Respect as a Hard Constraint in Multi-Objective Optimization: A Penalty-Based Safety-First Core for Reproducible Toy Benchmarks

Christof Krieg
maat-research.com | github.com/Chris4081

January 27, 2026

Abstract

We study *Respect* as a safety-first constraint in multi-objective optimization, implemented as a hard feasibility condition enforced by a large penalty term. We present a minimal optimization core that combines (i) weighted field objectives (e.g., Harmony), (ii) Occam-style complexity regularization, and (iii) *Respect* constraints expressed as inequality margins $g(x) \geq 0$. We demonstrate that the resulting energy landscape makes unsafe regions mathematically unstable: optimizers converge to solutions that satisfy constraints even when such solutions are not globally optimal under the base objective alone. We provide two toy benchmarks—an Occam tie-breaker and a Respect boundary corridor—to illustrate behavior, explainability, and reproducibility using standard solvers (L-BFGS-B and dual annealing).

1 Introduction

Many real-world optimization problems require not only high performance but also strong safety guarantees. In practice, safety is often layered on top of optimization (filters, post-checks, heuristics), which can fail under distribution shift or adversarial settings. This paper explores a simple but useful idea: treat safety (*Respect*) as a *hard* constraint within the optimization objective itself, such that constraint violations become energetically prohibitive.

Our contribution is intentionally minimal: a small, reproducible core and two toy experiments. The goal is not to claim a new solver, but to provide a crisp pattern for "safety-first" optimization and an explainability interface via constraint margins.

2 Problem Setup

We consider decision variables $x \in \mathcal{X}$ with bounds $x \in [a, b]$. Let $f_i(x)$ be objective "fields" (e.g., Harmony, Balance), combined as a weighted sum. Let $\Omega(x)$ be a complexity measure (Occam regularizer). Let constraints be written as margin functions $g_j(x) \geq 0$.

2.1 Energy Objective with Respect Penalty

We optimize an energy of the form:

$$E(x) \;=\; \sum_{i=1}^{m} w_i f_i(x) \;+\; \lambda_O\, \Omega(x) \;+\; \lambda_R \sum_{j=1}^{k} \alpha_j \big(\max(0, -g_j(x))\big)^2. \tag{1}$$

Here, λ_R is chosen large so that infeasible solutions are heavily penalized. The squared hinge penalty yields smooth behavior around feasibility boundaries and strongly discourages large violations.

2.2 Interpretation

- **Fields:** encode multiple objectives as scalar functions on a state derived from x.
- **Occam (optional):** resolves ties by preferring lower complexity when objective quality is similar.
- **Respect:** a feasibility layer that does not require additional post-processing; unsafe regions become unstable minima.

3 Optimization Procedure

We use standard optimizers on $E(x)$:

- **Local optimization:** L-BFGS-B with box bounds.
- **Global exploration (optional):** dual annealing for multimodal landscapes.

We treat "Creativity" as exploration intensity (temperature) rather than a direct objective term. This preserves scientific clarity: exploration affects search, not the definition of a good solution.

4 Implementation

4.1 Core Data Structures

We provide an implementation based on Equation (1). In our minimal library:

- a `Field` is a weighted scalar function of a state,
- a `Constraint` is a margin function with `margin>=0` indicating feasibility,
- `MaatCore.integrate` computes $E(x)$, and
- `MaatCore.seek` runs an optimizer over x.

4.2 Reference Python Snippet

Listing 1: Minimal safety-first optimization core (excerpt).

```
from dataclasses import dataclass
from typing import Callable, Any, Iterable, List, Sequence, Optional
import numpy as np
from scipy.optimize import minimize, dual_annealing

StateFn = Callable[[float], Any]
FieldFn = Callable[[Any], float]
ConstraintFn = Callable[[Any], float]

@dataclass(frozen=True)
```

```
class Constraint:
    name: str
    func: ConstraintFn
    weight: float = 1.0  # alpha_j

@dataclass(frozen=True)
class Field:
    name: str
    func: FieldFn
    weight: float = 1.0  # w_i
    def value(self, state: Any) -> float:
        return float(self.func(state)) * float(self.weight)

class MaatCore:
    def __init__(self,
                 fields: Iterable[Field],
                 constraints: Optional[Iterable[Constraint]] = None,
                 occam_lambda: float = 0.0,
                 safety_lambda: float = 1e6):
        self.fields = list(fields)
        self.constraints = list(constraints or [])
        self.occam_lambda = float(occam_lambda)
        self.safety_lambda = float(safety_lambda)

    def integrate(self, state: Any) -> float:
        total = sum(f.value(state) for f in self.fields)
        complexity = float(getattr(state, "complexity", 0.0))
        occam = self.occam_lambda * complexity

        penalty = 0.0
        for c in self.constraints:
            margin = float(c.func(state))
            if margin < 0:
                violation = -margin
                penalty += float(c.weight) * (violation ** 2)

        return total + occam + self.safety_lambda * penalty

    def seek(self, state_fn: StateFn, x0: Sequence[float], *,
             S: float = 0.0, use_annealing: bool = False,
             bounds=((0.0, 1.0),), maxiter: int = 1000, seed: int | None =
                 None):
        def objective(x_arr):
            x = float(np.atleast_1d(x_arr)[0])
            return self.integrate(state_fn(x))

        if use_annealing:
            if seed is not None:
                np.random.seed(int(seed))
            return dual_annealing(objective, bounds=list(bounds),
                                  initial_temp=10.0 * (1.0 + float(S)),
                                  maxiter=int(maxiter))

        return minimize(objective, x0=np.atleast_1d(x0).astype(float),
```

```
                    bounds=list(bounds), method="L-BFGS-B",
                    options={"maxiter": int(maxiter)})
```

5 Toy Benchmarks

5.1 Occam Tie-Breaker

We construct a landscape where two solutions have equal Harmony but different complexity. With $\lambda_O > 0$, the optimizer should select the lower-complexity basin.

Expected behavior: among equally good solutions, the chosen $x^\star$ minimizes $\Omega(x)$.

5.2 Respect Boundary Corridor

We define a "safe corridor" by enforcing $g(x) \geq 0$. Even if the unconstrained objective prefers an unsafe region, the penalty term should push the solution into feasibility.

Expected behavior: the final solution satisfies the constraint margin and is accompanied by a diagnostic report of margins.

6 Results

We report:

- $x^\star$ (best decision),
- objective breakdown (field contribution + Occam + penalty),
- constraint margins (OK / VIOLATION),
- success status from the optimizer.

6.1 Example Output (Qualitative)

In typical runs:

- Occam tie-breaker selects a simpler $x^\star$ when Harmony is equal.
- Respect corridor yields a boundary-adjacent $x^\star$ that remains feasible, with small positive margin.

7 Discussion

This penalty-based approach is intentionally simple and compatible with standard solvers. It provides a practical "safety-first" default and an explainability surface (margins) that developers can log or visualize.

7.1 Limitations

- Penalty methods can require careful tuning of λ_R for numerical stability.
- Hard feasibility is not formally guaranteed for finite λ_R; however, it becomes practically robust as λ_R grows.
- Toy benchmarks do not substitute for domain-specific constraints and real safety validation.

7.2 Future Work

- Replace penalties with exact constrained solvers where possible (e.g., interior-point methods).
- Extend to higher-dimensional $x \in \mathbb{R}^n$ and nontrivial constraint sets.
- Add automated plots of constraint boundaries and energy landscapes for debugging and pedagogy.

8 Reproducibility

We recommend:

- Pin solver versions and random seeds.
- Provide a one-command run for each benchmark.
- Log field breakdown and constraint margins for every run.

Acknowledgments

Thanks to collaborators and community feedback that helped sharpen the safety-first framing and the minimal reproducible examples.