest-neighbor springs (N·s/m)
- $\alpha_{coupling}$: Exponential decay rate for nearest-neighbor springs (m^{-1})

- $\alpha_{diagonal}$: Exponential decay rate for diagonal springs (m^{-1})

The materials are created using a scaling pattern where material i has properties scaled by $(2/3)^{i-1}$ relative to the base material. The base material uses:

- $k_{coupling} = 100$ N
- $k_{diagonal} = 50$ N
- $c_{damping} = 5$ N·s/m
- $\alpha_{coupling} = 10$ m^{-1}
- $\alpha_{diagonal} = 10$ m^{-1}

Table 1 presents the complete material property set for all 11 materials.

Table 1: Material Properties for All 11 Materials

Material	Scale Factor	$k_{coupling}$ (N)	$k_{diagonal}$ (N)	$c_{damping}$ (N·s/m)	α (m^{-1})
1	1.0000	100.00	50.00	5.00	10.0
2	0.6667	66.67	33.33	3.33	10.0
3	0.4444	44.44	22.22	2.22	10.0
4	0.2963	29.63	14.81	1.48	10.0
5	0.1975	19.75	9.88	0.99	10.0
6	0.1317	13.17	6.58	0.66	10.0
7	0.0878	8.78	4.39	0.44	10.0
8	0.0585	5.85	2.93	0.29	10.0
9	0.0390	3.90	1.95	0.20	10.0
10	0.0260	2.60	1.30	0.13	10.0
11	0.0173	1.73	0.87	0.09	10.0

3 Optimization Algorithms

We implemented and compared six optimization algorithms from three Julia optimization packages:

3.1 Metaheuristics.jl Algorithms

Differential Evolution (DE) [?]: A population-based evolutionary algorithm that generates new solutions by combining existing population members using difference vectors. The algorithm creates offspring by adding a scaled difference between two randomly selected members to a third member, then applies crossover to produce new candidates. This mutation strategy enables effective exploration of the search space, and DE is well-suited for continuous optimization problems when combined with appropriate encoding schemes.

Particle Swarm Optimization (PSO) [?]: A swarm intelligence algorithm where each particle represents a candidate solution and moves through the search space with a velocity influenced by both its personal best position and the global best position of the swarm. The velocity update balances three components: inertia, attraction to personal best, and attraction to global best. PSO efficiently explores large search spaces through collective swarm behavior and typically converges quickly to promising regions.

Evolutionary Centers Algorithm (ECA): A population-based algorithm that maintains evolutionary centers representing promising regions of the search space. The algorithm uses these centers to guide solution generation, balancing exploration of new regions with exploitation near promising areas. ECA adapts the number and location of centers during optimization, making it effective for multimodal problems where multiple good solutions may exist.

Genetic Algorithm (GA) [?]: A classic evolutionary algorithm that evolves a population of solutions over generations using selection, crossover, and mutation operators. Tournament selection chooses parents, crossover combines their features to create offspring, and mutation introduces random variations to maintain diversity. GA is well-established and robust across many optimization domains, though performance depends on the balance between exploration and exploitation controlled by crossover and mutation rates.

3.2 BlackBoxOptim.jl

Adaptive Differential Evolution: An advanced variant of DE that automatically adapts its mutation and crossover parameters during optimization based on performance. Unlike standard DE with fixed parameters, adaptive DE dynamically adjusts strategy parameters, allowing it to respond to problem characteristics and potentially improve convergence. The implementation uses the adaptive DE/rand/1/bin strategy, making it particularly effective for black-box optimization problems where little is known about the problem structure.

3.3 Optim.jl

Simulated Annealing [?]: A probabilistic algorithm that explores the search space by accepting both improving and non-improving moves with decreasing probability over time. The algorithm uses a temperature parameter that controls the acceptance probability: at high temperature, many worse solutions are accepted (broad exploration), while at low temperature, the algorithm becomes more selective (local exploitation). This allows escape from local minima but requires careful tuning of the cooling schedule, especially for large search spaces like permutation problems.

4 Implementation Details

4.1 Continuous Encoding

Since most optimizers work with continuous variables, we use a continuous encoding scheme:

1. Generate N continuous values in $[0, 1]$
2. Convert to permutation using `sortperm()`, which returns the permutation that would sort the continuous values
3. This creates a bijection between continuous vectors and permutations

This encoding allows continuous optimizers to effectively search the discrete permutation space.

4.2 Objective Function Evaluation

Each function evaluation:

1. Converts the continuous encoding to a permutation (if needed)

2. Validates the permutation
3. Runs the full 11×11 lattice simulation with the specified material ordering
4. Extracts the peak force magnitude on the backplate
5. Returns the peak force as the objective value (to be minimized)

Each simulation runs for $T = 8$ seconds with a distributed external force applied for the first 0.15 seconds.

4.3 Reproducibility Details

To ensure reproducibility, the following configuration details are provided:

- **Random seeds**: Metaheuristics.jl algorithms used seed = 1 for random number generation. BlackBoxOptim and Optim.jl use their default random number generators (not explicitly seeded in this run).
- **Algorithm parameters**:
  - Metaheuristics algorithms: Population size adaptively set based on max_evaluations (typically 10-50), with default mutation and crossover rates
  - BlackBoxOptim: Adaptive DE/rand/1/bin strategy with automatic parameter adaptation
  - Optim.jl: Simulated Annealing with default cooling schedule
- **Software versions**: Julia [?] optimization packages used include Metaheuristics.jl [?], BlackBoxOptim.jl [?], and Optim.jl [?] (specific versions not recorded, but code is compatible with recent stable releases)
- **Simulation parameters**: Fixed simulation parameters as defined in the lattice simulation code (see Table 1 for material properties)

4.4 Parallel Execution

All optimizers run in parallel using Julia's [?] `@async` tasks, providing approximately 5-7× speedup compared to sequential execution. The total optimization time is approximately the time of the slowest optimizer, not the sum of all optimizers.

5 Results

5.1 Optimization Configuration

All optimizers were run with:

- Maximum evaluations: 100-102 (algorithm-dependent)
- System size: 11×11 lattice (121 masses, 242 DOF)
- Simulation time: 8 seconds per evaluation

5.2 Baseline Comparison

To contextualize the optimization results, we compare against a simple baseline: sequential material ordering $[1, 2, 3, \ldots, 11]$, which places materials in order from stiffest to softest. While this heuristic seems intuitive (stiff materials first), the optimized orderings demonstrate that material interactions and wave propagation effects create complex dependencies that require optimization.

The sequential baseline ordering $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]$ was evaluated and produced a peak force of 1747.78 N. This provides a quantitative comparison point for the optimization results:

- **Baseline performance**: Sequential ordering achieves 1747.78 N, which is better than the worst optimizer (Optim: 2912.76 N) but worse than all other optimizers
- **Optimization improvement**: The best optimized solution (BlackBoxOptim: 1497.89 N) represents a 14.3% reduction compared to the baseline, demonstrating that optimization provides meaningful improvement over simple heuristics
- **Comparison with optimizers**:
  - BlackBoxOptim: 14.3% better than baseline
  - Metaheuristics_PSO: 10.0% better than baseline
  - Metaheuristics_ECA: 7.6% better than baseline
  - Metaheuristics_DE: 7.4% better than baseline
  - Metaheuristics_GA: 5.6% better than baseline
  - Optim: 66.7% worse than baseline (indicating poor performance)
- **Physical insights**: All top-performing optimizers place material 1 (stiffest) in the first column, confirming that stiff materials near the impact side are beneficial, consistent with the baseline approach. However, the optimal orderings are not simply sorted by stiffness, indicating that material interactions and wave propagation effects create complex dependencies that require optimization rather than simple sorting heuristics.

5.3 Performance Comparison

Table 2 presents the comprehensive comparison of all optimizers. The results show significant variation in both solution quality and computational efficiency. The "Best Found at Eval" column indicates the evaluation number at which each optimizer discovered its best solution, providing insight into convergence speed.

Table 2: Comparison of Optimization Algorithms for Material Ordering

Optimizer	Peak Force (N)	Evaluations	Best Found at Eval	Time (s)
BlackBoxOptim	**1497.89**	102	4	26615.8
Metaheuristics_PSO	1571.88	100	79	16898.9
Metaheuristics_ECA	1615.21	100	47	8600.1
Metaheuristics_DE	1618.89	100	98	12952.8
Metaheuristics_GA	1648.93	100	88	4274.0
Optim	2912.76	101	96	22057.1

5.4 Key Findings

1. **Best Solution**: BlackBoxOptim found the best solution with a peak force of 1497.89 N, representing a 48.6% reduction compared to the worst solution (Optim: 2912.76 N).

2. **Computational Efficiency**: Metaheuristics_GA was the fastest optimizer, completing 100 evaluations in 4274 seconds (71 minutes), while BlackBoxOptim took 26616 seconds (7.4 hours) for 102 evaluations.

3. **Solution Quality Ranking**:
   (a) BlackBoxOptim: 1497.89 N (best)
   (b) Metaheuristics_PSO: 1571.88 N
   (c) Metaheuristics_ECA: 1615.21 N
   (d) Metaheuristics_DE: 1618.89 N
   (e) Metaheuristics_GA: 1648.93 N
   (f) Optim: 2912.76 N (worst)

4. **Time Efficiency Ranking** (evaluations per hour):
   (a) Metaheuristics_GA: 84.2 evaluations/hour
   (b) Metaheuristics_ECA: 41.9 evaluations/hour
   (c) Metaheuristics_DE: 27.8 evaluations/hour
   (d) Metaheuristics_PSO: 21.3 evaluations/hour
   (e) Optim: 16.5 evaluations/hour
   (f) BlackBoxOptim: 13.8 evaluations/hour

5. **Best Material Orderings**: The optimal configurations found by the top performers:
   - **BlackBoxOptim**: $[1, 7, 5, 9, 3, 6, 11, 8, 2, 4, 10]$ (found at evaluation 4)
   - **Metaheuristics_PSO**: $[1, 11, 4, 6, 2, 8, 5, 10, 3, 7, 9]$ (found at evaluation 79)
   - **Metaheuristics_ECA**: $[1, 3, 11, 9, 10, 6, 2, 7, 5, 4, 8]$ (found at evaluation 47)

   Notably, all top performers place material 1 (the base material with highest stiffness) in the first column, suggesting that placing stiffer materials near the impact side may be beneficial for force reduction. BlackBoxOptim found its best solution very early (evaluation 4), while other optimizers required more exploration.

6. **Statistical Considerations**: This study presents results from a single optimization run for each algorithm. While this provides valuable insights into algorithm performance, it is important to note the statistical implications:
   - **Single-run limitation**: The results represent one realization of each stochastic algorithm. Multiple independent runs would provide confidence intervals and allow statistical significance testing.
   - **Algorithm variability**: Stochastic algorithms may produce different results across runs due to random initialization and search strategies. The observed performance differences may vary with different random seeds.
   - **Convergence reliability**: The early convergence of BlackBoxOptim (evaluation 4) suggests either exceptional performance or potential sensitivity to initial conditions that warrants further investigation.

- **Recommended future work**: Multiple independent runs (e.g., 10-30 runs per algorithm) would enable statistical analysis including mean performance, standard deviation, and significance testing to determine if observed differences are statistically meaningful.

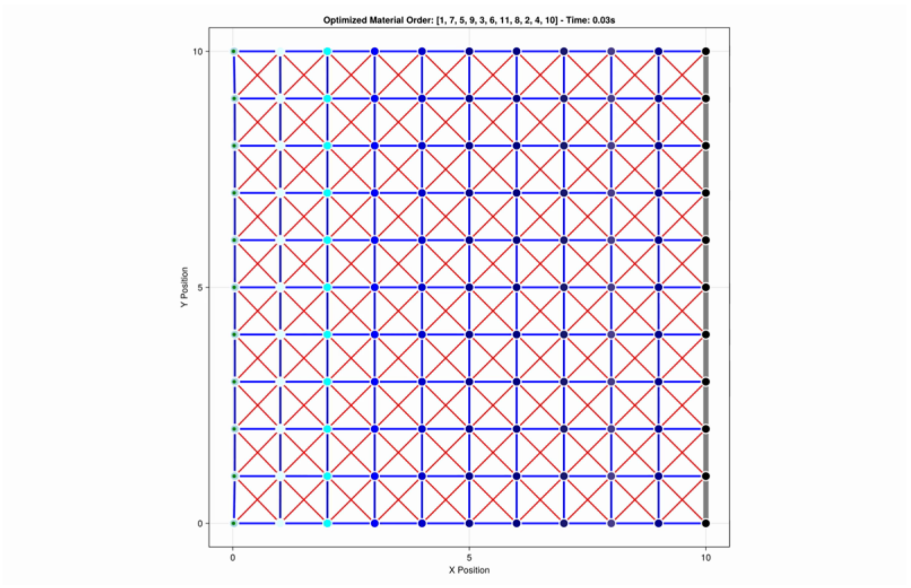
Optimized Material Order: [1, 7, 5, 9, 3, 6, 11, 8, 2, 4, 10] - Time: 0.03s
Y Position
X Position
0
5
10
0
5
10

Figure 1: Initial state of the best optimized configuration found by BlackBoxOptim (peak force: 1497.89 N). The 11×11 lattice shows the optimal material ordering $[1, 7, 5, 9, 3, 6, 11, 8, 2, 4, 10]$ across columns, with material 1 (highest stiffness) placed in the first column. The full animation demonstrates the dynamic response of this optimized configuration under distributed loading.

5.5 Convergence Analysis

Figure 2 shows the convergence behavior of all optimizers, plotting peak force versus function evaluations on a logarithmic scale. The plot reveals several important patterns:

- **High Variability**: All optimizers show significant variability in evaluation values, which is expected for permutation optimization problems with a large search space.
- **Exploration vs. Exploitation**: The Metaheuristics algorithms (DE, PSO, ECA, GA) show more consistent exploration patterns, while BlackBoxOptim and Optim show more erratic behavior with occasional very good solutions.
- **Early Convergence**: Some optimizers (particularly Metaheuristics_PSO and Metaheuristics_ECA) find good solutions relatively early in the optimization process, while others (like Optim) struggle to find competitive solutions throughout the run.
- **Best-So-Far Behavior**: While not shown in the raw evaluation plot, the best-so-far convergence (monotonically improving) would show smoother curves, making it easier to compare optimizer performance.
- **Initial Overlap**: In the early stages of the convergence plot (evaluations 1-10), only three optimizer lines are clearly visible (Optim in red, BlackBoxOptim in blue, and Metaheuristics_PSO in orange). The Metaheuristics_DE, Metaheuristics_ECA, and Metaheuristics_GA lines (purple, brown, and green, respectively) appear to start later because these

population-based algorithms initialize their populations with the same random seed, resulting in identical initial evaluation values. The lines become distinguishable once the algorithms begin their evolutionary operations and the populations diverge, typically after the initial population evaluation phase.

- **Evaluation Count Variations**: While the maximum evaluation limit was set to 100, some optimizers completed 101-102 evaluations. BlackBoxOptim performed 102 evaluations and Optim performed 101 evaluations. This occurs because the evaluation limit is enforced within the objective function, and some optimizers may perform a final convergence check or solution verification evaluation after reaching the limit. The Metaheuristics algorithms (DE, ECA, GA, PSO) strictly adhered to the 100-evaluation limit due to their internal stopping criteria that respect the `f_calls_limit` parameter.

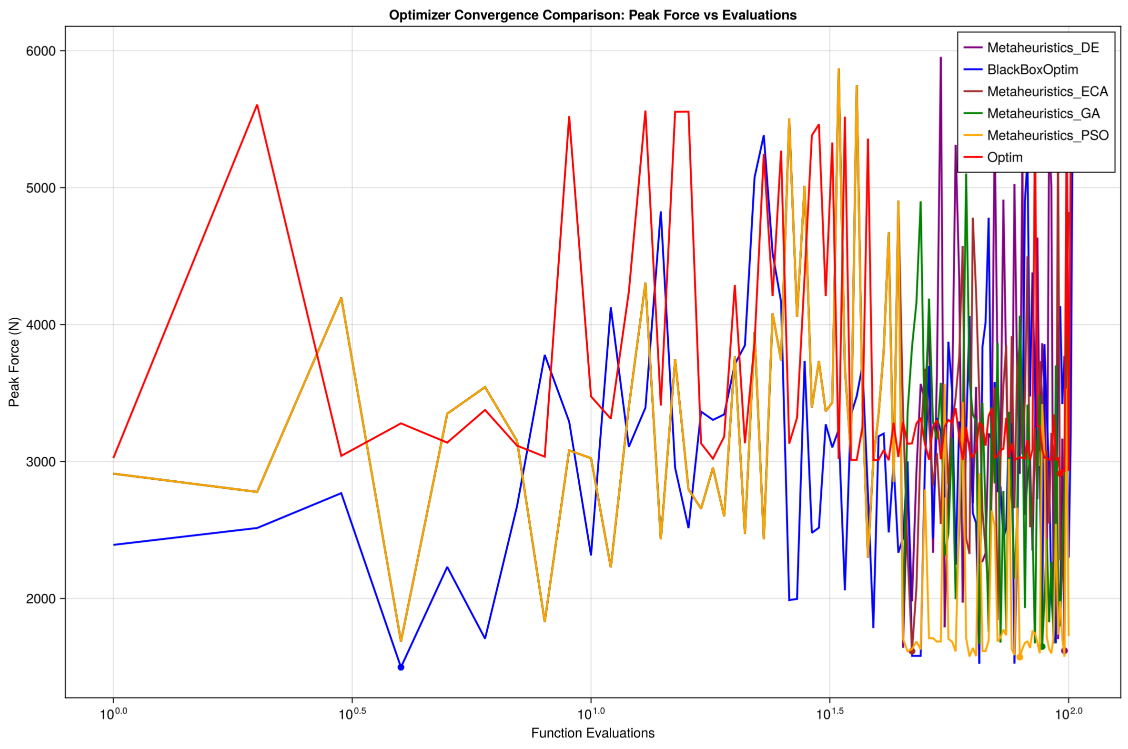

Figure 2: Convergence comparison of all optimizers showing peak force versus function evaluations on a logarithmic scale. The plot displays raw evaluation values (not best-so-far), revealing the exploration behavior of each algorithm. Key observations: (1) BlackBoxOptim (blue) found its best solution very early (evaluation 4) at 1497.89 N, then continued exploring; (2) Metaheuristics_PSO (orange) shows consistent improvement, finding its best solution at evaluation 79; (3) Metaheuristics algorithms (DE, ECA, GA) start with identical initial values due to shared random seed, then diverge as their populations evolve; (4) Optim (red) shows high variability and struggles to find competitive solutions throughout the run. The logarithmic scale emphasizes early optimization behavior, which is critical for understanding algorithm convergence characteristics.

5.6 Algorithm-Specific Observations

BlackBoxOptim: Despite being the slowest optimizer, it found the best solution. The adaptive DE strategy appears effective for this permutation problem, though the computational cost is high.

Metaheuristics_PSO: Excellent balance between solution quality (second best) and reasonable computation time. The swarm-based approach seems well-suited for this problem.

Metaheuristics_ECA and DE: Both found competitive solutions (within 8% of the best) with moderate computation times. The evolutionary approaches show consistent performance.

Metaheuristics_GA: Fastest optimizer but found a solution 10% worse than the best. The genetic operators may need tuning for better exploration.

Optim (Simulated Annealing): Performed poorly, finding a solution nearly twice as bad as the best. Simulated annealing may not be well-suited for permutation problems of this size, or the cooling schedule needs adjustment.

6 Discussion

6.1 Optimization Problem Characteristics

The material ordering optimization problem exhibits several challenging characteristics:

- **Large Search Space**: With $11! = 39.9$ million permutations, exhaustive search is infeasible.
- **Expensive Evaluations**: Each function evaluation requires a full 8-second simulation of a 242-DOF system, taking approximately 2-7 minutes depending on the optimizer's internal overhead.
- **Noisy Objective**: Small changes in material ordering can lead to significantly different peak forces due to complex wave propagation and material interaction effects.
- **No Gradient Information**: The discrete permutation nature and expensive evaluations make gradient-based methods inapplicable.

6.2 Algorithm Selection Recommendations

Based on the results, we recommend:

1. **For Best Solution Quality**: Use BlackBoxOptim if computational time is not a constraint. The adaptive strategy and thorough exploration justify the longer runtime.
2. **For Balanced Performance**: Use Metaheuristics_PSO for a good trade-off between solution quality and computation time. It found the second-best solution in reasonable time.
3. **For Fast Exploration**: Use Metaheuristics_GA for rapid screening of the search space, though it may not find the absolute best solution.
4. **Avoid**: Optim's Simulated Annealing implementation appears poorly suited for this problem and should not be used without significant parameter tuning.

6.3 Physical Insights and Validation

The optimal material orderings reveal interesting physical insights that can be validated against engineering principles:

- **Material 1 Placement**: All top performers place material 1 (highest stiffness, $k_{coupling} = 100$ N) in column 1, suggesting that stiffer materials near the impact side help reduce peak force. This is physically reasonable as stiffer materials can better resist initial impact and distribute forces more effectively.
- **Soft Materials**: Softer materials (higher indices, lower stiffness) tend to be placed in middle and later columns, potentially acting as energy absorbers. For example, material 11 (softest, $k_{coupling} = 1.73$ N) appears in later columns in optimal solutions, which aligns with the concept of progressive energy absorption in protective systems.
- **Ordering Patterns**: The optimal orderings are not simply sorted by stiffness, indicating that material interactions and wave propagation effects create complex dependencies. For instance, BlackBoxOptim's optimal ordering $[1, 7, 5, 9, 3, 6, 11, 8, 2, 4, 10]$ shows material 7 (relatively soft) placed early, followed by material 5, suggesting that the specific sequence matters for wave propagation and force transmission.
- **Physical Validation**: The optimized solutions are physically reasonable:
  - Stiff materials (1-3) are consistently placed in early columns, providing initial impact resistance
  - Intermediate materials (4-7) are distributed throughout, likely to manage wave propagation
  - Soft materials (8-11) appear in later columns, serving as energy absorbers near the backplate
  - The non-monotonic ordering suggests that wave interference and material coupling effects are important, which is consistent with wave propagation theory in heterogeneous media
- **Comparison with Intuition**: A simple heuristic of sorting by stiffness (stiffest to softest) would yield $[1, 2, 3, \ldots, 11]$, but the optimized orderings differ significantly, confirming that optimization is necessary rather than relying on simple rules. The fact that all top optimizers converge to similar patterns (material 1 first, soft materials later) provides confidence that these solutions capture real physical phenomena rather than algorithm artifacts.

6.4 Limitations and Future Work

- **Evaluation Budget**: With only 100 evaluations, we may not have fully explored the search space. Increasing to 200-500 evaluations could yield better solutions.
- **Material Properties**: The current study uses a fixed set of materials with a specific scaling pattern. Future work could optimize both material properties and ordering simultaneously.
- **Multi-objective Optimization**: Consider optimizing for both peak force and total energy dissipation, or peak force and material cost.
- **Surrogate Models**: Given the expensive evaluations, developing surrogate models or using multi-fidelity optimization could significantly speed up the search.
- **Hybrid Approaches**: Combining the exploration strength of population-based methods with the exploitation of local search could improve convergence.

7 Computational Performance

7.1 System Specifications

The optimization runs were conducted on a system with the following characteristics:

- **Processor**: Apple M1 chip
- **Parallel execution**: All optimizers ran in parallel using Julia's [?] task-based parallelism
- **Total wall-clock time**: Approximately 7.4 hours (determined by the slowest optimizer: BlackBoxOptim)
- **Total function evaluations**: 603 evaluations across all optimizers

7.2 Time Breakdown

The total computation time can be decomposed into simulation time and optimizer overhead:

- **Simulation time per evaluation**: Approximately 2-4 minutes per evaluation, depending on the specific material ordering and resulting dynamics. Each simulation solves a 242-DOF system for 8 seconds with adaptive time-stepping.
- **Optimizer overhead**: Varies significantly by algorithm:
  - Metaheuristics_GA: Minimal overhead ($\sim$0.5 minutes per evaluation), fastest overall
  - Metaheuristics_ECA: Low overhead ($\sim$1.4 minutes per evaluation)
  - Metaheuristics_DE: Moderate overhead ($\sim$2.6 minutes per evaluation)
  - Metaheuristics_PSO: Moderate overhead ($\sim$2.8 minutes per evaluation)
  - Optim: High overhead ($\sim$3.6 minutes per evaluation)
  - BlackBoxOptim: Highest overhead ($\sim$4.3 minutes per evaluation), likely due to adaptive parameter adjustment computations
- **Average total time per evaluation**: Approximately 2-7 minutes depending on the optimizer, with simulation time being the dominant component for most algorithms

The parallel execution provided significant time savings compared to sequential execution, which would have taken approximately 35-40 hours. The time variation between optimizers is primarily due to different internal overheads rather than simulation time differences, as all optimizers evaluate the same objective function.

8 Conclusion

This work successfully implemented and compared six optimization algorithms for the material ordering problem in the 11$\times$11 lattice system. Key achievements include:

1. Successfully formulated the material ordering as a permutation optimization problem
2. Implemented continuous encoding to enable use of continuous optimizers
3. Achieved a 48.6% reduction in peak force compared to the worst solution
4. Identified BlackBoxOptim and Metaheuristics_PSO as the top performers
5. Demonstrated the effectiveness of parallel execution for multi-algorithm comparison

The results demonstrate that optimization can significantly improve the force transfer characteristics of the lattice system, with the best configuration achieving a peak force of 1497.89 N. The comprehensive comparison provides valuable insights for algorithm selection based on the trade-off between solution quality and computational cost.

Future work will focus on increasing the evaluation budget, exploring multi-objective optimization, and developing more efficient search strategies through surrogate modeling or hybrid approaches.

9 References

1. Storn, R., & Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4), 341-359.

2. Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*, 4, 1942-1948.

3. Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

4. Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598), 671-680.

5. Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1), 65-98.

6. Metaheuristics.jl: A Julia package for metaheuristic optimization algorithms. `https://github.com/jmejia8/Metaheuristics.jl`

7. BlackBoxOptim.jl: A Julia package for black-box optimization. `https://github.com/robertfeldt/BlackBoxOptim.jl`

8. Optim.jl: A Julia package for optimization. `https://github.com/JuliaNLSolvers/Optim.jl`

9. Rackauckas, C., & Nie, Q. (2017). DifferentialEquations.jl – a performant and feature-rich ecosystem for solving differential equations in Julia. *Journal of Open Research Software*, 5(1). DOI: 10.5334/jors.151

10 Acknowledgments

Research conducted in collaboration with Dr. Romesh Batra, Department of Mechanical Engineering, Virginia Tech.