BPS5231

Artificial Intelligence

for

Sustainable

Building

Design

L4

Ang Yu Qian

Assistant Professor

L04.1

Design Space Exploration

DSE Basics

Random Search

Grid Search

L04.2

Optimization

Basics

Paper Plane
Design of Experiment

Genetic Algorithm

L04.3

Beyond Form

Rhino | Grasshopper

Galapagos

Nano Banana

L04.1

Design Space Exploration

DSE Basics

Random Search

Grid Search

L04.2

Optimization

Basics

Paper Plane
Design of Experiment

Genetic Algorithm

L04.3

Beyond Form

Rhino | Grasshopper

Galapagos

Nano Banana

"*if you want to have good ideas, you must have lots of ideas and learn to throw away the bad ones*"

\- Linus Pauling -

Design space exploration (DSE) is the systematic process of analyzing and evaluating a wide range of possible design alternatives to identify solutions that best meet specific requirements, such as performance, cost, and other parameters

DSE involves generating and examining different design options and systematically narrowing these down based on set criteria or optimization goals.

Why is Design Space Exploration important?

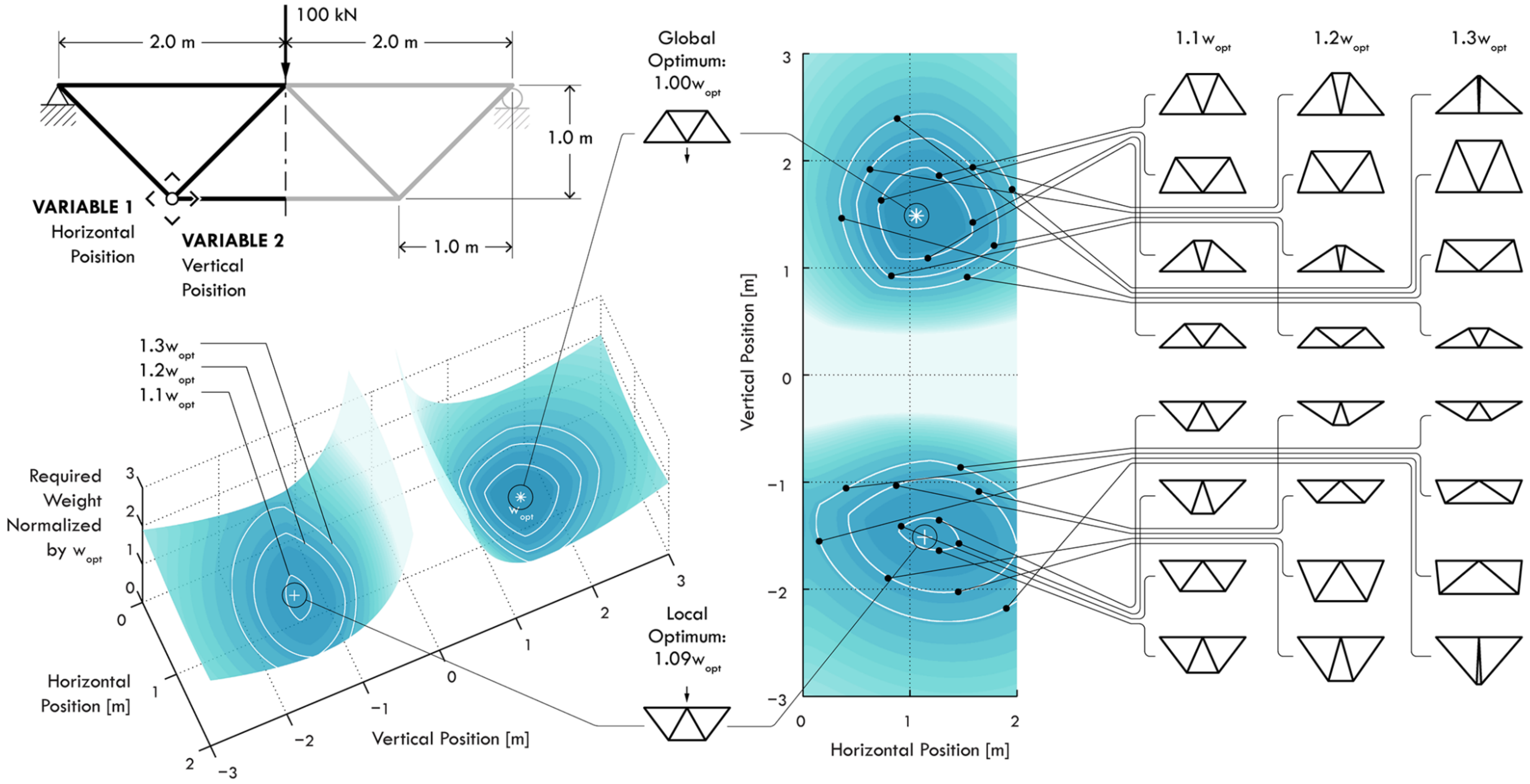
Early Stage Optimization: allows for optimization and adjustments before implementation, reducing costly changes later in the process.

Handling Complexity: helps manage the complexity brought by numerous design choices and requirements, making it critical for large systems with millions or billions of possible configurations.

Tradeoff Analysis: By considering multiple objectives, such as performance vs. power consumption, DSE supports multi-objective decision-making and innovation

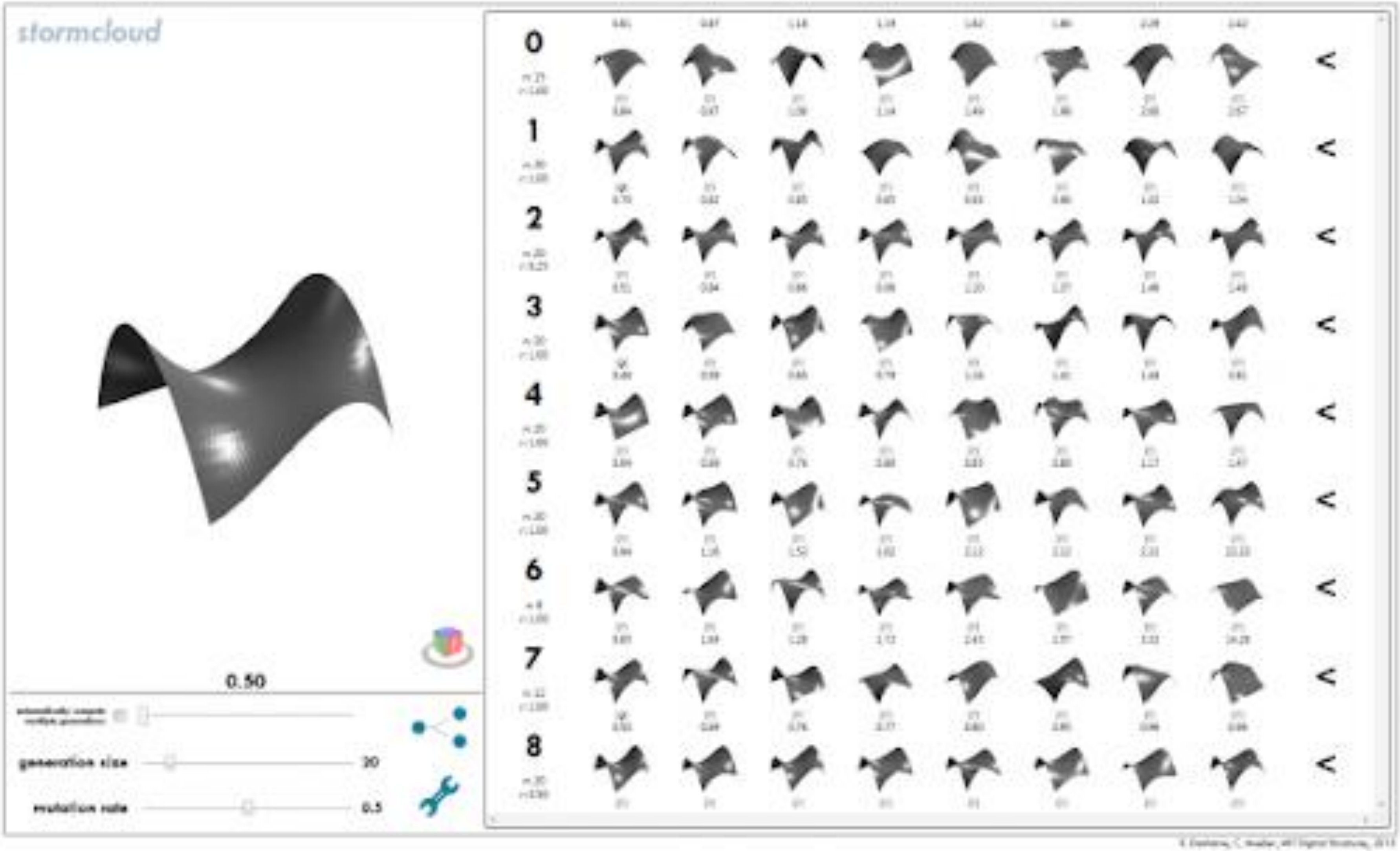
Applicable in many domains, especially sustainable building design

Design Space Exploration



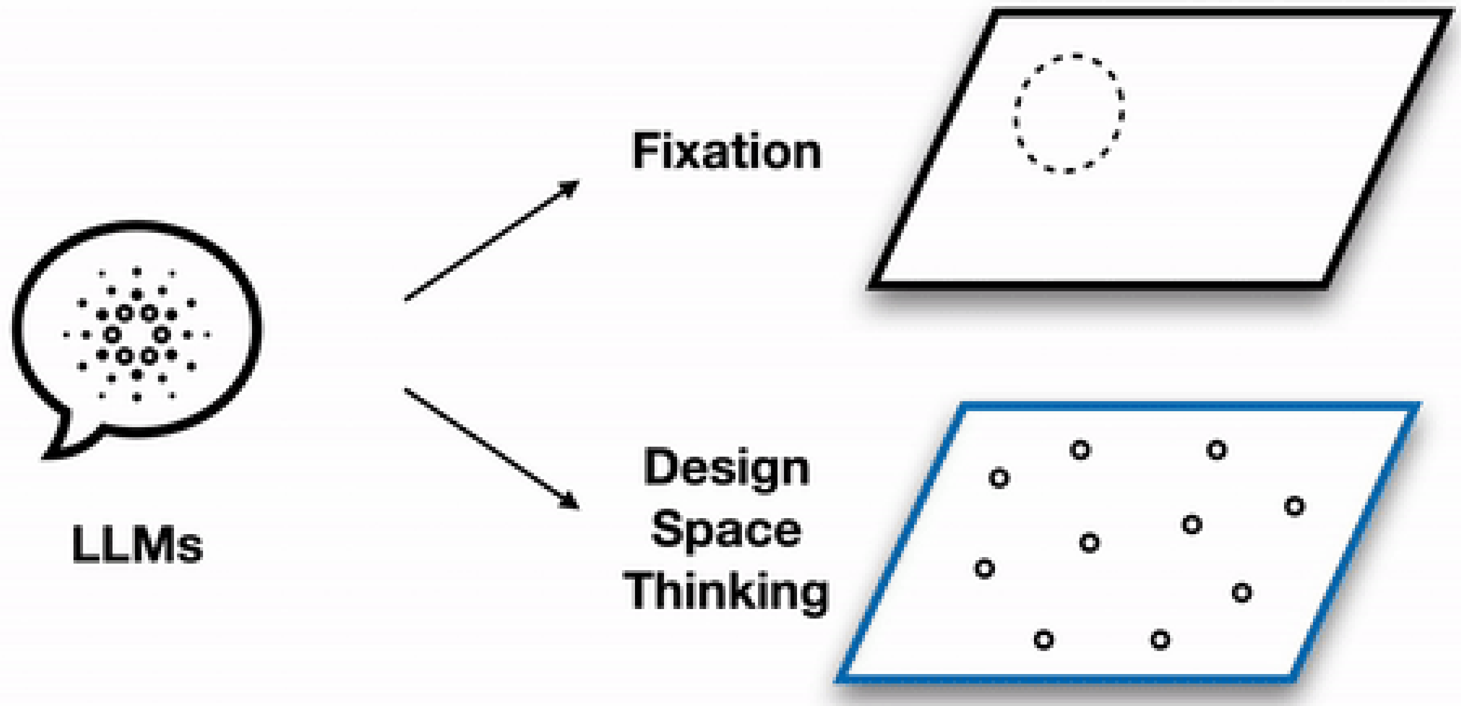
Source: Professor Caitlin Mueller, MIT

Design Space Exploration



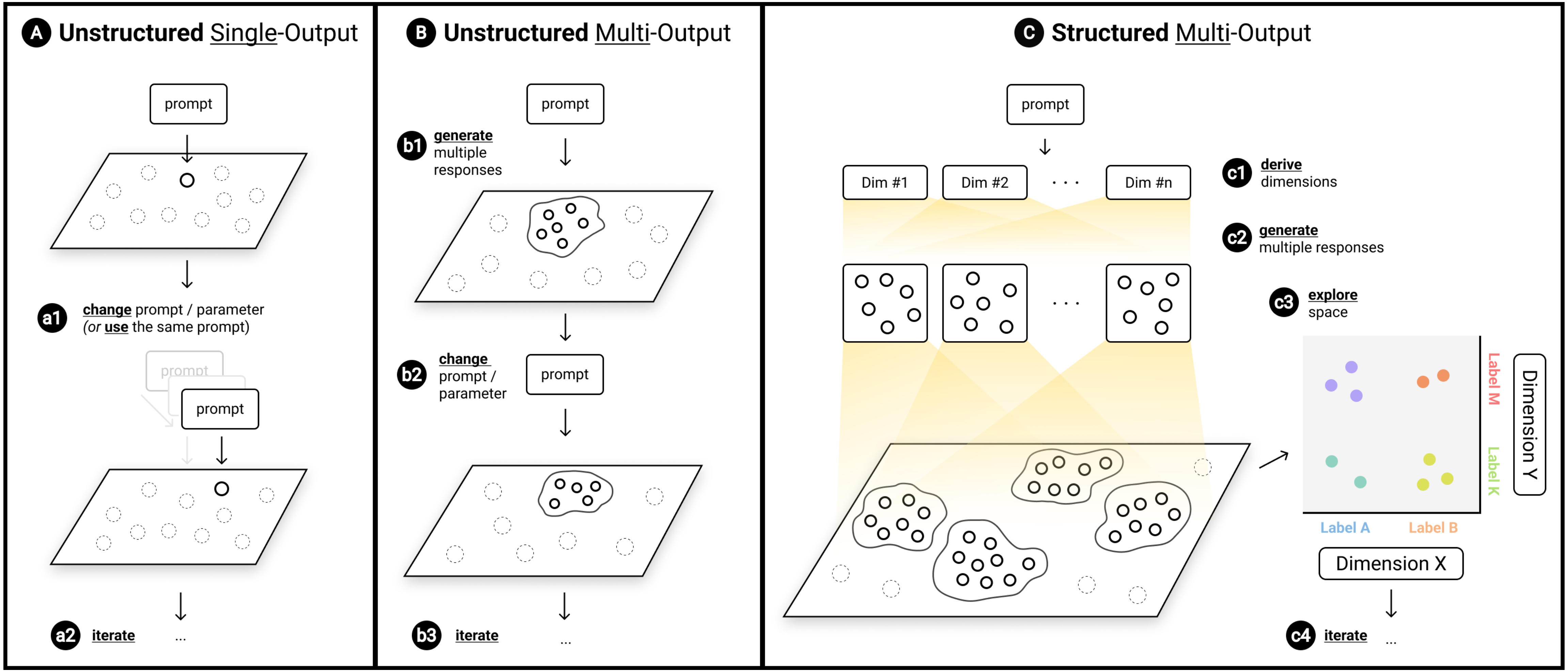
Source: Professor Caitlin Mueller, MIT

Design Space Exploration



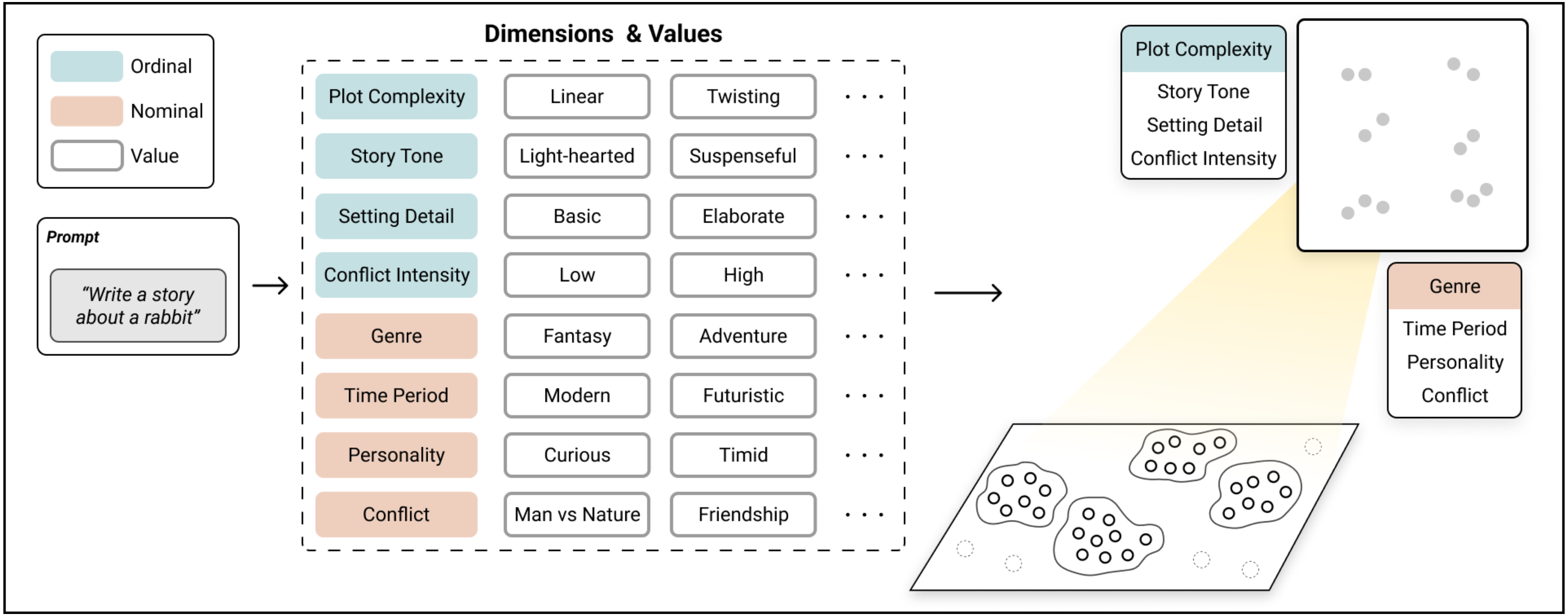
Source: Suh, et al. (2024). Luminate: Structured Generation and Exploration of Design Space with Large Language Models for Human-AI Co-Creation

Design Space Exploration



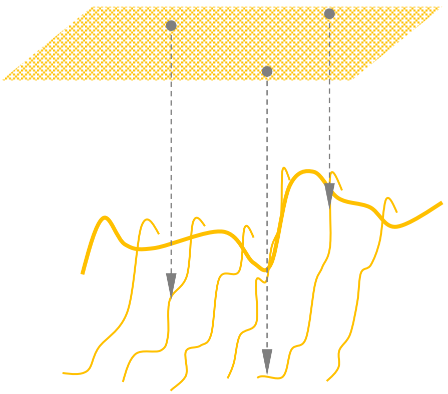
Source: Suh, et al. (2024). Luminate: Structured Generation and Exploration of Design Space with Large Language Models for Human-AI Co-Creation

Design Space Exploration

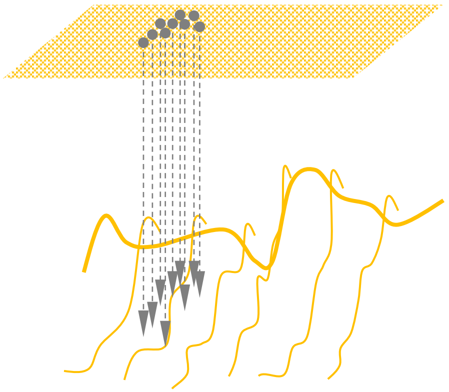


Source: Suh, et al. (2024). Luminate: Structured Generation and Exploration of Design Space with Large Language Models for Human-AI Co-Creation

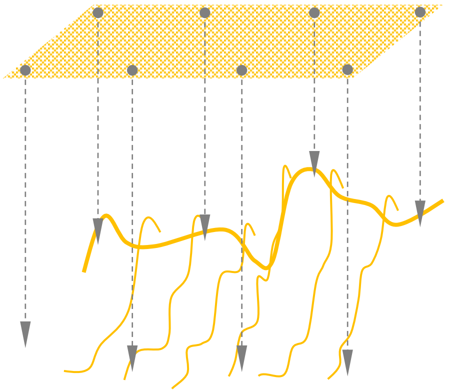
Design Space Exploration



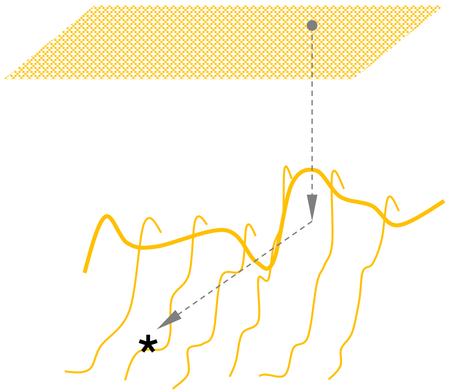
guess-and-check



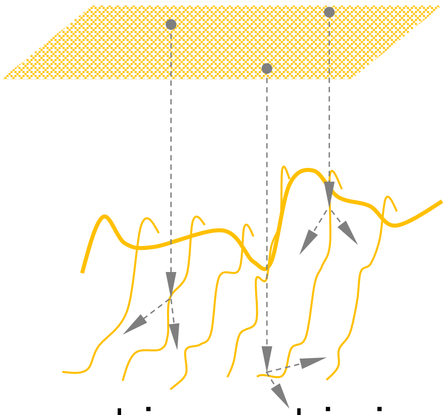
real-time feedback



grid search



optimization



interactive optimization

Source: Professor Caitlin Mueller, MIT

Google Colab
bit.ly/BPS5231-L4

Here are 5 verified passes (rounded values):

WWR 0.183, Shading 0.021 m, Insulation 0.872, Orientation 94.3°
→ EUI 118.25, Daylight 55.1, TDH 14.6 h/yr, Cost 99.1

WWR 0.190, Shading 0.046 m, Insulation 0.845, Orientation 98.0°
→ EUI 119.69, Daylight 55.0, TDH 16.1 h/yr, Cost 99.6

WWR 0.183, Shading 0.0048 m, Insulation 0.899, Orientation 84.0°
→ EUI 117.21, Daylight 55.8, TDH 14.5 h/yr, Cost 100.0

WWR 0.183, Shading 0.0046 m, Insulation 0.887, Orientation 76.4°
→ EUI 117.74, Daylight 55.5, TDH 15.9 h/yr, Cost 99.0

WWR 0.197, Shading 0.0018 m, Insulation 0.877, Orientation 66.8°
→ EUI 119.21, Daylight 56.7, TDH 18.6 h/yr, Cost 99.2

Source: Professor Caitlin Mueller, MIT

Possible ways to steer your sliders to a pass

Keep WWR low ~ 0.18–0.20 to control cost and TDH while still scraping into daylight ≥55.

Insulation high ~ 0.85–0.90 to drag EUI below 120 (this eats cost, so compensate elsewhere).

Shading almost zero (≈ 0 to 0.05 m) — shading helps EUI/TDH, but it kills daylight and adds cost in this toy.

Orientation around 70–100° (roughly E/W-ish) works with the daylight and TDH balance in the surrogate.

Source: Professor Caitlin Mueller, MIT

If you'd like the feasible region to be less difficult / knife-edge, you can tweak any of these (in the surrogate):

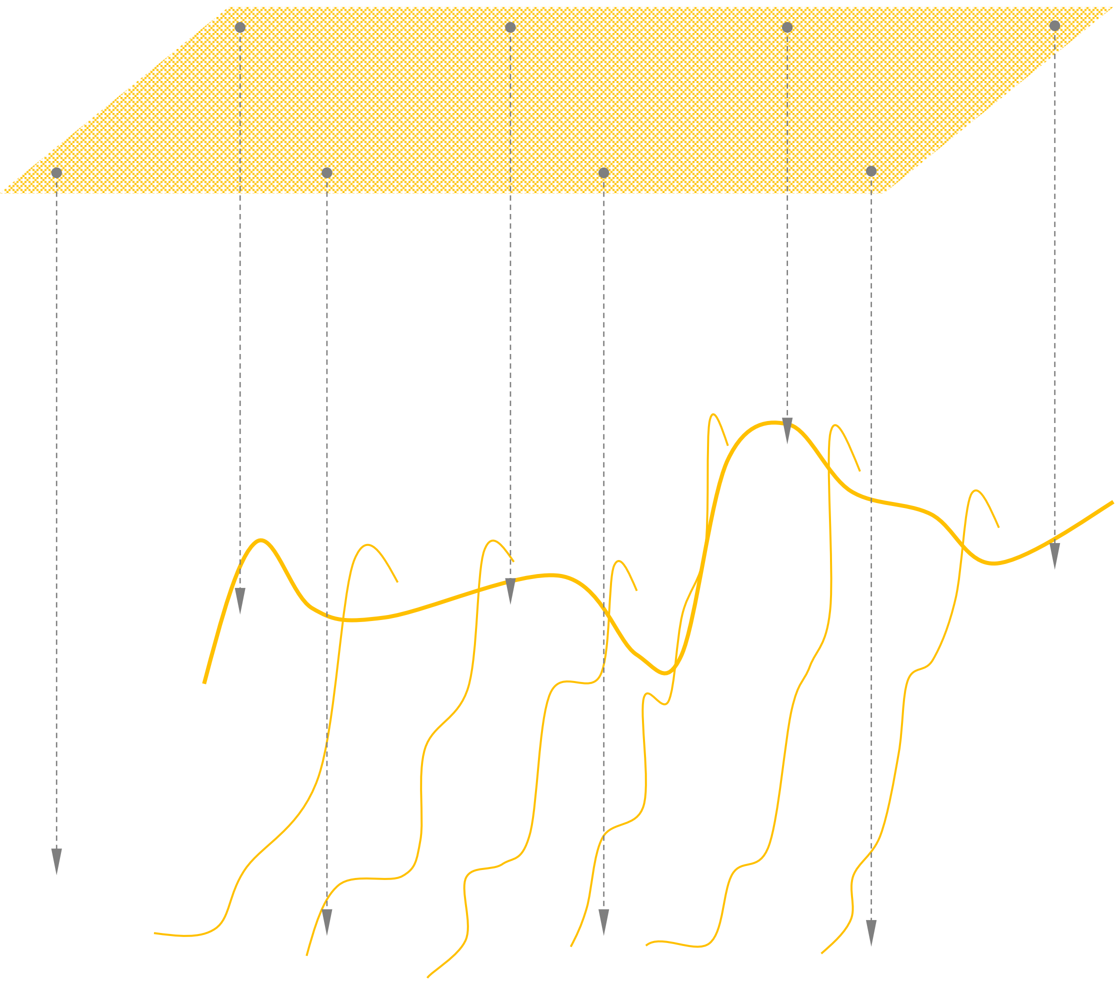
Lower EUI_base slightly (e.g., 145 → 150 currently)

Reduce the cost weight on insulation or shading (e.g., 75.0 insulation → 65.0 insulation)

Ease the daylight lower bound (e.g., 55 → 52)

Source: Professor Caitlin Mueller, MIT

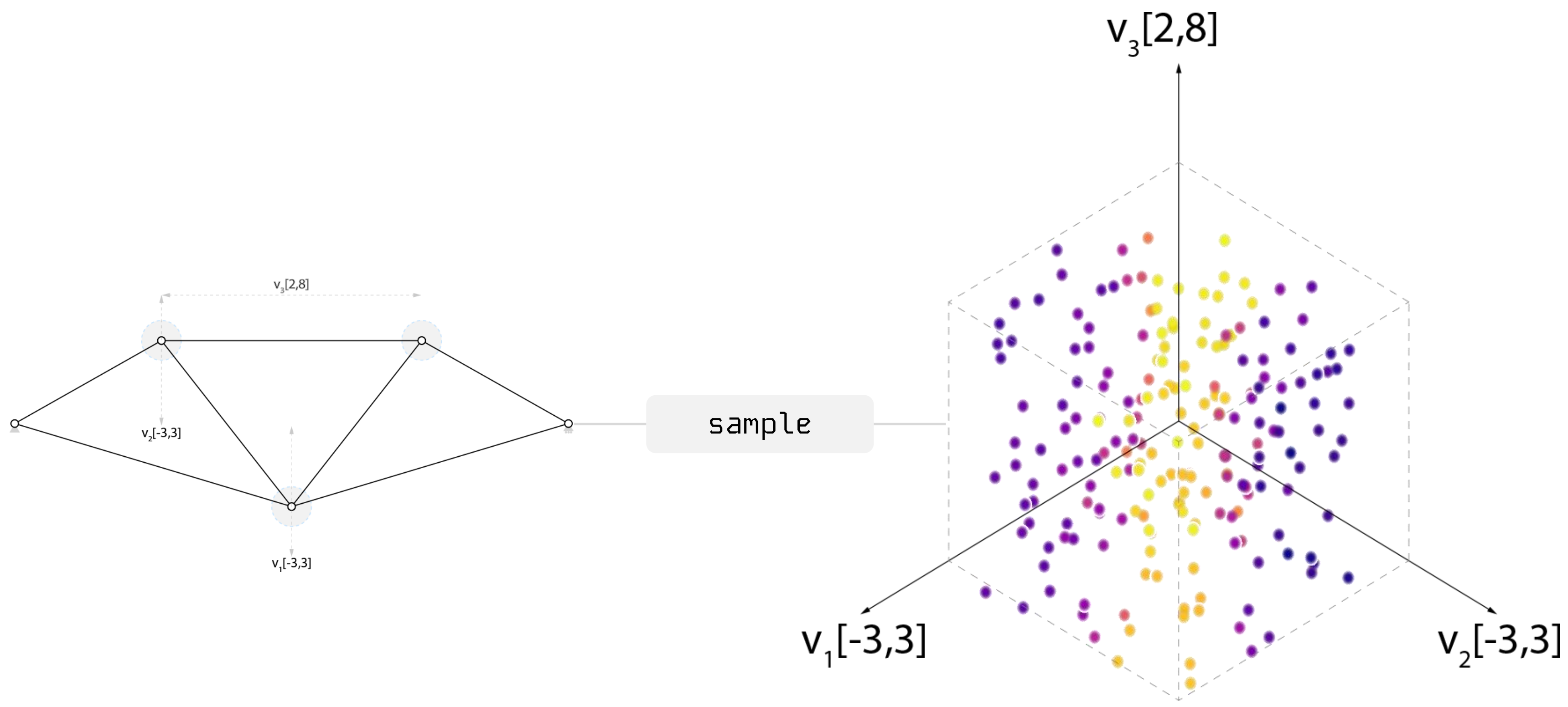
Design Space Exploration (Grid Search)

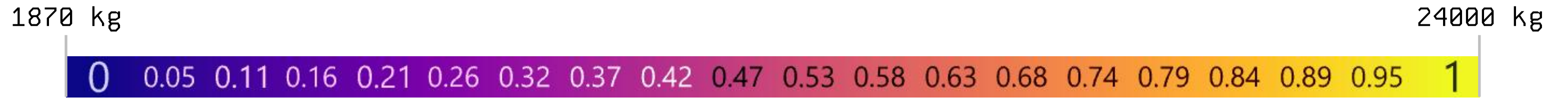


grid search / sampling

Source: *Professor Caitlin Mueller, MIT*

Design Space Exploration (Grid Search)

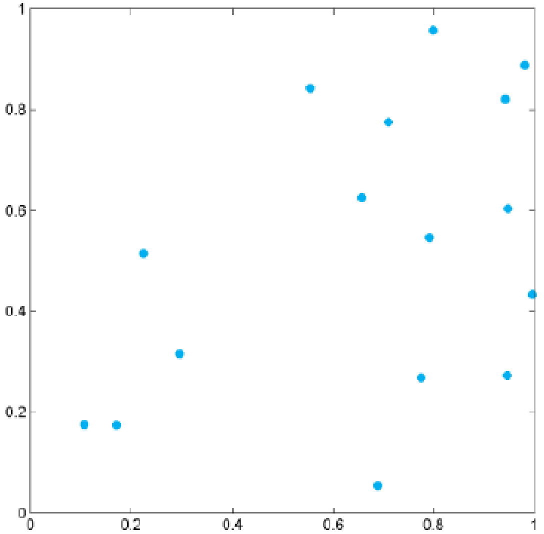




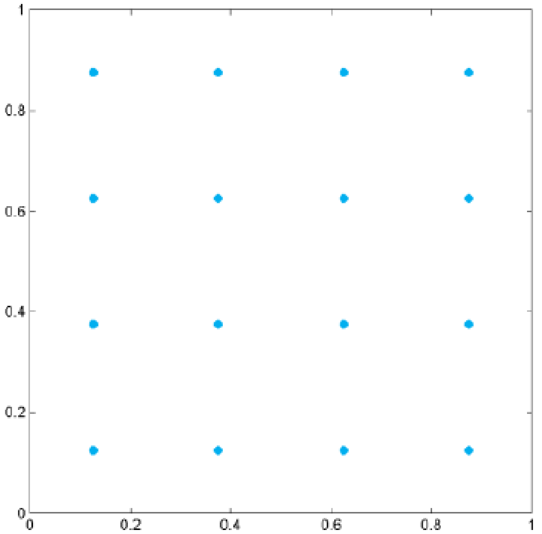
Source: Professor Caitlin Mueller, MIT

Design Space Exploration (Grid Search)

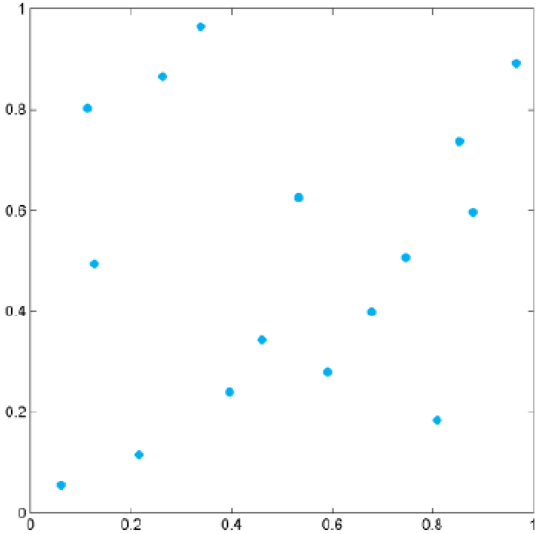
Random / Uniform



Grid



Latin Hypercube



Source: Professor Caitlin Mueller, MIT

Google Colab
bit.ly/BPS5231-L4

L04.1
Design Space Exploration

DSE Basics

Random Search

Grid Search

L04.2
Optimization

Basics

Paper Plane
Design of Experiment

Genetic Algorithm

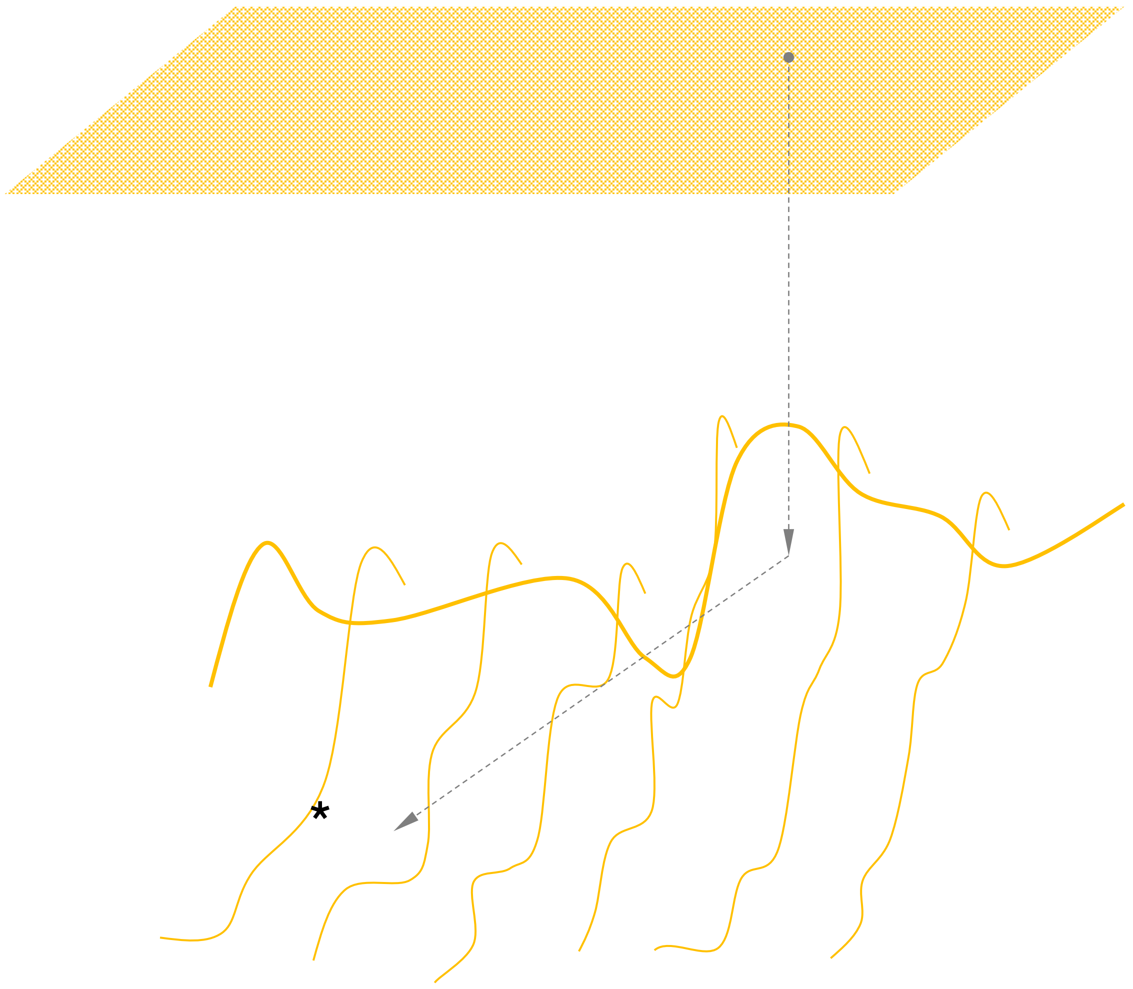
L04.3
Beyond Form

Rhino | Grasshopper

Galapagos

Nano Banana

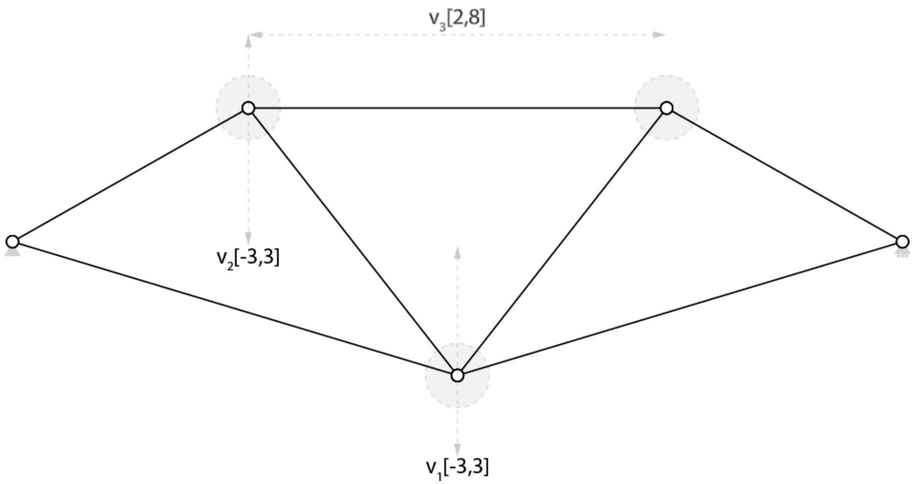
Design Space Exploration (Optimization)



optimization

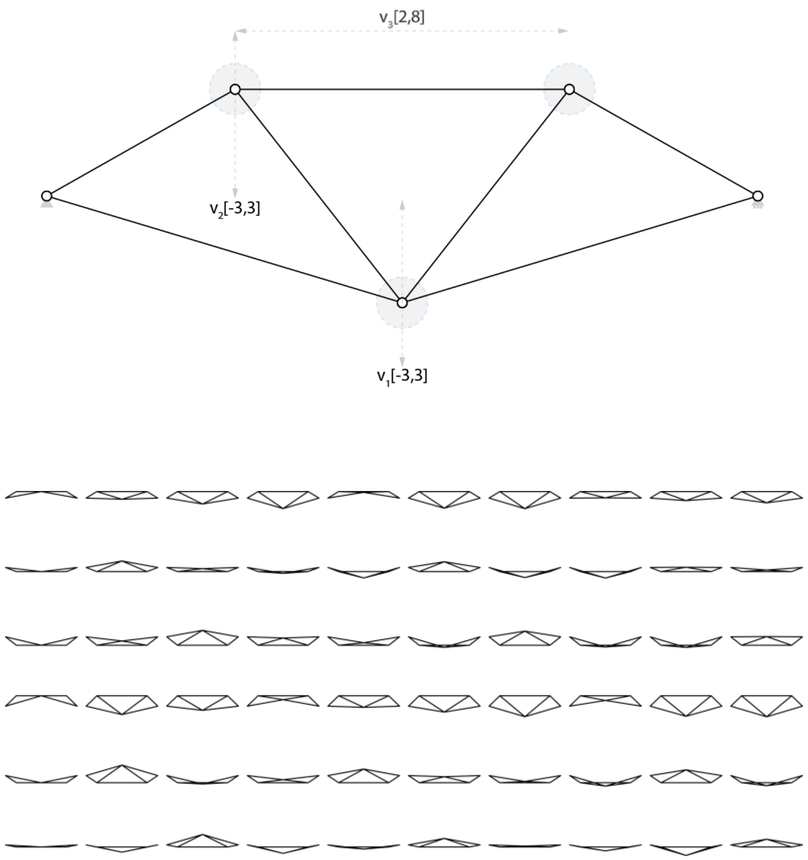
Source: Professor Caitlin Mueller, MIT

Design Space Exploration (Problem Statement)

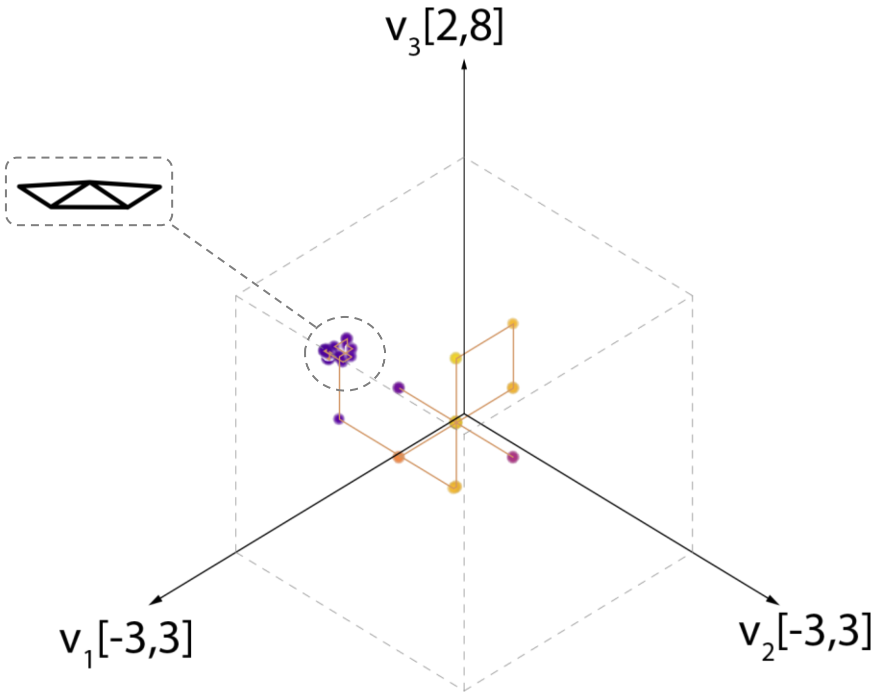


Scottish Parliament
Miralles, Edinburgh, 2014

Source: Professor Caitlin Mueller, MIT



3D Design Space [3 variables, 1 objective]



Weight Optimization

Source: Professor Caitlin Mueller, MIT

Types of Design Variables

- Continuous:
  e.g., Window-to-wall ratio, shading depth, insulation thickness
- Discrete / Integer:
  e.g., Number of floors, orientation (in fixed increments)
- Binary (0/1):
  e.g., “Green roof installed or not”
- Boolean (True/False):
  e.g., “Use natural ventilation?”

Constraints

Comfort thresholds
Regulatory codes
(e.g., max glazing ratio)
Budget / material limits

What are we optimizing?

- Energy Use Intensity (EUI)
- Daylight availability
- Thermal comfort (TDH)
- Cost / Embodied carbon

Minimize $f_1(x)$ = Energy Use Intensity (EUI)
Maximize $f_3(x)$ = Daylight availability
Minimize $f_2(x)$ = Cost / Embodied Carbon
Minimize $f\ (x)$ = Thermal discomfort hours

Optimization Workflow

1. Define design variables (controlled by designers)

2. Set objectives & constraints (targets: energy, daylight, comfort, cost)

3. Search the design space using algorithms
    - Random / grid search
    - Gradient-based optimization
    - Evolutionary / genetic algorithms
    - Multi-objective optimization (Pareto front)

4. Select optimal solutions that balance performance + sustainability goals

objective function

Minimize the building's energy use intensity (EUI) by tuning design variables such as window-to-wall ratio and shading, while ensuring daylight availability and occupant comfort requirements within budget.

constraints

design variables

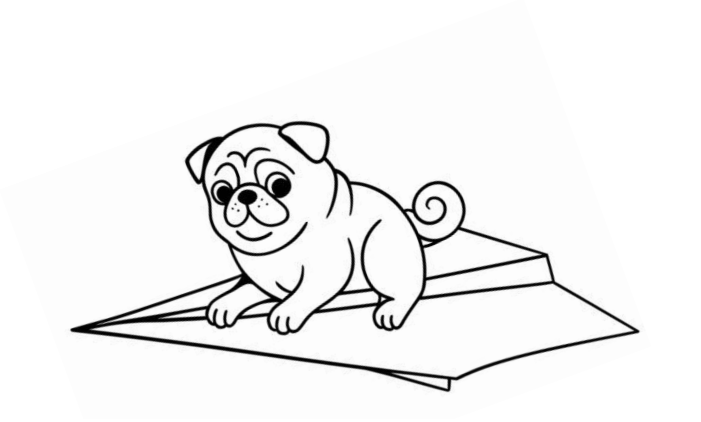
Design Experiment

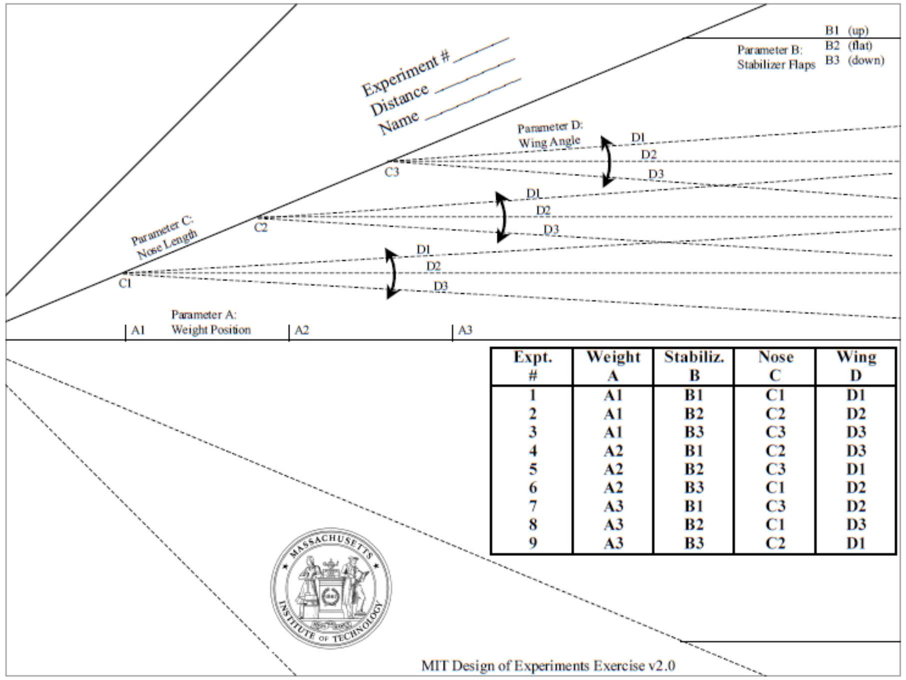
Objective: Maximize Airplane Glide Distance

Design variables:
1. Weight distribution
2. Stabilizer orientation
3. Nose length
4. Wing angle

Three levels for each design variable

Full factorial design: $3^4 = 81$ experiments





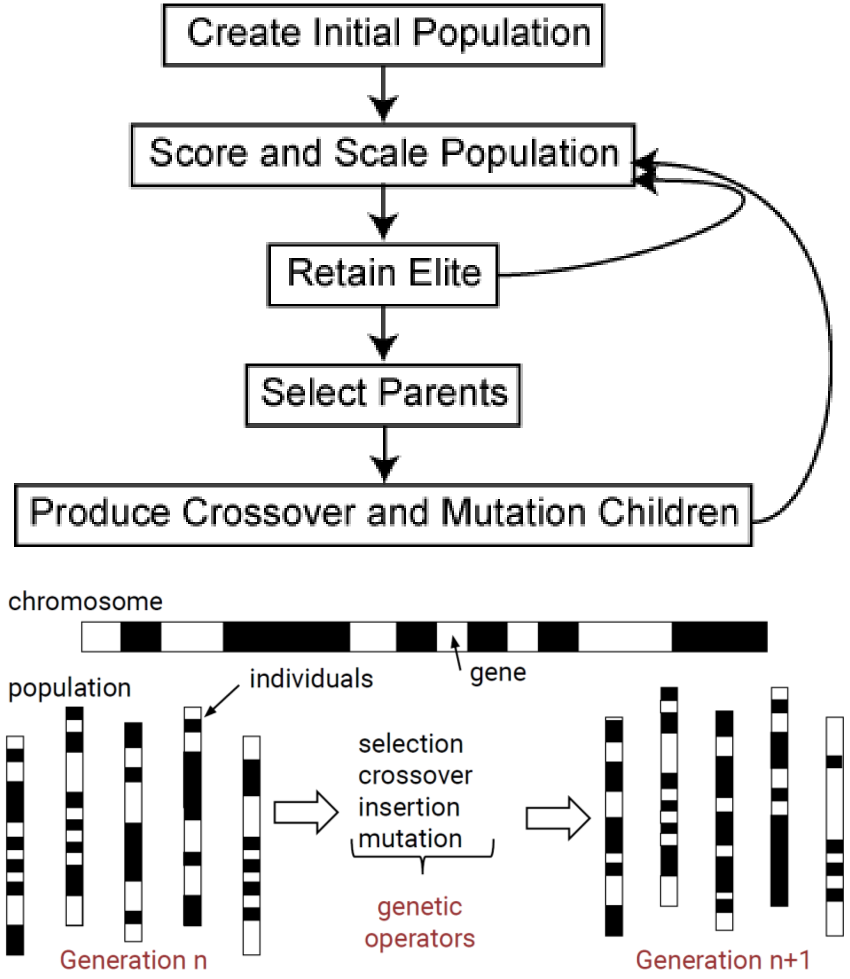
Expt. #	Weight A	Stabiliz. B	Nose C	Wing D
1	A1	B1	C1	D1
2	A1	B2	C2	D2
3	A1	B3	C3	D3
4	A2	B1	C2	D3
5	A2	B2	C3	D1
6	A2	B3	C1	D2
7	A3	B1	C3	D2
8	A3	B2	C1	D3
9	A3	B3	C2	D1

Record your scores:
(choose 3 variable set/experiments)
bit.ly/BPS5231-class

Optimization Approaches

1. Gradient-based optimization (gradient descent)
2. Evolutionary / genetic algorithms
3. Multi-objective optimization

Genetic Algorithm



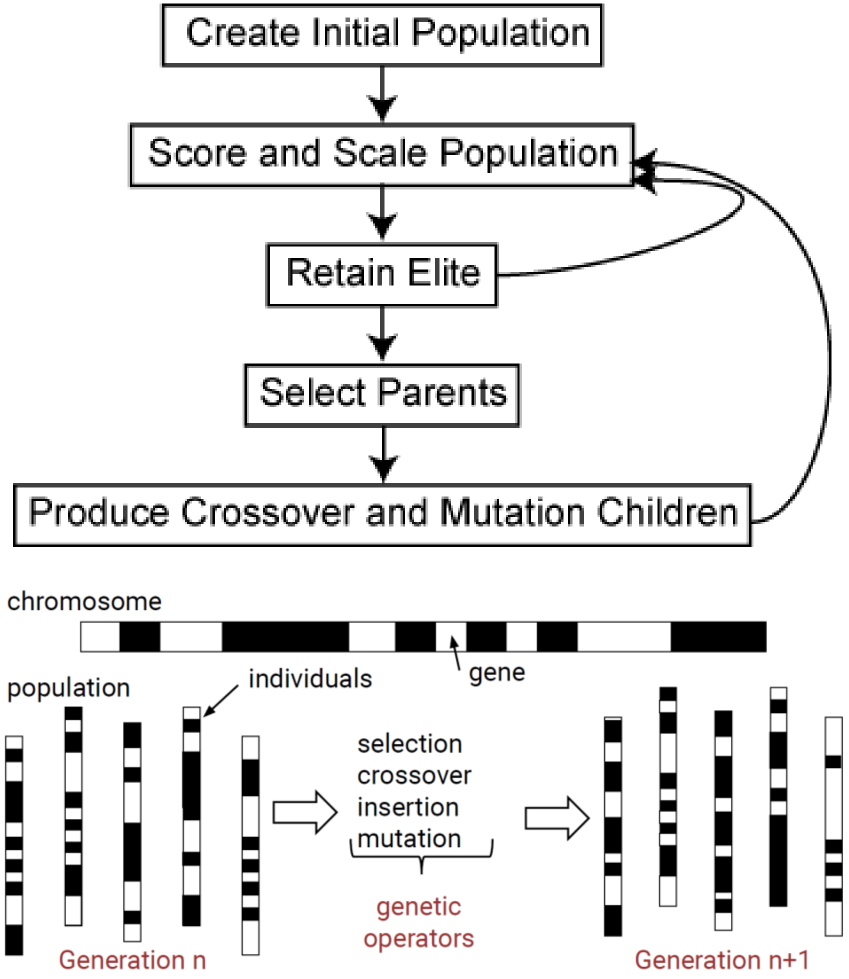
The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution.

The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals from the current population to be parents and uses them to produce the children for the next generation.

Over successive generations, the population "evolves" toward an optimal solution. You can apply the genetic algorithm to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, nondifferentiable, stochastic, or highly nonlinear.

Source: Matlab

Genetic Algorithm



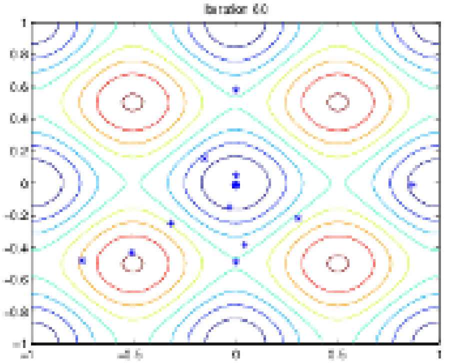
Algorithm flow:

1. Initialize a population of random design solutions
   (e.g., different combinations of WWR, shading, insulation, orientation)
2. Evaluate fitness of each design
   (objectives: energy, daylight, comfort, cost)
3. Select the fittest designs to "reproduce"
4. Crossover – combine parts of two designs to form new solutions
5. Mutation – randomly tweak a variable to introduce diversity
6. Repeat until stopping criteria met (e.g., convergence or generations)

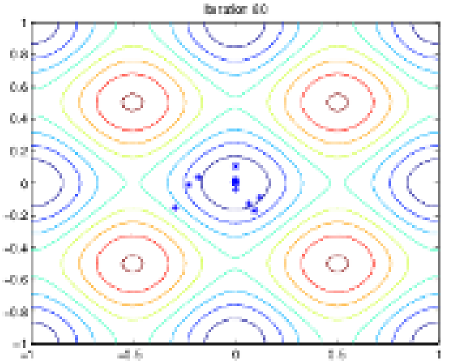
Source: Matlab

Genetic Algorithm

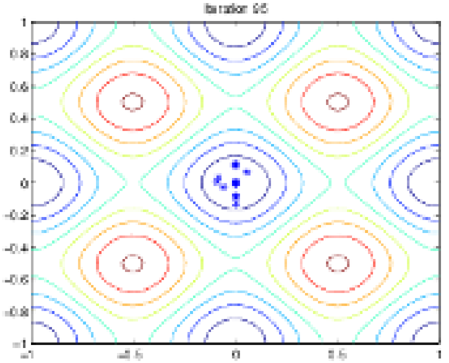
iteration 60



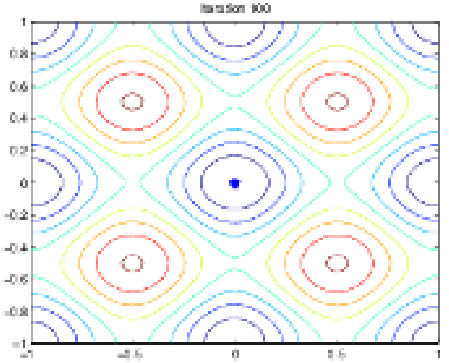
iteration 80



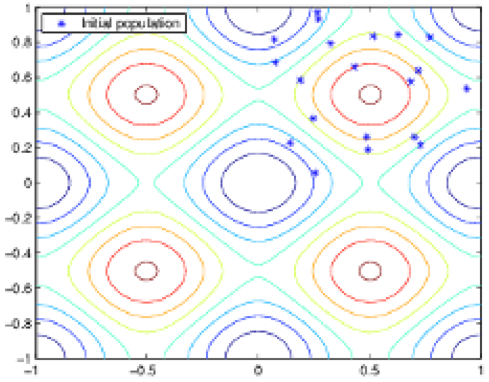
iteration 95



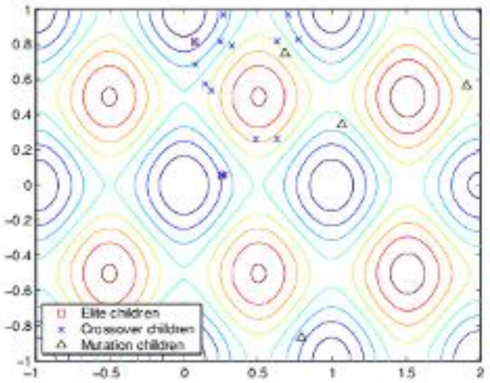
iteration 100



initial population



mutation / crossover children



The algorithm begins by creating a random initial population

The algorithm creates crossover children by combining pairs of parents in the current population, and creates mutation children by randomly changing the genes of individual parents

Further reading: https://www.mathworks.com/help/gads/how-the-genetic-algorithm-works.html

Source: Matlab

Outline of Genetic Algorithm

1. The algorithm begins by creating a random initial population.
2. The algorithm then creates a sequence of new populations. At each step, the algorithm uses the individuals in the current generation to create the next population. To create the new population, the algorithm performs the following steps:
3. Scores each member of the current population by computing its fitness value. These values are called the raw fitness scores.
4. Scales the raw fitness scores to convert them into a more usable range of values. These scaled values are called expectation values.
5. Selects members, called parents, based on their expectation.
6. Some of the individuals in the current population that have lower fitness are chosen as elite. These elite individuals are passed to the next population.
7. Produces children from the parents. Children are produced either by making random changes to a single parent—mutation—or by combining the vector entries of a pair of parents—crossover.
8. Replaces the current population with the children to form the next generation.
9. The algorithm stops when one of the stopping criteria is met. See Stopping Conditions for the Algorithm.
10. The algorithm takes modified steps for linear and integer constraints. See Integer and Linear Constraints.
11. The algorithm is further modified for nonlinear constraints. See Nonlinear Constraint Solver Algorithms for Genetic Algorithm.

Source: Matlab

Read up on other optimization techniques
(e.g. particle swarm, simulated annealing)

Multi-objective Optimization

Real-world design problems rarely have just one goal

Example in building design:

- Minimize Energy Use Intensity (EUI)
- Minimize Cost / Carbon
- Maximize Daylight availability
- Minimize Thermal discomfort

Trade-offs exist → improving one may worsen another

Instead of one “best” solution → we find a set of optimal compromises

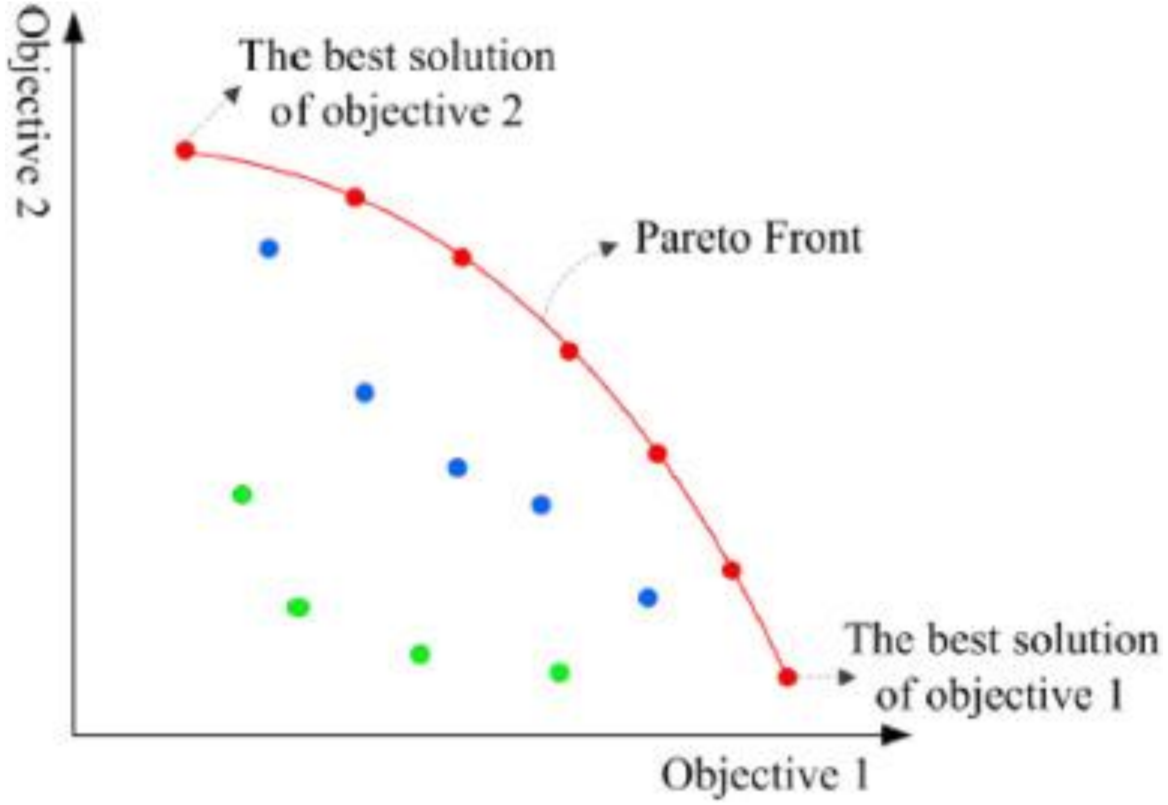
Source: Matlab

Pareto Optimality

Pareto Optimal Solution
No other solution is better in all objectives simultaneously

Pareto Front
The set of non-dominated solutions representing the trade-off curve/surface

Helps designers see the spectrum of choices and make informed decisions



Source: Stir Welding and Processing, 2014

Pareto Optimality

Each point = a design option
(a unique combination of WWR, shading, insulation, etc.).

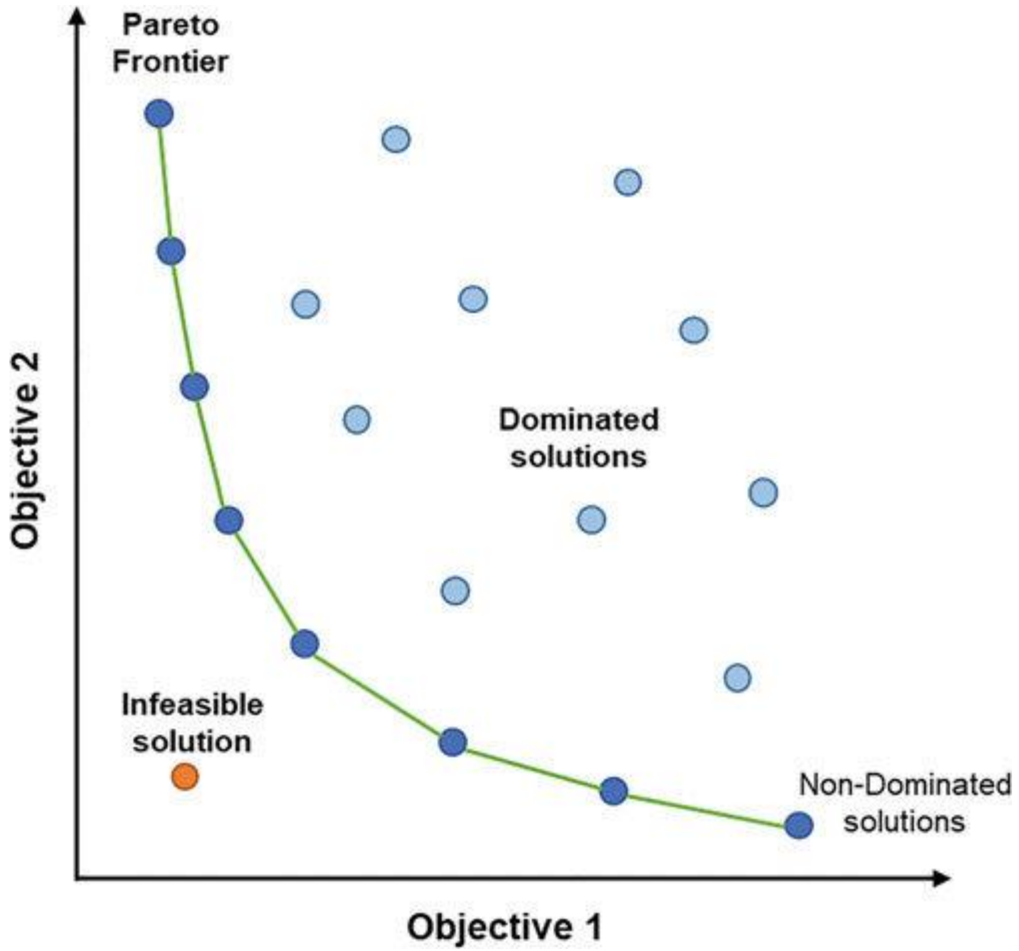
The Pareto front = the set of non-dominated designs:

- You cannot improve one objective (e.g., lower EUI) without worsening another (e.g., higher cost or less daylight).

The front shows the trade-off boundary of what is achievable.

You can see how improving one objective sacrifices another.

- Example: Lowering EUI usually increases cost



Source: Xuhan Liu

Eastport, Maine

NREL

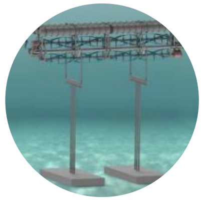
What renewable energy systems mix is feasible for the town?

Optimization (Multi-Objective Optimization Example)

Solar

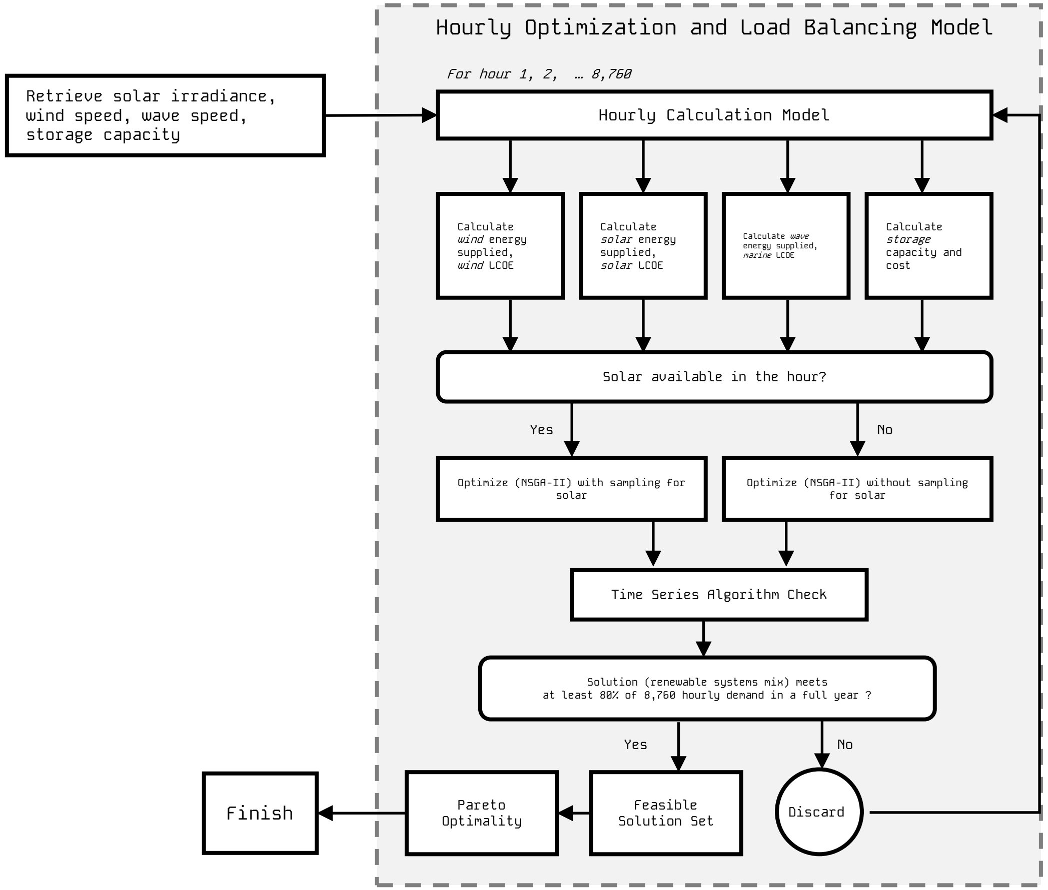
Wind

Tidar

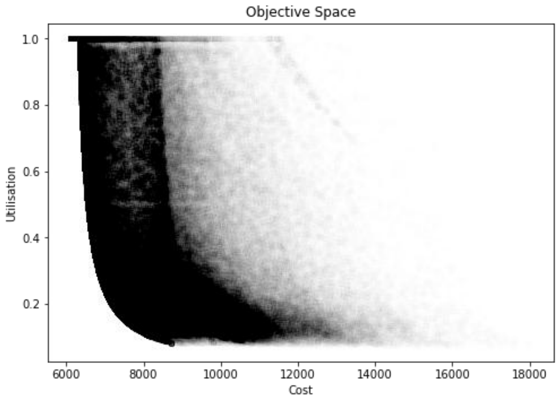


Battery

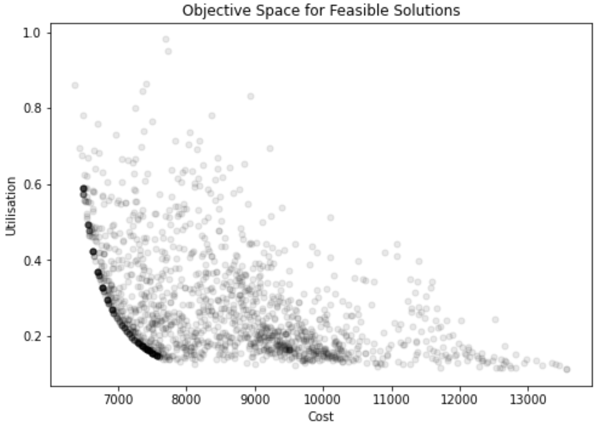
Objectives
1. Minimize Total Renewable Energy Cost (LCOE)
2. Maximize Utilization (demand/power capacity)
3. Minimize Variation (abs(demand – energy supplied))

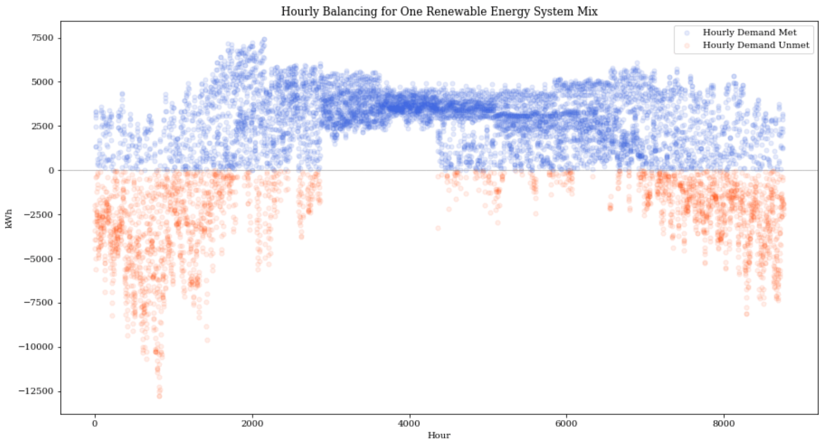


Full hourly optimization



Filtered feasibility optimization

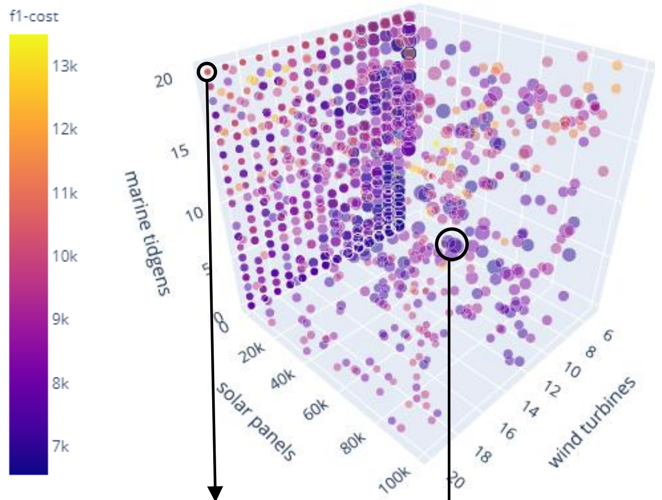




- Time series algorithm check
- Only accept solution set if it satisfies at least 80% of hourly demand (7,008 out of 8,760 hours)

Optimization (Multi-Objective Optimization Example)

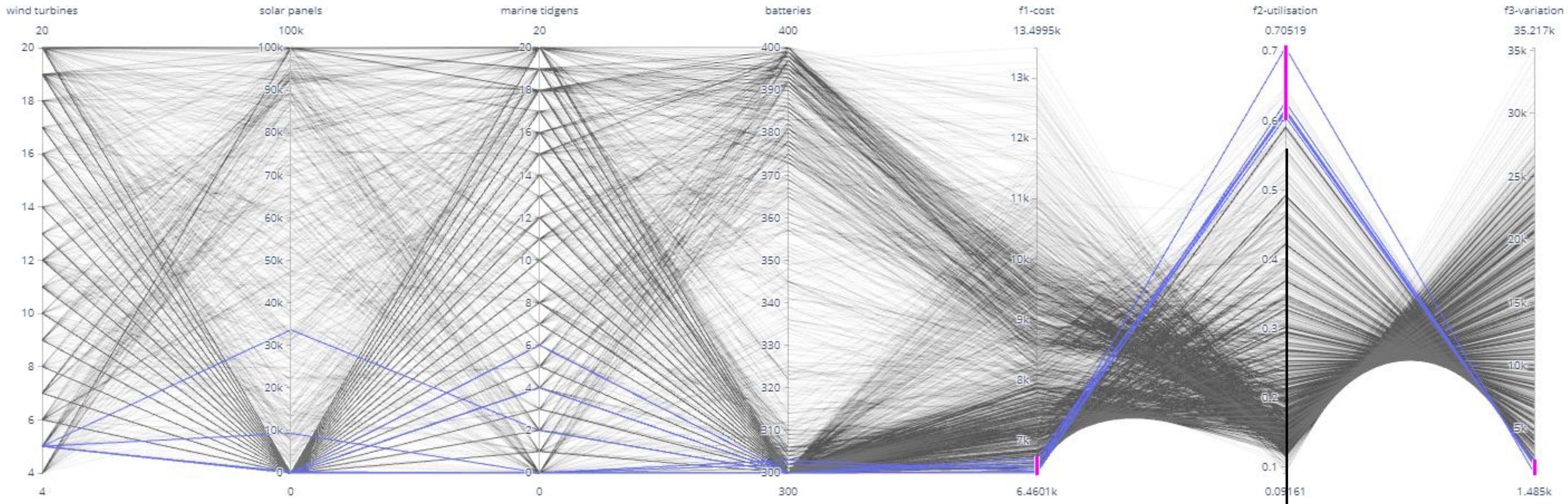
brightness – cost   size – utilization



high cost, low utilization

low cost, high utilization

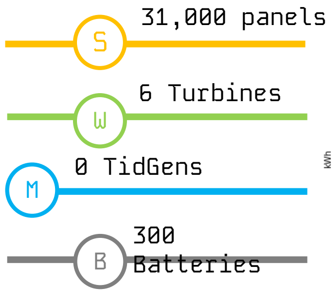
design variables   objectives

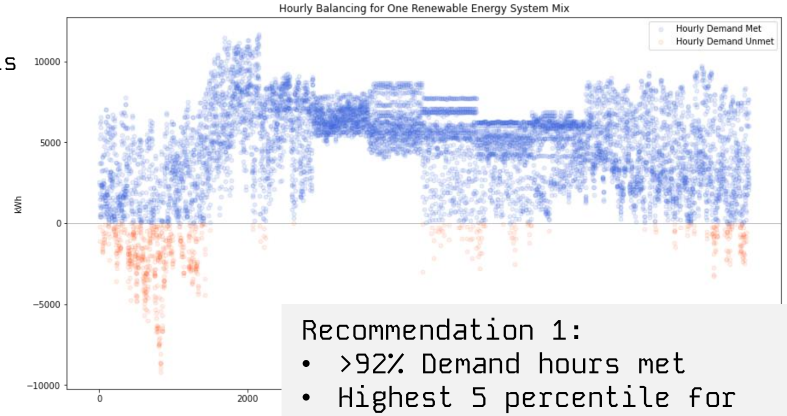


filter low cost

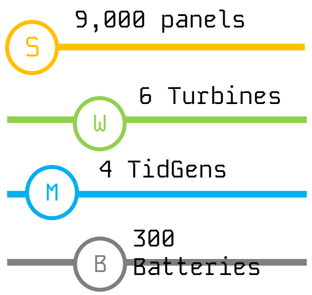
filter high utilization

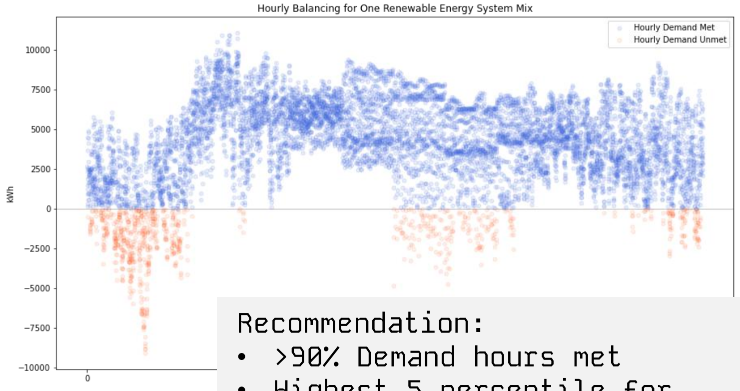
filter low variation





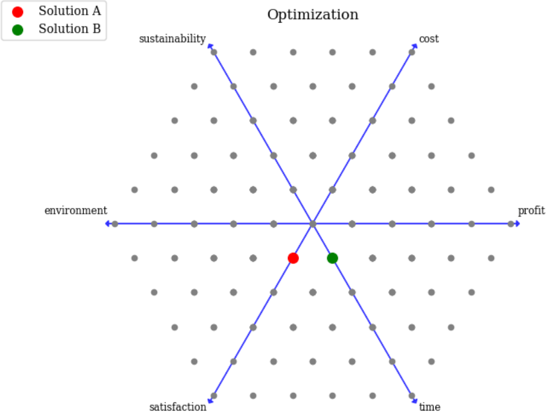
Recommendation 1:
- >92% Demand hours met
- Highest 5 percentile for utilization
- Lowest 5 percentile for cost

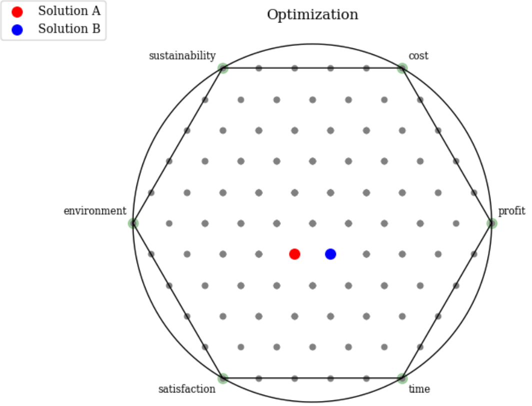


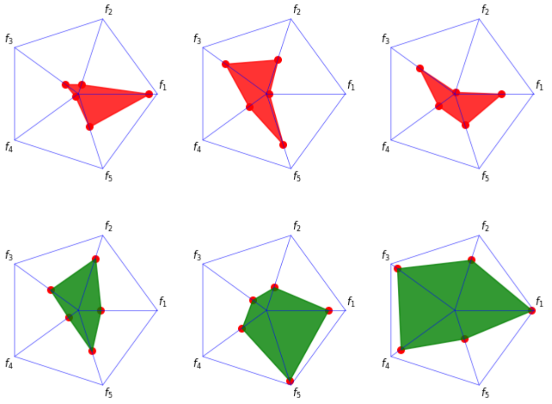


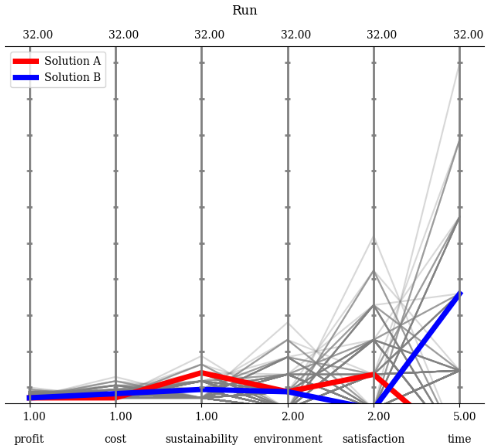
Recommendation:
- >90% Demand hours met
- Highest 5 percentile for utilization
- Lowest 5 percentile for cost

Pymoo









List Of Algorithms ¶

Algorithm	Class	Objective(s)	Constraints	Description
Genetic Algorithm	GA	single	x	A modular implementation of a genetic algorithm. It can be easily customized with different evolutionary operators and applies to a broad category of problems.
Differential Evolution	DE	single	x	Different variants of differential evolution which is a well-known concept for in continuous optimization especially for global optimization.
Biased Random Key Genetic Algorithm	BRKGA	single	x	Mostly used for combinatorial optimization where instead of custom evolutionary operators the complexity is put into an advanced variable encoding.
Nelder Mead	NelderMead	single	x	A point-by-point based algorithm which keeps track of a simplex with is either extended reflected or shrunk.
Pattern Search	PatternSearch	single	x	Iterative approach where the search direction is estimated by forming a specific exploration pattern around the current best solution.
CMAES	CMAES	single		Well-known model-based algorithm sampling from a dynamically updated normal distribution in each iteration.
Evolutionary Strategy	ES	single		The evolutionary strategy algorithm proposed for real-valued optimization problems.
Stochastic Ranking Evolutionary Strategy	SRES	single	x	An evolutionary strategy with constrained handling using stochastic ranking.
Improved Stochastic Ranking Evolutionary Strategy	ISRES	single	x	An improved version of SRES being able to deal dependent variables efficiently.
NSGA-II	NSGA2	multi	x	Well-known multi-objective optimization algorithm based on non-dominated sorting and crowding.
R-NSGA-II	RNSGA2	multi	x	An extension of NSGA-II where reference/aspiration points can be provided by the user.

L04.1
Design Space Exploration

DSE Basics

Random Search

Grid Search

L04.2
Optimization

Basics

Paper Plane
Design of Experiment

Genetic Algorithm

L04.3
Beyond Form

Rhino | Grasshopper

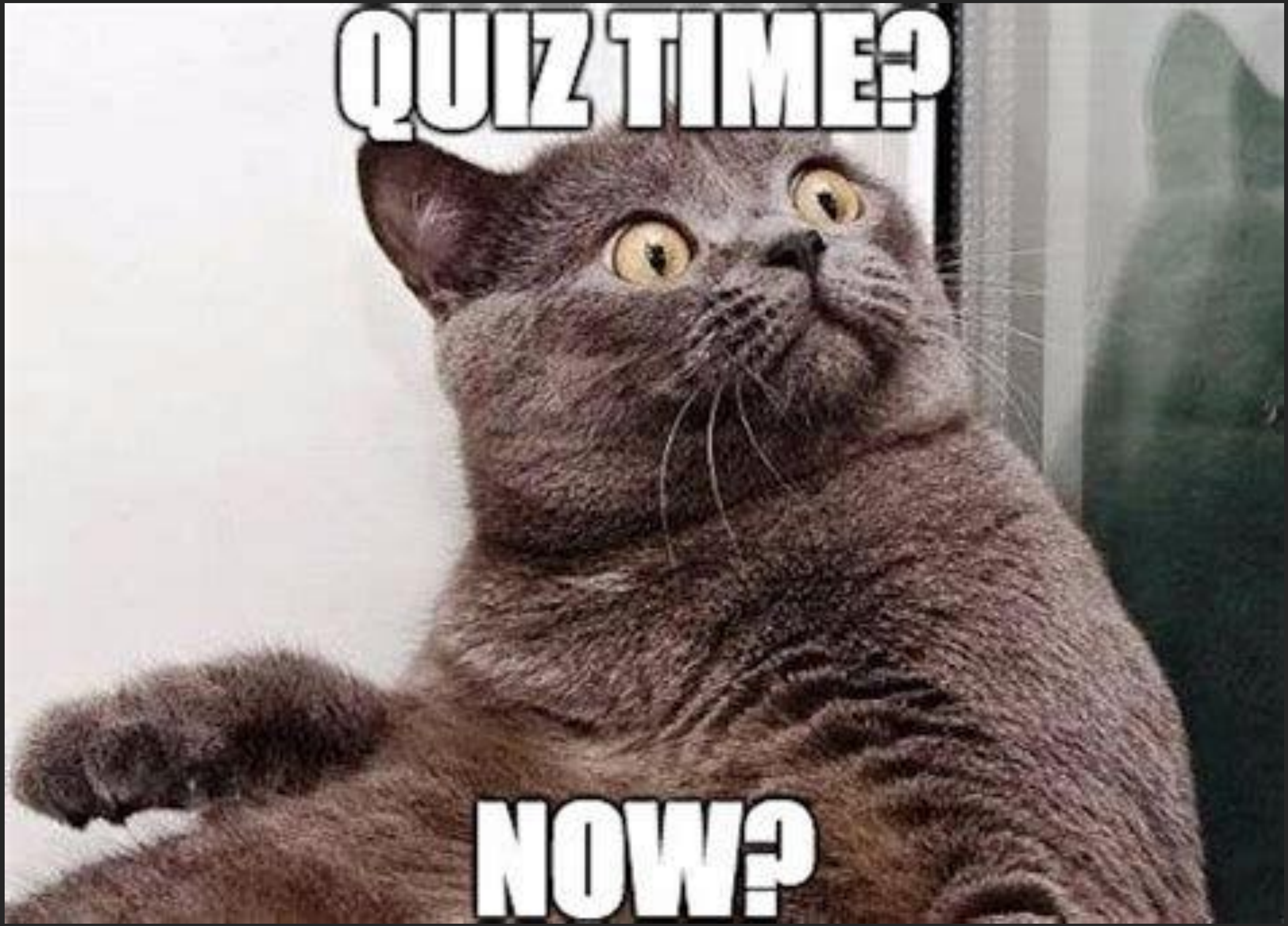
Galapagos

Nano Banana

Beyond Form

In-class activity

bit.ly/BPS5231-L4-BeyondForm


QUIZ TIME?
NOW?