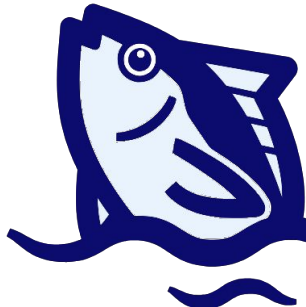
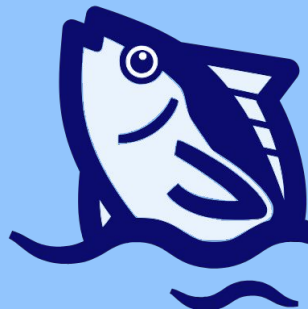
Tunny
Lecture & Hands-on



Before It Begins



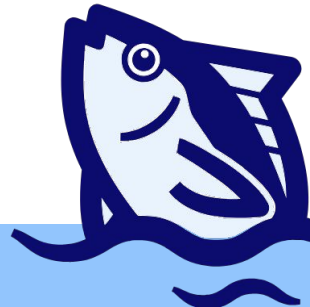
Components used in Hands-on

Required Software

- ➔ Rhinoceros 8 & Grasshopper
  - ◆ Windows is recommended
- ➔ Git
  - ◆ It's the Git software itself, not GitHub or GitLab

If you haven't installed it yet, please install it when you have time.

- ➔ Tunny (v1.3.2)



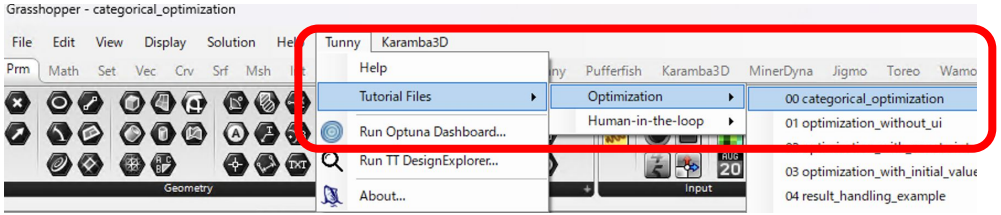
First: Set Up Your Tunny Environment

While Tunny offers numerous features, unlike other optimization components.

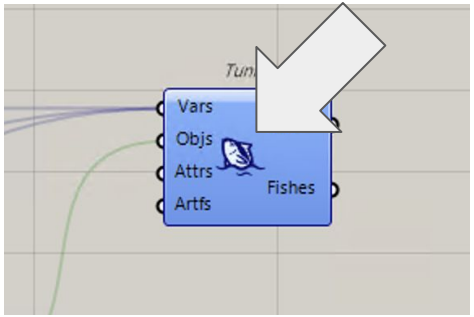
it requires setup process within Grasshopper. After opening the tutorial file, click the Tunny icon to run the setup.

Since the setup process takes some time, please complete this step before starting hands-on session.

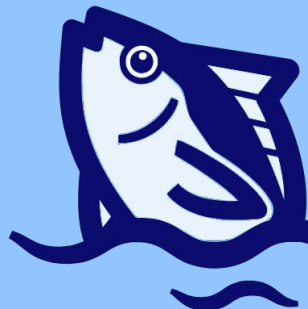
① Open Tutorial File



② Double Click!



Self-Introduction



Self-Introduction

Hiroaki NATSUME

2015.04–2019.12 Structural Engineer at Japanese architectural design firms

2020.01– hrntsm (Tunny development & support)

2020.11 Shanghai International WS Technical Tutor

2022.04 Architectural Information Society Annual Academic Conference
Panelist

2022.07 Digital FUTURES2022 Tutor

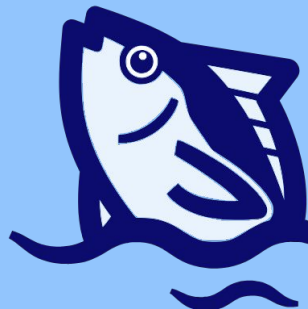
2022.08 Architectural Information Society Session vol11 Tutor

2024.03 Georgia Institute of Technology Spring Seminar Guest Lecturer

Tunny developer

I work as a computational designer in architectural design and mechanical engineering for vehicles and heavy industry.

Lecture Session



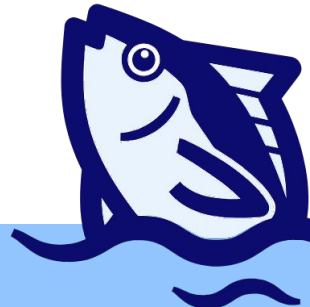
Today's Theme

I will introduce "**Tunny**", a "**design space exploration tool**" that operates on Grasshopper, along with its background knowledge and software integration.

Tunny's main feature is "**black box optimization**."

Before explaining the detailed functions of Tunny,

I will first talk about optimization.

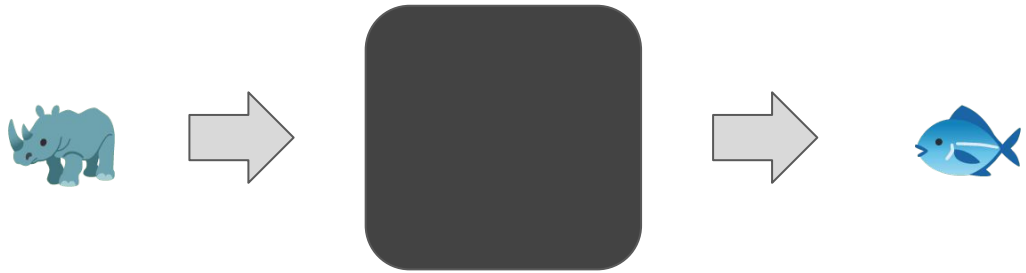


What is Black-Box Optimization?

This is an optimization for problems where only the result is returned when a value is input.

Since the operation is a black box and it is unclear what operations produce that value, it is called "**black-box optimization.**"

Optimization using Grasshopper is categorized as black-box optimization because the outcome remains unknown until the calculation is performed within Grasshopper.

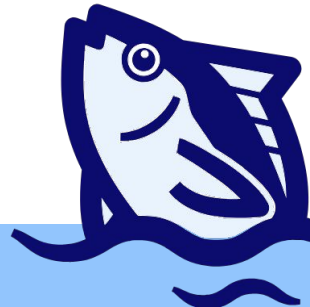


What is "**Not**" Black-Box Optimization?

For problems that can be clearly expressed as equations, such as the following, the process and resulting output can be calculated using methods other than black-box optimization.

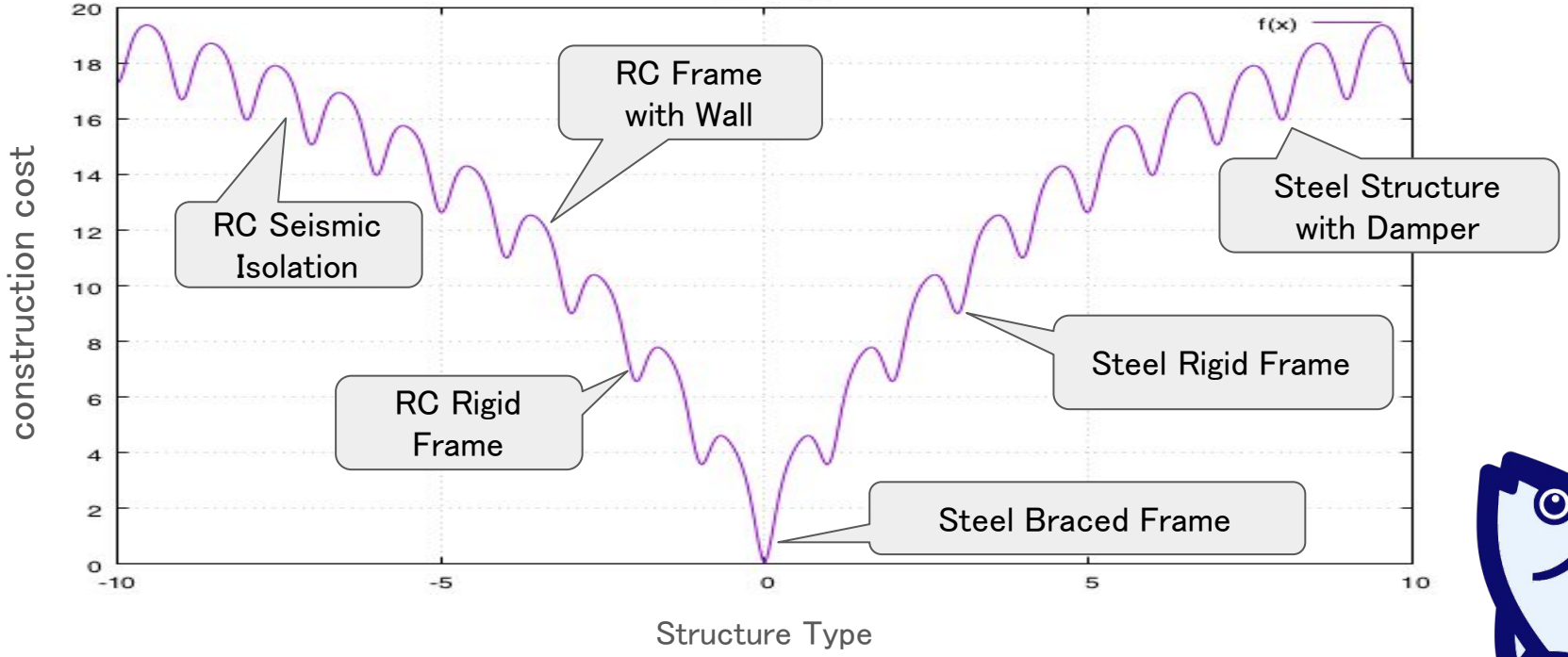
Problems that can be clearly expressed as mathematical equations allow computers to automatically perform operations like differentiation and rapidly search for minimum values.

$$f_x = x^2 + x + 1$$



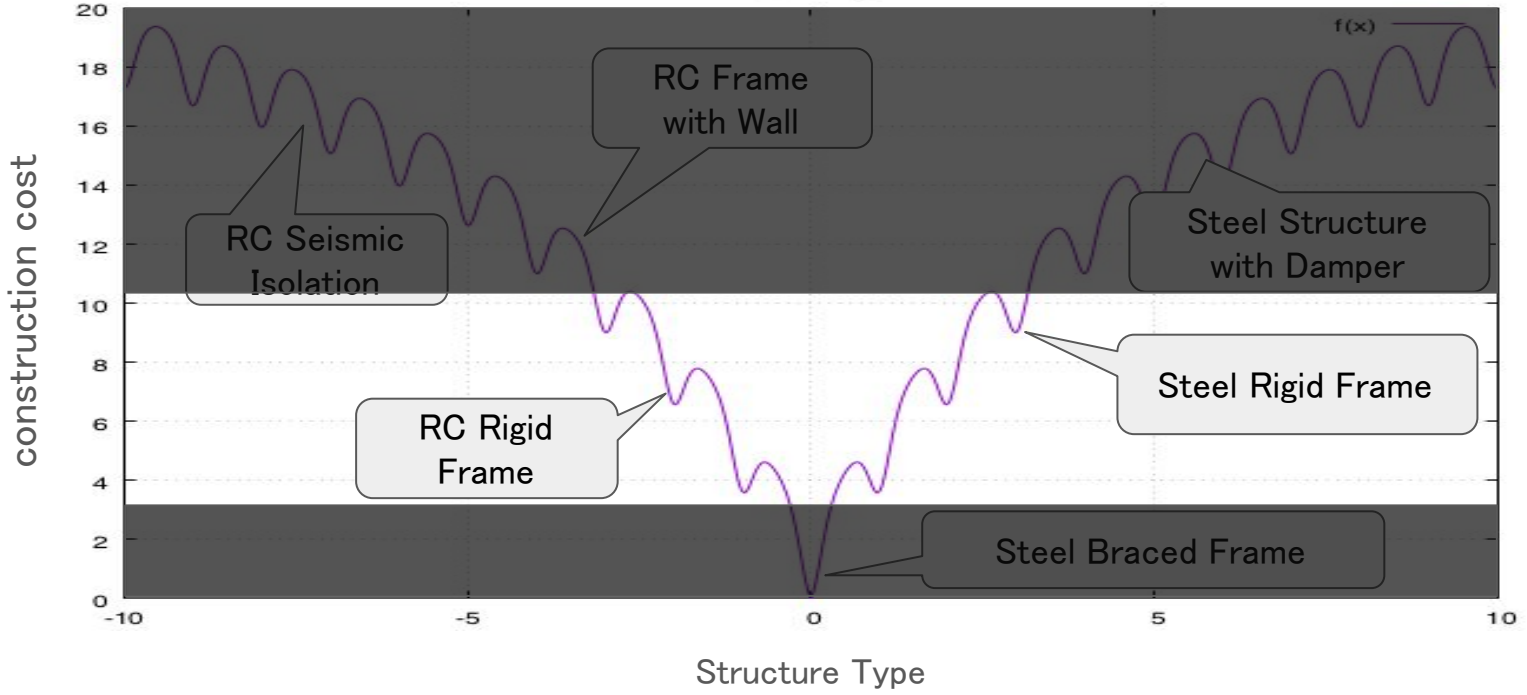
Is Design an Optimization Problem?

Generally, it is said that the optimization of design problems is a constrained optimization problem with multiple, often conflicting objectives.



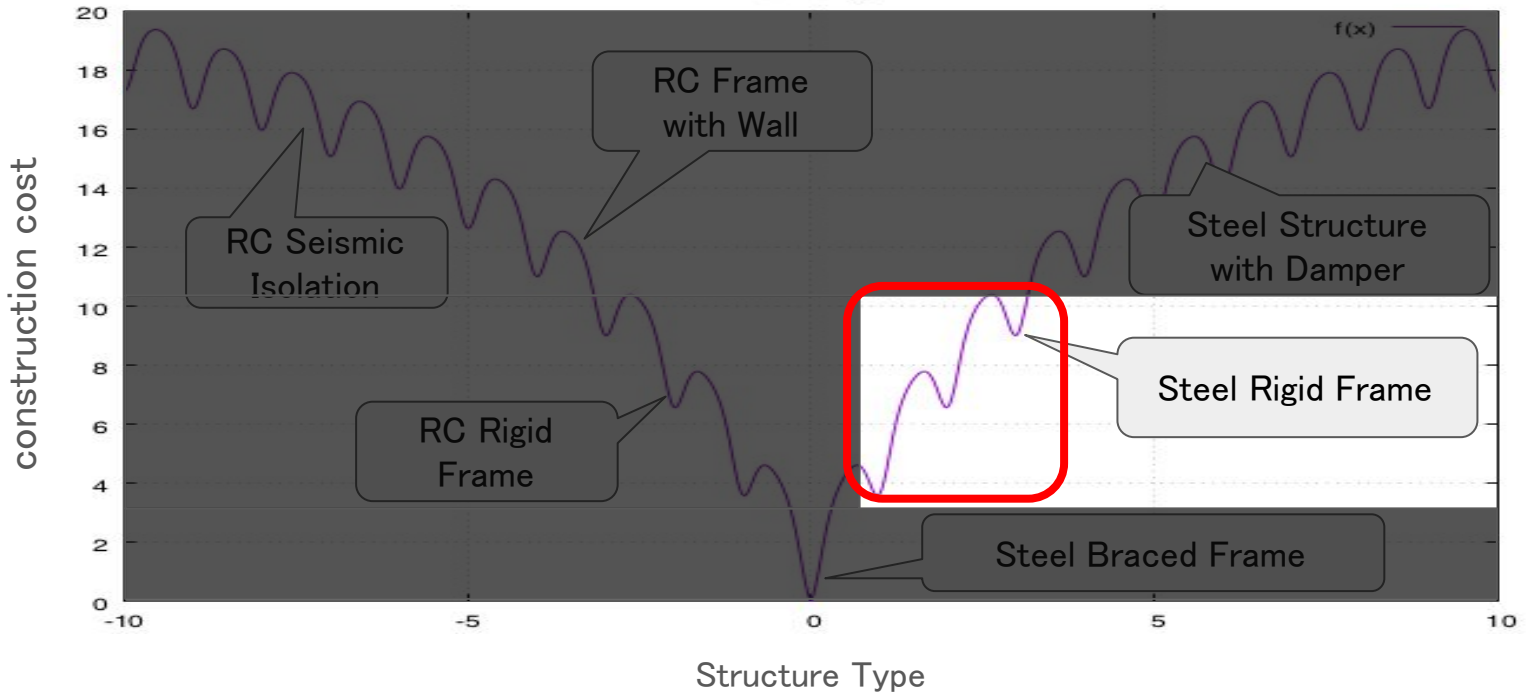
Is Design an Optimization Problem?

For example, "Rigid Frame structures" are effective for considering planar freedom under the constraint of "office" space



Is Design an Optimization Problem?

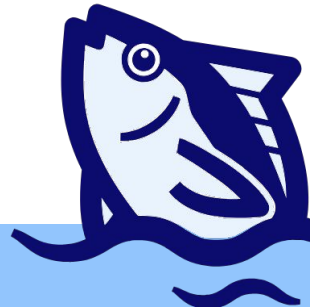
For instance, in this current project, concrete is difficult to secure, so material constraints forced us to use "Steel Rigid Frame"



Is Design an Optimization Problem?

In this example explanation, we determined the structural form very simply, but in the actual design process, it is decided through coordination with many stakeholders.

Such actions are not well-suited for optimization. It is something the designer themselves should do.



What to optimize in the design?

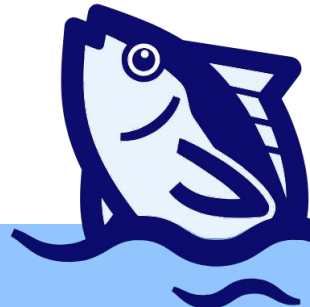
While computer optimization is excellent at performing detailed repetitive calculations, it struggles with condition organization as discussed earlier.

However, within a fixed framework, it is ″skilled at″ fine-tuning and quantification such as:

- ➔ How many column and beam sections should we use?
- ➔ Where should we place shear resistance elements?
- ➔ How should the members be arranged?

Let's tackle optimization with a clear understanding of what it does and doesn't do well.

The hands-on following the lecture will showcase several case studies.



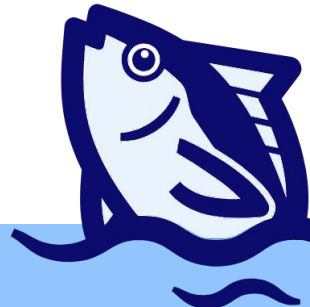
Is optimization even necessary?

Changing the subject, is seeking a single optimal result always necessary in the first place?

Have you ever wondered about things like the following? This requires statistical analysis, not optimization.

- ➔ Which Grasshopper slider values affect the cost?
- ➔ How does beam span affect safety?
- ➔ How does window opening affect annual daylight gain?

Tunny not only optimizes but also supports such statistical processing by broadly exploring the design space.



What software can we use for optimization?

There are various optimization tools available for design applications, not just within Grasshopper.

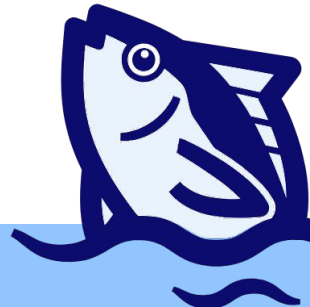
➔ Commercial optimization platforms primarily for mechanical design
- ◆ Enterprise-level solutions that are very expensive, but offer high versatility and comprehensive support
  - modeFrontier (ESTECO)
  - HEEDS (Siemens)
  - Optimus (noesis)
  - Isight (Dassault)

➔ Optimization plugins for Grasshopper
- ◆ Typically free, with community support available
  - Galapagos
  - Wallacei
  - Opossum
  - Octopus
  - Tunny (today's focus)



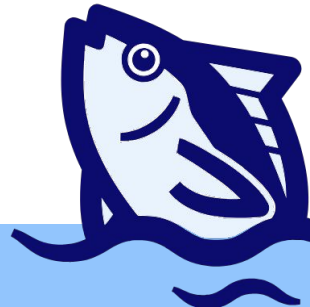
What Led Me to Develop Tunny

- ➔ Genetic Algorithms (GA), commonly used in architectural design, require a large number of iterations to converge
- ➔ Before developing Tunny, I was using Wallacei, which only supports NSGA-II
  - ◆ Back then, many of my projects took over a week per optimization run
- ➔ This timeline was impractical for most project schedules, so I needed a faster-converging optimization tool
- ➔ During an internal Bayesian optimization study session, a colleague remarked, "This doesn't seem too hard to build"
  - ◆ As a Grasshopper plugin developer, I had always wanted to try implementing an optimization engine
  - ◆ That was the push I needed—development began!



Tunny Development Timeline

- ➔ Early 2022 – Development began
- ➔ 2022/04/14 – v0.1 (First release 🐟)
- ➔ 2022/05/02 – v0.2 (Added grid sampling and GenePool support)
- ➔ 2022/06/05 – v0.3 (Added Optuna-Dashboard integration)
- ➔ 2022/07/09 – v0.4 (Added attribute support and component coloring)
- ➔ 2022/09/03 – v0.5 (Added constraint support and advanced optimization settings)
- ➔ 2022/12/23 – v0.6 (Added FishEgg for initial population seeding)
- ➔ 2023/03/21 – v0.7 (Added journal-based result storage)
- ➔ 2023/07/24 – v0.8 (Added human-in-the-loop and NSGA-III)
- ➔ 2023/12/30 – v0.9 (Added preferential Bayesian optimization)
- ➔ 2024/01/27 – v0.10 (Stability improvements)
- ➔ 2024/03/20 – v0.11 (Added new BO-GP and categorical optimization)
- ➔ 2024/06/22 – v0.12 (Added TT-DesignExplorer and top menu integration)



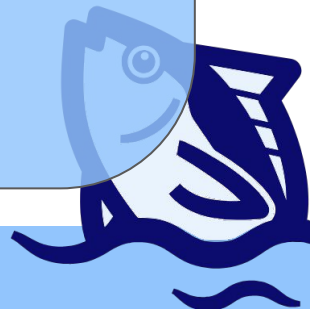
Tunny Development Timeline

- Early 2022 – Development began
- 2022/06/05 – v0.3 (Added Optuna-Dashboard integration)
- 2022/07/09 – v0.4 (Added attribute support and component coloring)
- 2022/09/03 – v0.5 (Added constraint support and advanced optimization settings)
- 2022/12/23 – v0.6 (Added FishEgg for initial population seeding)
- 2023/12/30 – v0.9 (Added preferential Bayesian optimization)
- 2024/01/27 – v0.10 (Stability improvements)
- 2024/03/20 – v0.11 (Added new BO-GP and categorical optimization)

With the incredible support from the community

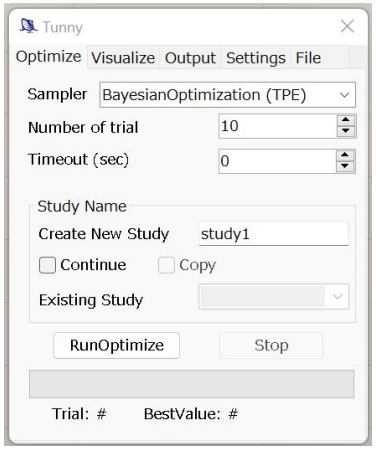
2025/08 – v1.0 finally released!

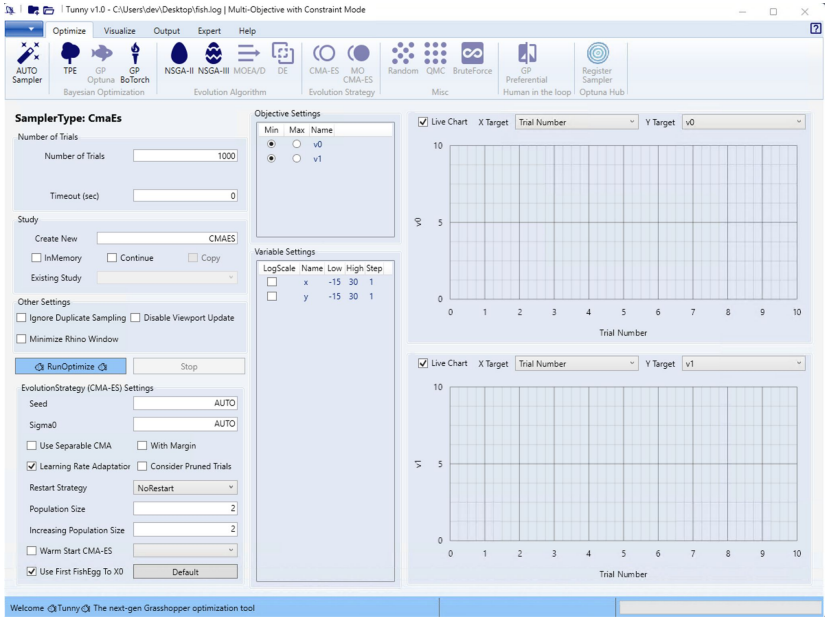
Today, Tunny is used by worldwide



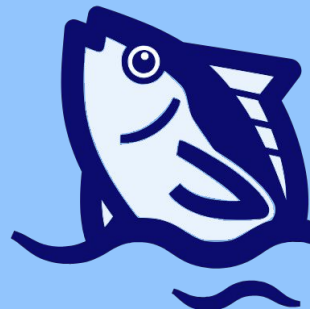
Evolution from v0 to v1

The interface evolved from a plain, text-based UI to a modern, icon-rich ribbon interface.
This dramatically improved feature discoverability and usability, bringing Tunny closer to the polished, user-friendly experience of commercial software.





Key Features of Tunny



Optimization + Design Space Exploration

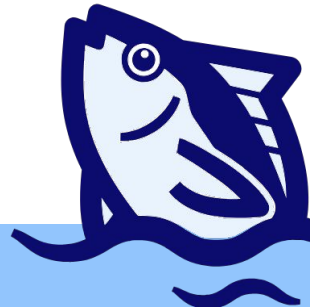
Tunny offers much more than basic optimization—it provides a comprehensive suite of features. Our goal is to bridge the gap between Grasshopper optimization plugins like others and enterprise-level platforms, offering the best of both worlds.

➔ Sampling Techniques

- AutoSelection
- BO-TPE
- BO-GP
- HEBO
- NSGA-II
- NSGA-III
- MOEA/D
- DE
- CMA-ES
- INGO
- Random
- Quasi-MonteCalro
- BrutoForce

➔ Visualization Tools

- Histroy
- ParetoFront
- ParallelCoordinate
- Slice
- Contour
- Rank
- Timeline
- EDF
- Importance
- Optuna-Dashboard
- TT-DesignExplorer



Expanded Exploration Techniques

Tunny supports a significantly wider range of optimization algorithms than other Grasshopper plugins:

- ➔ **Auto Sampler** – Automatic algorithm selection
- ➔ **Differential Evolution** – Evolutionary strategy for global optimization
- ➔ **MOEA/D** – Specialized multi-objective optimization
- ➔ **MO-CMA-ES** – Multi-objective variant of CMA-ES
- ➔ **c-TPE** – Constraint Enhanced TPE
- ➔ **HEBO** – Fast-converging Bayesian optimization
- ➔ **INGO** – Improved CMA-ES variant

Coming Soon:

- ➔ **Gray Wolf Optimization** – Swarm-based algorithm inspired by wolf hunting behavior
- ➔ **Whale Optimization** – Swarm-based algorithm inspired by whale hunting behavior

We continually expand our algorithm library based on user needs and research developments. Feature requests and contributions are always welcome!

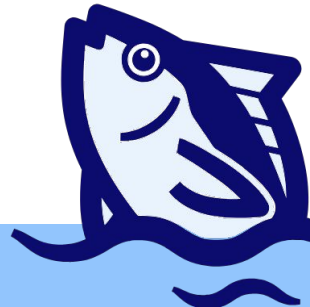
Powered by Optuna: State-of-the-Art Optimization Engine

Tunny uses **Optuna** as its optimization engine—a leading hyperparameter optimization framework trusted by researchers and industry worldwide.

Key Features of Optuna:

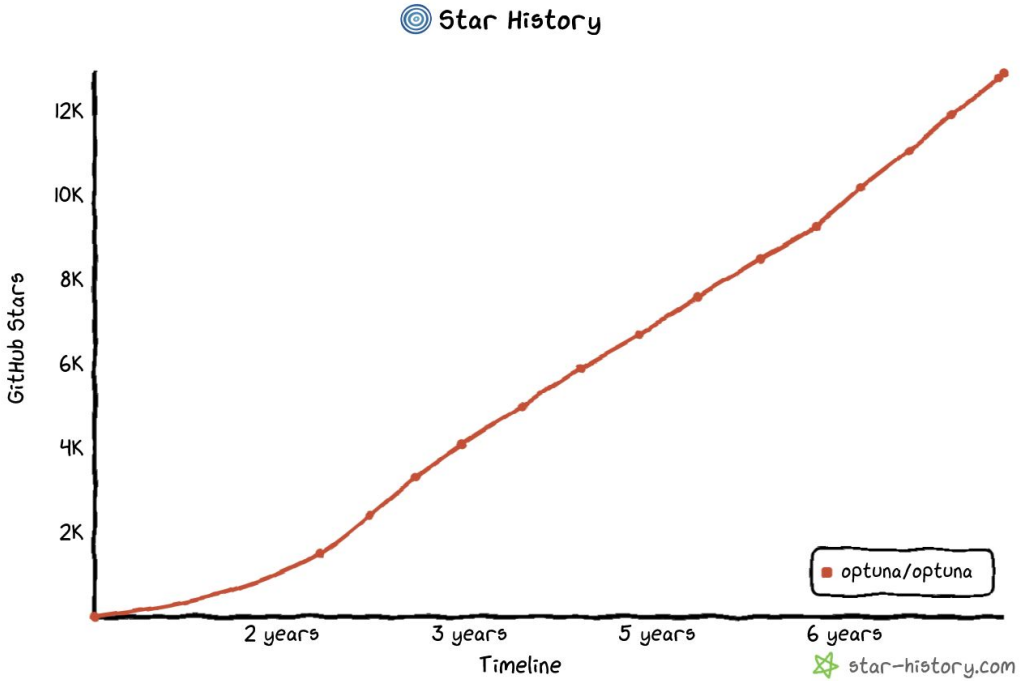
- ➔ **State-of-the-art algorithms:** Efficiently searches large spaces with advanced methods like TPE, CMA-ES, and Gaussian processes
- ➔ **Flexible search spaces:** Define complex optimization problems using simple Python-like logic
- ➔ **Easy parallelization:** Scale optimization across multiple cores without additional code
- ➔ **Framework agnostic:** Used across PyTorch, TensorFlow, scikit-learn, and now—Grasshopper design optimization

By leveraging Optuna, **Tunny brings cutting-edge optimization technology directly to your parametric design workflow.**



Optuna: Actively Developed and Widely Trusted

Optuna is actively developed and widely trusted, with over 13,000 GitHub stars and continuous growth in the research and industry communities.

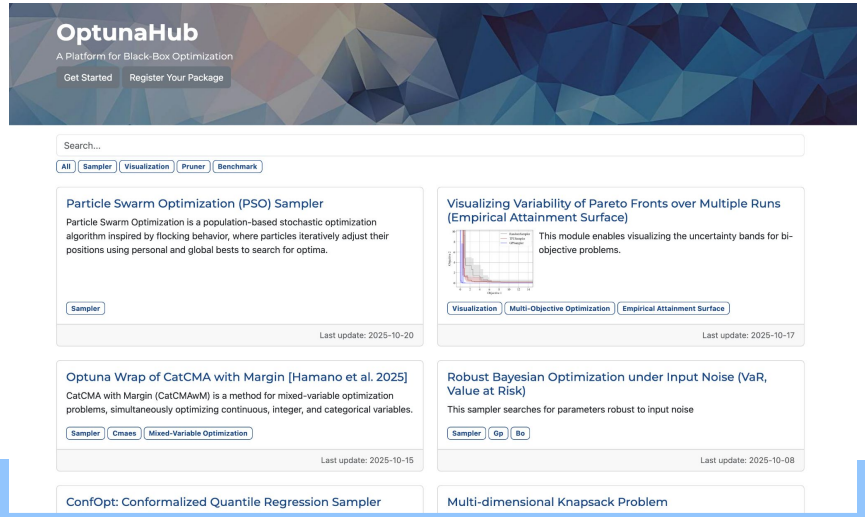


Hub of Optimization Methods

The Optuna team maintains a website called OptunaHub where you can publish your own optimization programs.

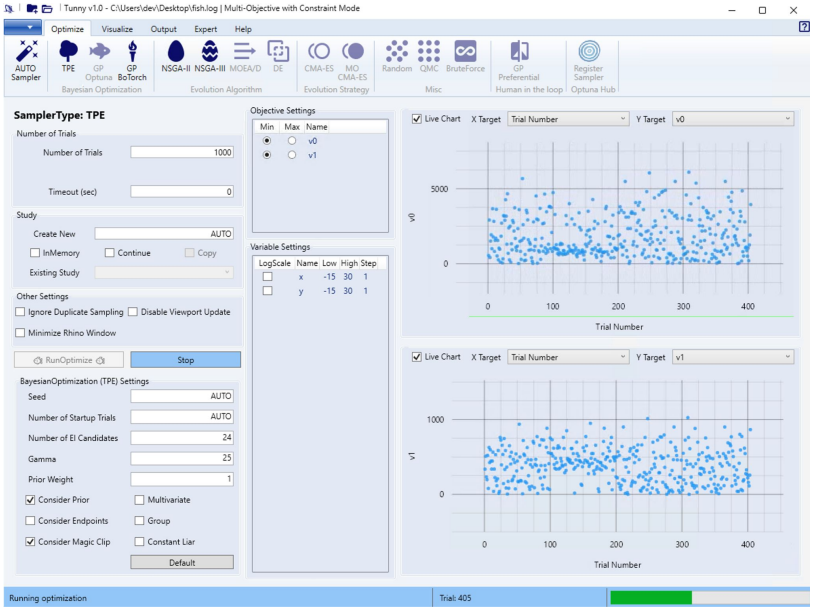
It's similar to Food4rhino for Rhino.

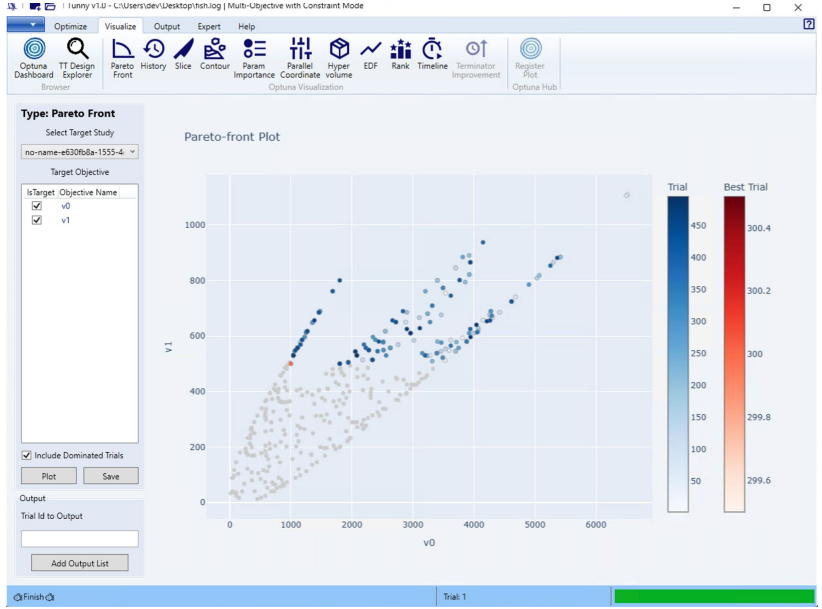
Since Tunny uses Optuna as its optimization engine, registering your optimization method here allows you to use it within Grasshopper via Tunny.

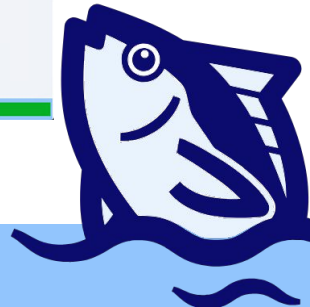


Built-in Result Analysis

Quick result checks are available right in the Tunny UI—no dashboard required.

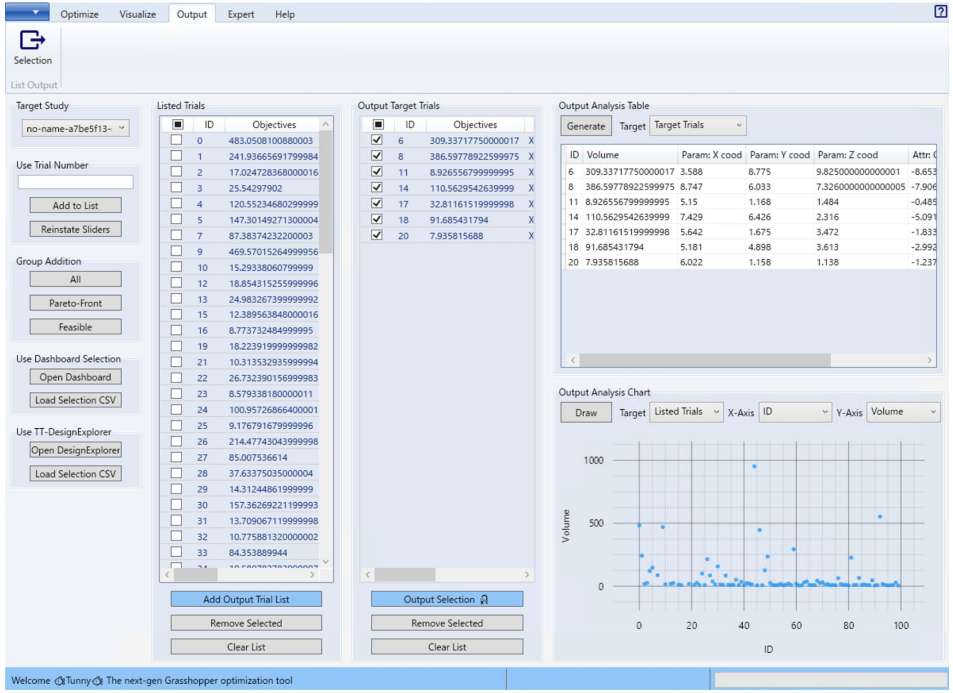






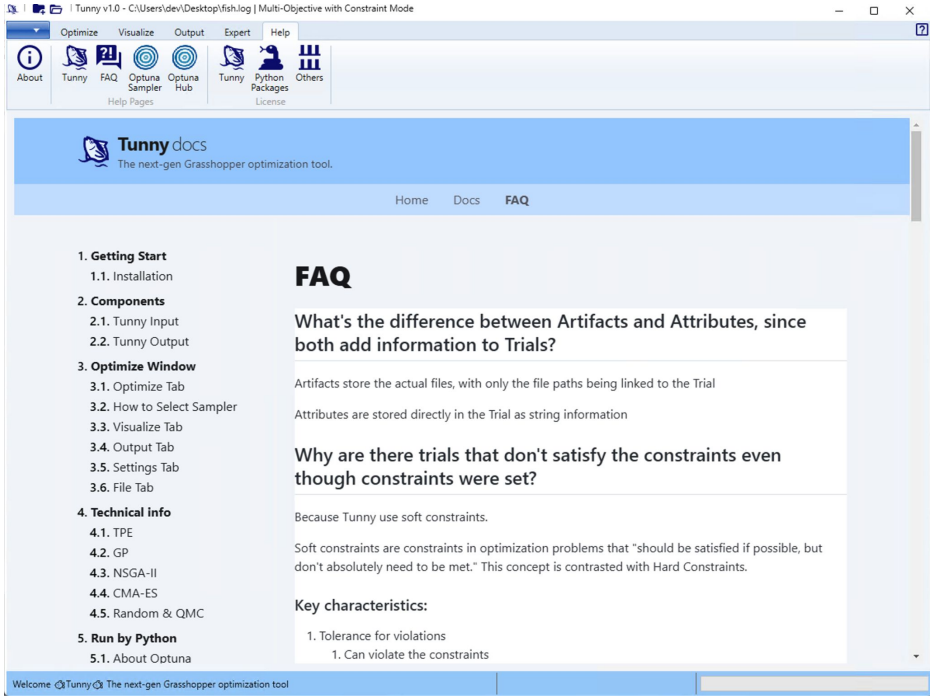
Wallacei-Style Result Output Interface

The interface provides Wallacei-style result handling. You can also import selections from Optuna Dashboard or TT-DesignExplorer back into Tunny for output control.



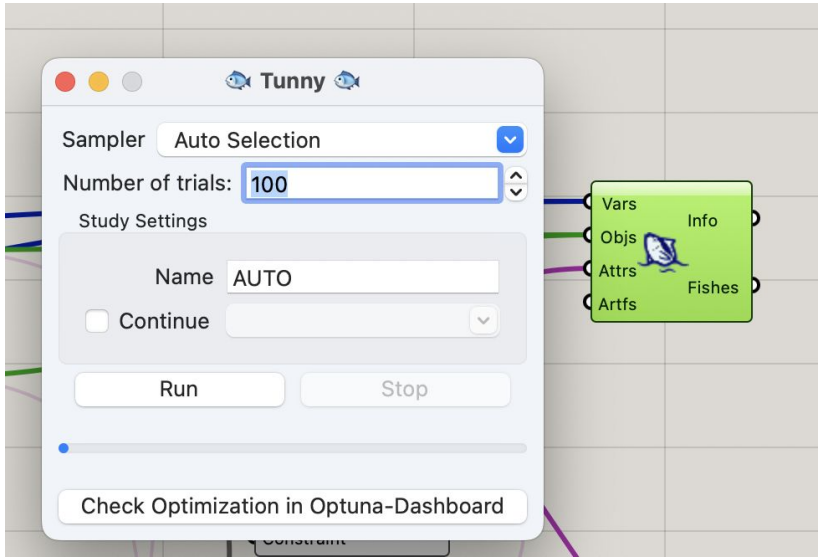
Easy Access to Help

Access documentation, FAQs, and AI-powered support directly from the ribbon interface.



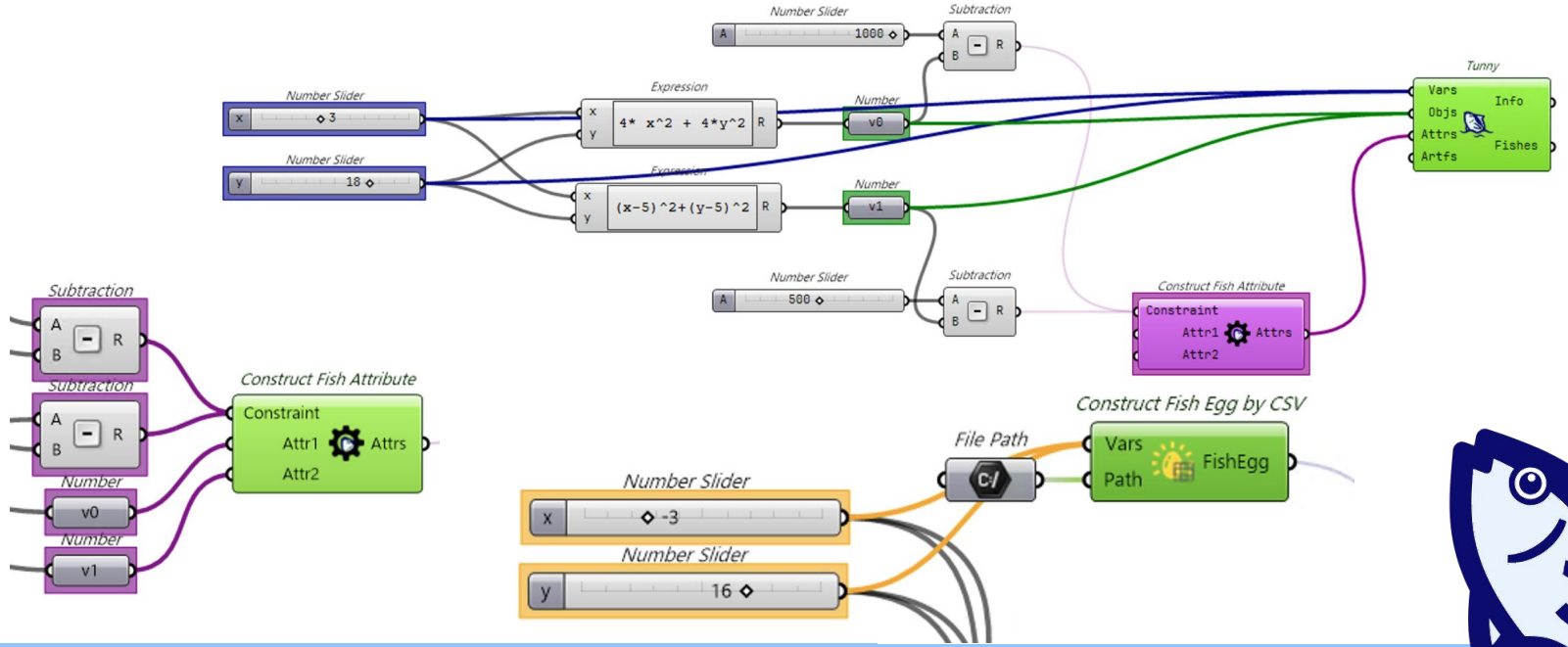
Cross-Platform Support

While not feature-identical to the Windows version, Tunny also supports macOS. In fact, Tunny is one of the few Grasshopper plugins for macOS that supports both constraints and multi-objective optimization.



Visual Feedback Through Color-Coded Components

Each function is color-coded, making it easy to see what role each part of the Grasshopper file plays.

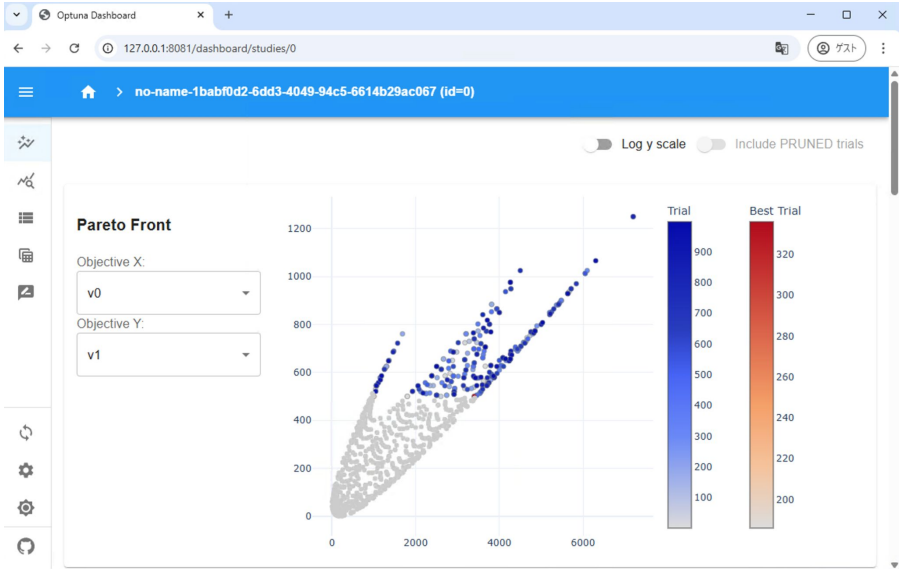


Result Persistence and Dashboard

Tunny saves optimization results as a text file, separate from the Grasshopper document.

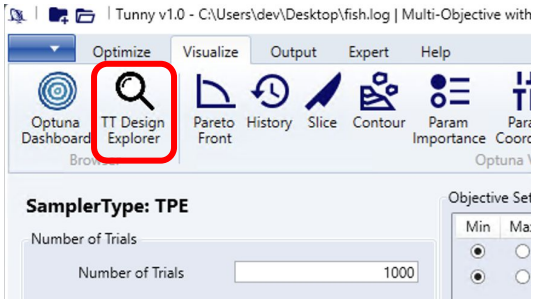
By default, results are saved to a fish.log file on your desktop.

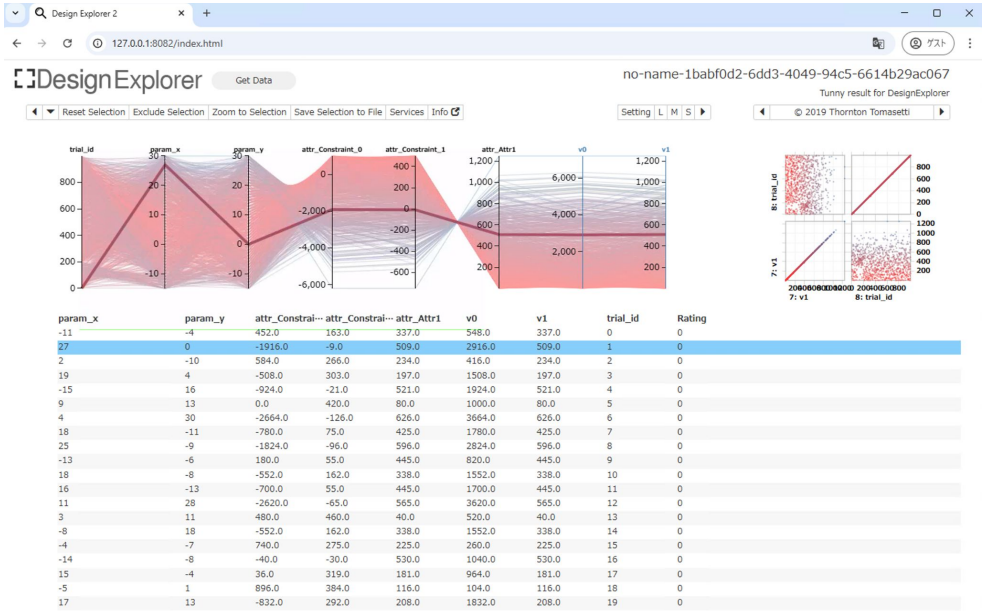
Once saved, you can view and explore your optimization results through a web-based dashboard—even without Grasshopper. This allows you to analyze results and present to stakeholders anywhere.



TT-DesignExplorer Integration

Tunny seamlessly exports to Design Explorer by Thornton Tomasetti.
If you're already familiar with this industry-standard analysis tool, you can continue using it to explore your Tunny optimization results.

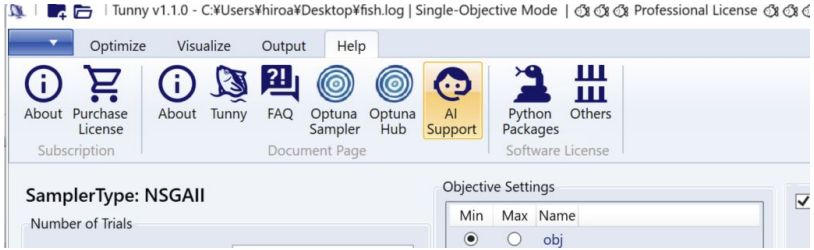


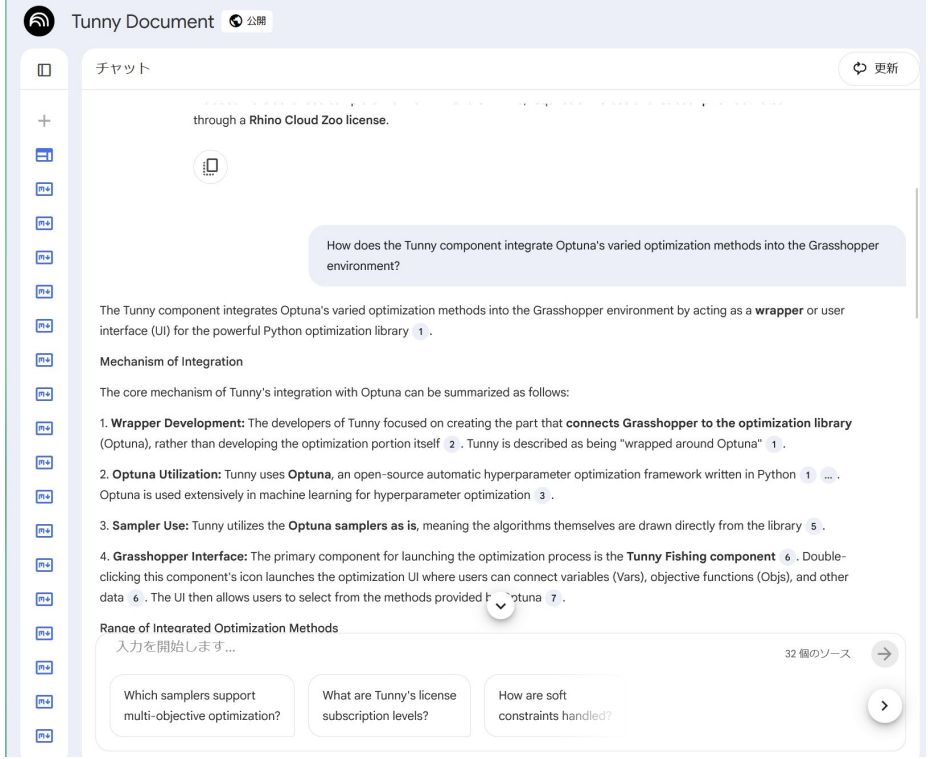


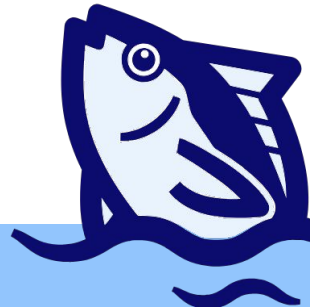
AI-Powered Support

With so many features, it can be challenging to master everything Tunny offers.

Our AI assistant lets you ask questions about any feature in natural language, making it easy to get the help you need.



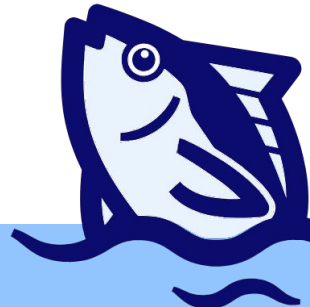




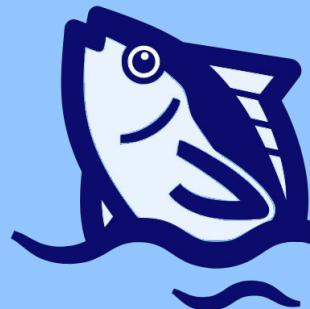
Important Notes

Users have reported that Tunny may not function properly in the following environments. We are aware of these issues and plan to address them in upcoming releases:

- ➔ Grasshopper running within Rhino.Inside.Revit
- ➔ Grasshopper definitions that include Python3 components
- ➔ Rhino where Wallacei is installed
  - ◆ though in most cases, Tunny works fine even with Wallacei installed

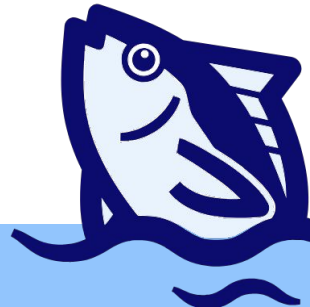


Understanding Sampling Methods



Tunny includes many available methods

- Sampling Methods
  - AutoSelection
  - BO-TPE
  - BO-GP
  - HEBO
  - NSGA-II
  - NSGA-III
  - MOEA/D
  - DE
  - CMA-ES
  - INGO
  - Random
  - Quasi-MonteCalro
  - BrutoForce
- Visualization Methods
  - Histroy
  - ParetoFront
  - ParallelCoordinate
  - Slice
  - Contour
  - Rank
  - Timeline
  - EDF
  - Importance
  - Optuna-Dashboard
  - TT-DesignExplorer



Tunny includes many available methods.

- Sampling Methods
  - AutoSelection
  - BO-TPE
  - BO-GP
  - HEBO
  - NSGA-II
  - NSGA-III
  - MOEA/D
  - DE
  - CMA-ES
  - INGO
  - Random
  - Quasi-MonteCalro
  - BrutoForce

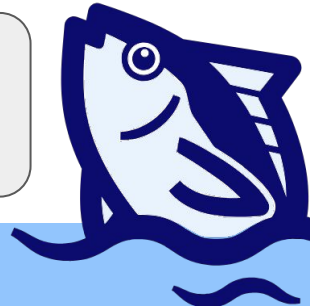
Automatic selection of high-performing algorithms

Bayesian optimization.
Achieve fast convergence by building surrogate models.

Evolutionary algorithms.
Wallacei also includes NSGA-II, which, while slower to converge, provides reliable performance across a wide range of problems.

Evolution Strategies.
Currently limited to single-objective problems in Tunny, but excels at handling highly multimodal landscapes with very fast convergence.

Sampling-only methods for exploring relationships between variables and objectives.



Tunny includes many available methods.

- Sampling Methods
  - AutoSelection
  - BO-TPE
  - BO-GP
  - HEBO
  - NSGA-II
  - NSGA-III
  - MOEA/D
  - DE
  - CMA-ES
  - INGO
  - Random
  - Quasi-MonteCalro
  - BruteForce

Automatic selection of high-performing algorithms

Bayesian optimization.
Achieve fast convergence by building surrogate models.

Evolutionary algorithms.
Wallacei also includes NSGA-II, which, while slower to converge, is reliable for performance across a wide range of problems.

Evolution Strategies.
Currently limited to single-objective problems in Tunny, but excels at handling highly multimodal landscapes with very fast convergence.

Sampling-only methods for exploring relationships between variables and objectives.

🤩 **"So many options—perfect! I can always find the best algorithm for optimal performance!"**

😵 **"Wait, there are HOW many? This is overwhelming... I don't know where to start..."**

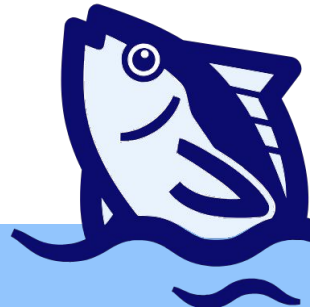
Which one are you?

So many options—awesome!
But wait... which one do I actually need?

- ➔ GA like NSGA-II is common in other Grasshopper plugins too
  - ◆ Should I just stick with this one?
- ➔ Can I always use TPE, Tunny's top recommendation?
- ➔ What's the point of non-optimizing methods likeRandom or QMC?
- ➔ There are so many visualization options—which should I use?

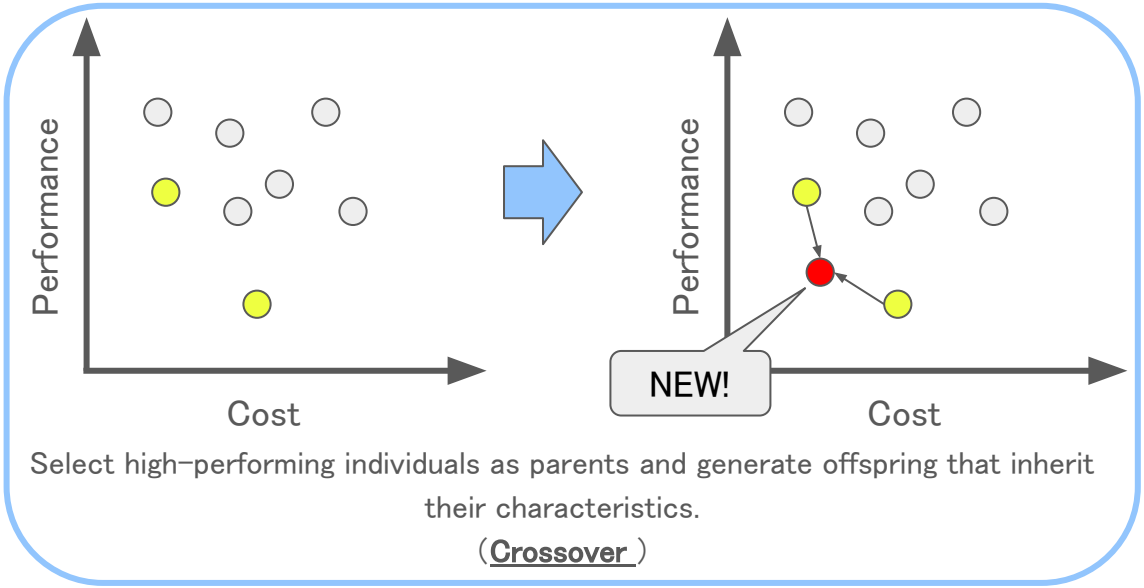
Let's explore three key algorithms in Tunny: NSGA-II, TPE, and QMC.

But first: v1 introduces "**Auto Sampler**", which automatically selects the best algorithm for you. We recommend starting there!



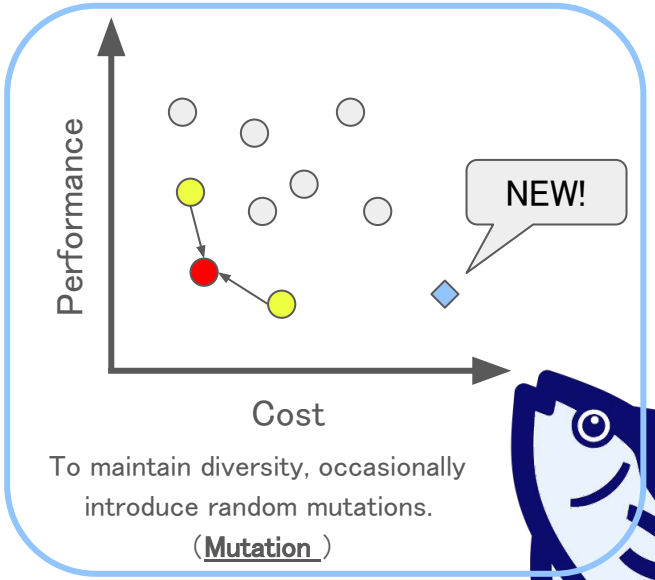
Genetic Algorithms (NSGA-II, NSGA-III)

This is probably the most familiar algorithm to many users. Inspired by biological evolution, it's widely available in other Grasshopper optimization tools like Galapagos, Wallacei, and Octopus. Like natural evolution, results improve gradually over time.



Select high-performing individuals as parents and generate offspring that inherit their characteristics.

(**Crossover**)



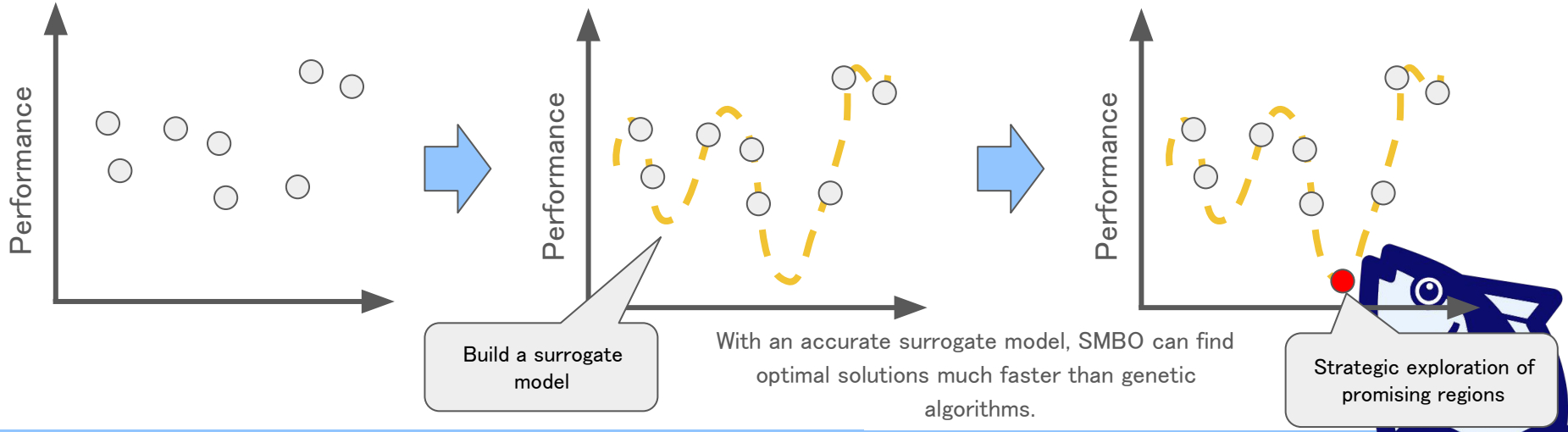
To maintain diversity, occasionally introduce random mutations.

(**Mutation**)

SMBO - Bayesian Optimization Methods (TPE, GP)

This approach may be less familiar to many. SMBO stands for Sequential Model-Based Optimization.

Think of it like creating a trend line or curve fit in Excel—the algorithm builds a surrogate model (an approximation function) of your design space, then uses this model to efficiently guide the optimization process.

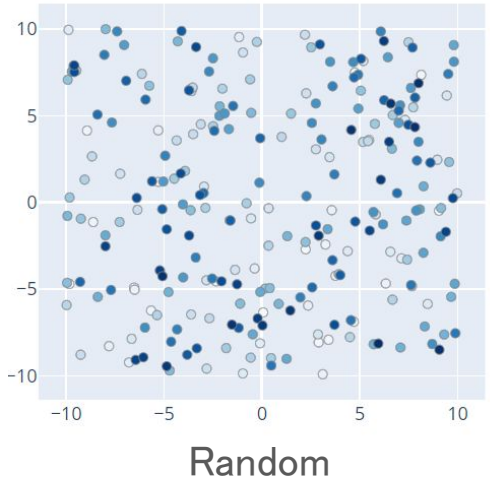


QMC (Quasi-Monte Carlo): Low-Discrepancy Sampling

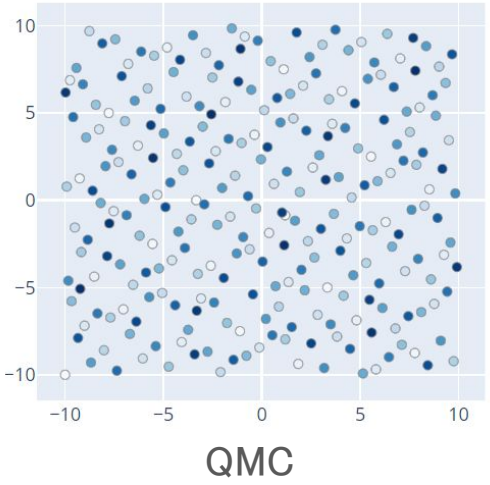
This is especially useful for machine learning and design space exploration, which we'll discuss later. QMC minimizes sampling bias through highly uniform space coverage.

Q: Doesn't random sampling avoid bias too?

A: Random sampling is only truly uniform with infinite samples. In practice, it creates clusters and gaps. QMC uses mathematical theory to distribute samples more evenly with fewer points.

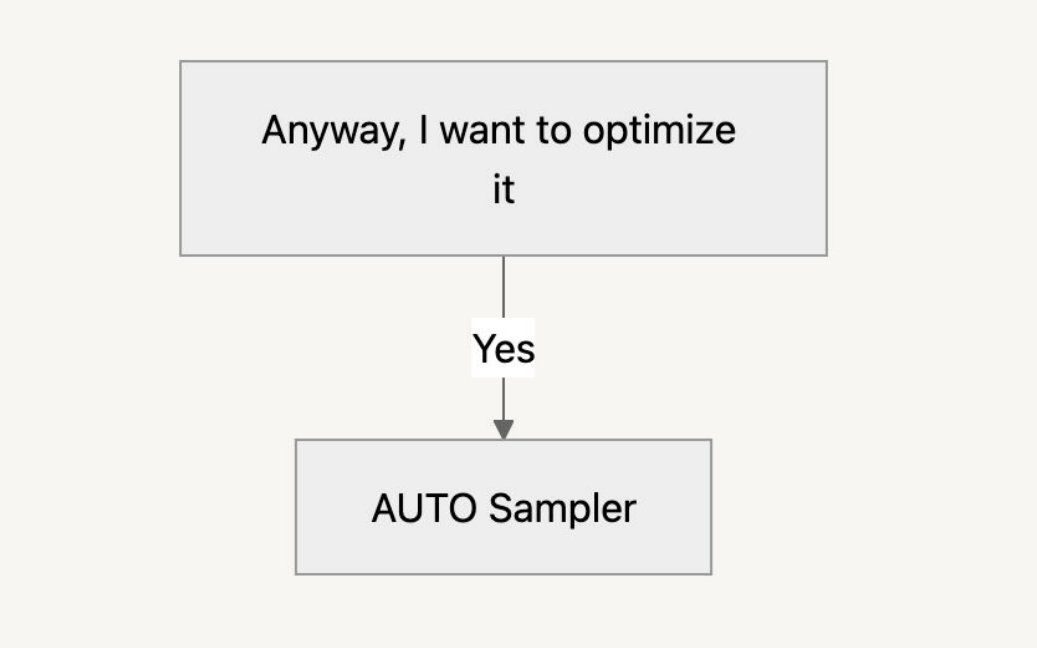


Random

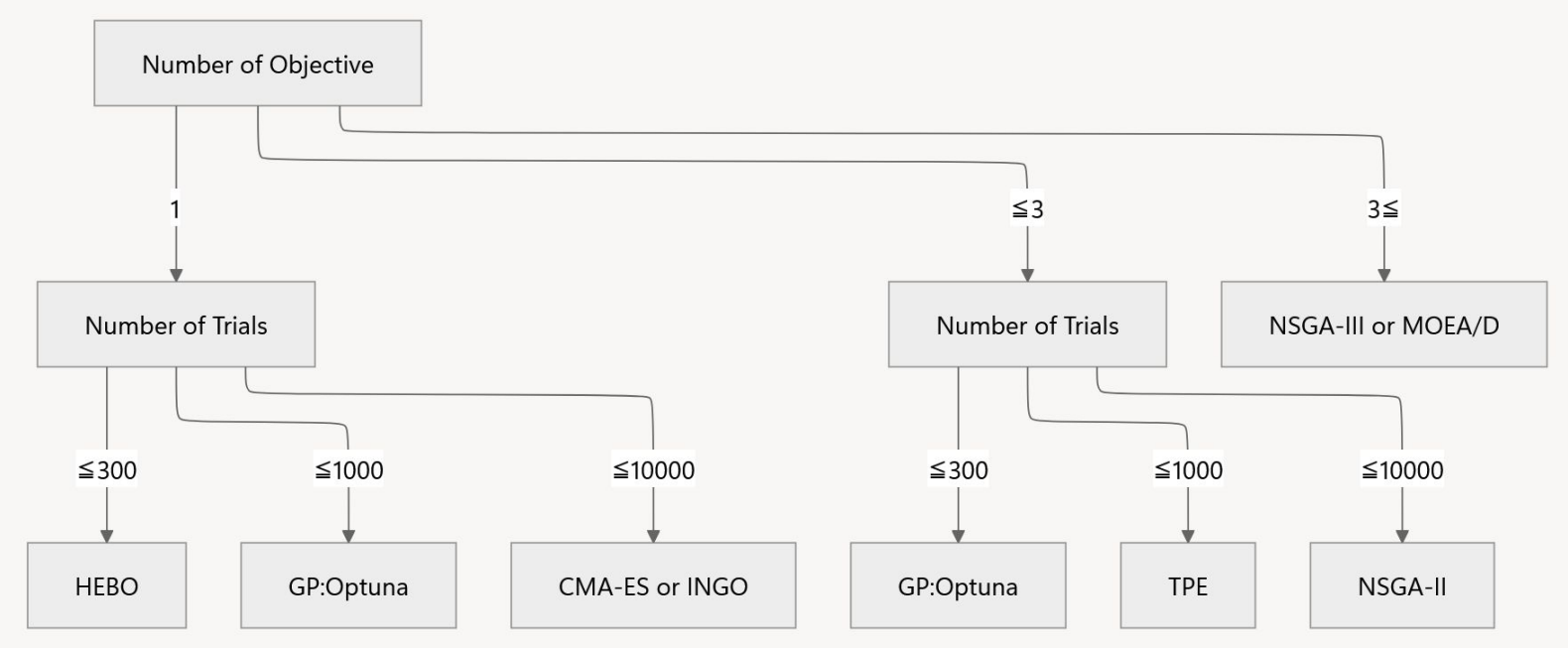


QMC

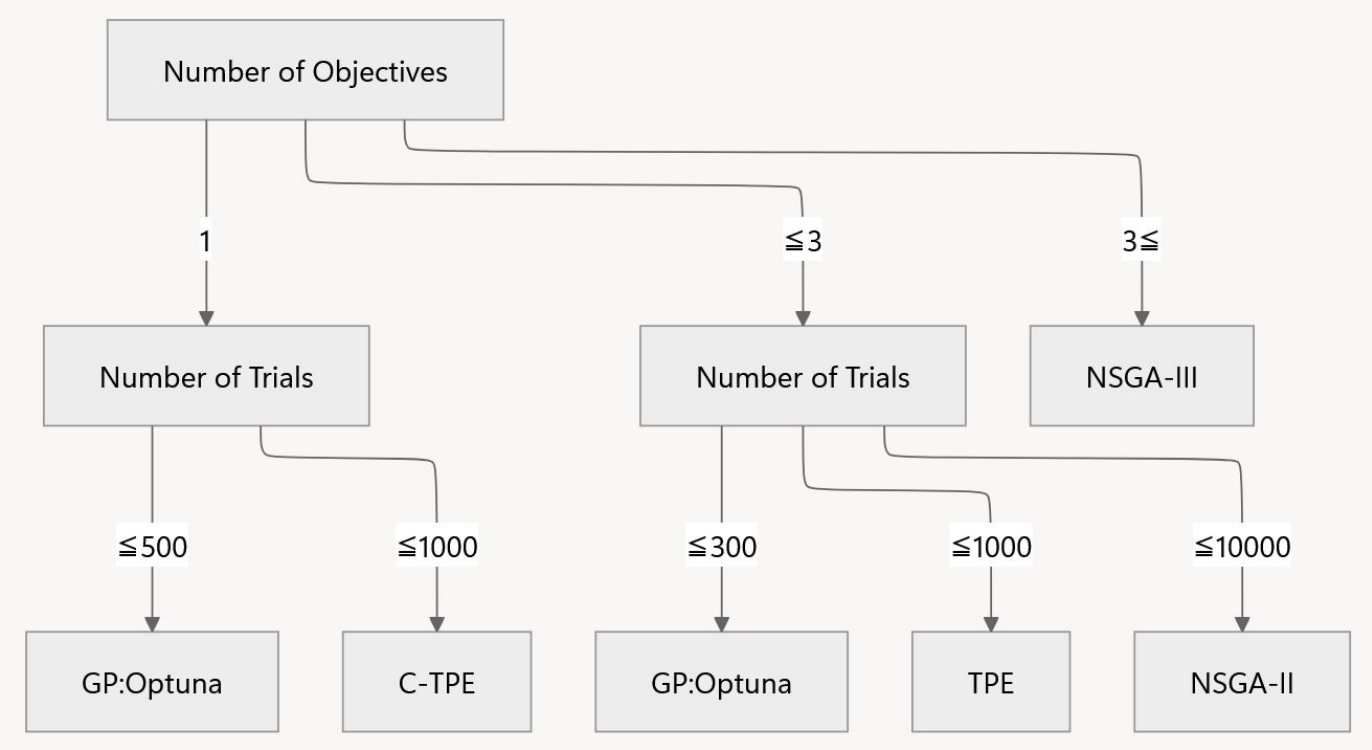
Optimization Method Selection Guide (First Try)



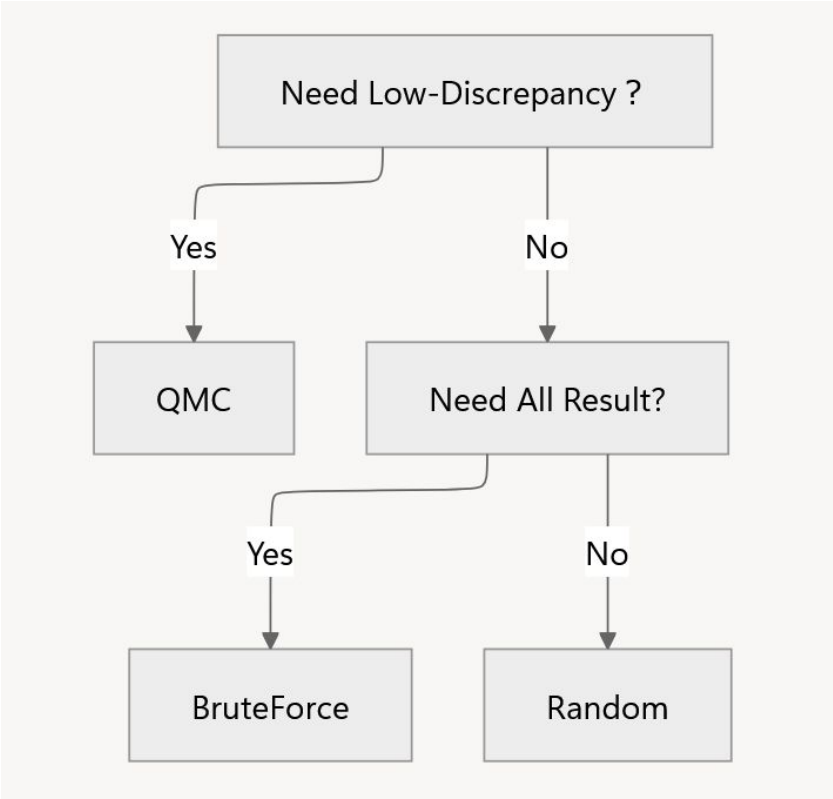
Optimization Method Selection Guide (Without Constraints)



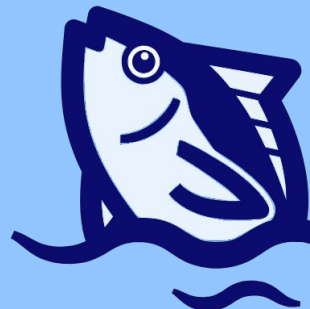
Optimization Method Selection Guide (With Constraints)



Design Space Exploration



Performance Benchmarks

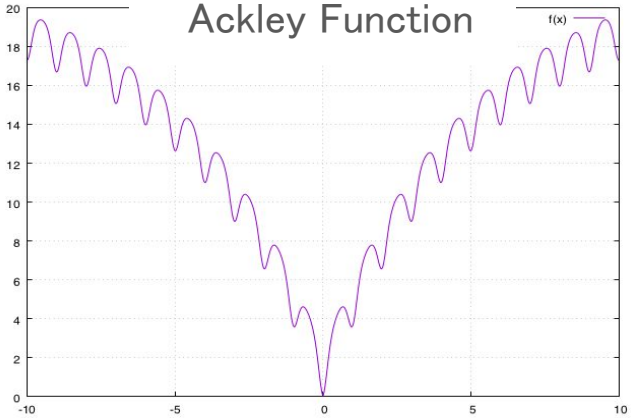


Comparison with Wallacei: Setup Verification

Wallacei is one of the most widely used optimization tools in Grasshopper.

Let's compare the results from each tool using their default settings.

We'll use the Ackley function as our test case.



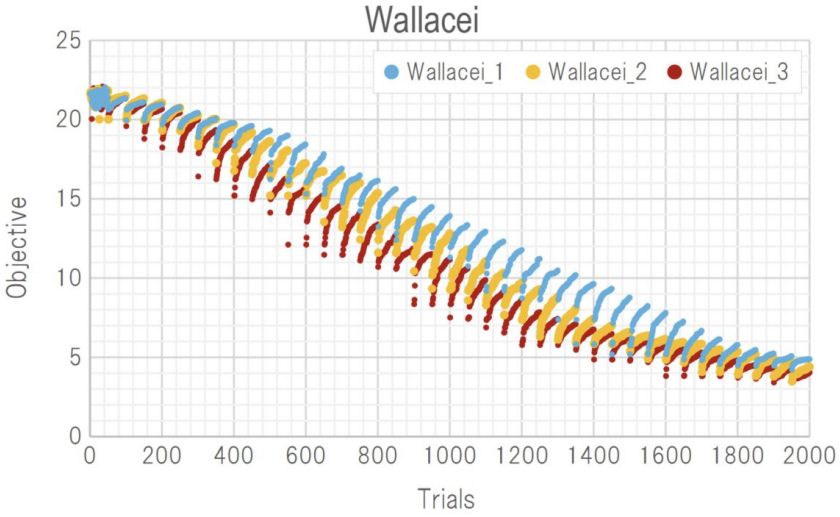
Wallacei uses the following optimization configuration:

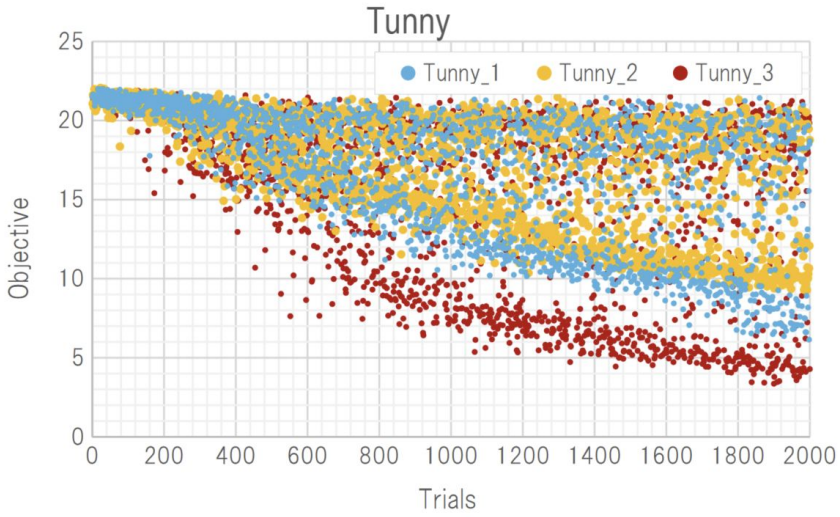
- Algorithm: NSGA-II
- Crossover: SBX (Simulated Binary Crossover)
- Mutation: Polynomial

Comparison with Wallacei: Results

Wallacei's default configuration focuses on intensively searching promising regions (exploitation).In contrast, Tunny's defaults balance thorough exploration of the entire design space with exploitation of good solutions.

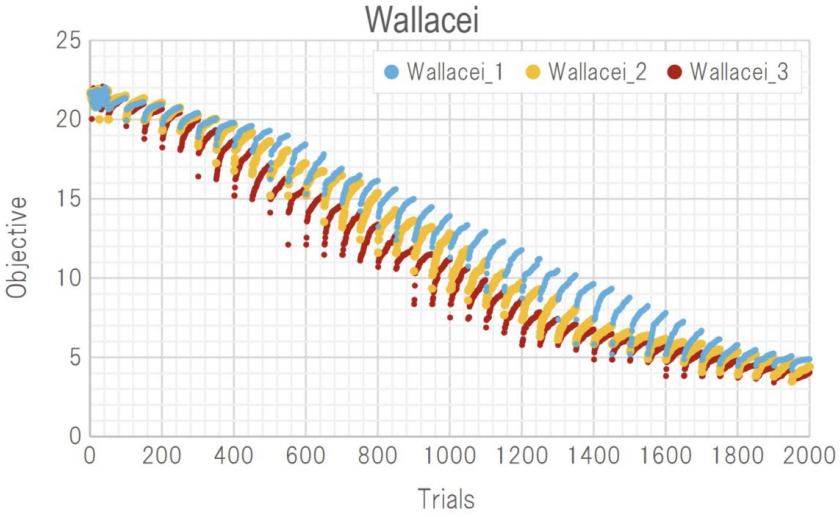
Even with the same underlying GA, these configuration differences lead to significantly different results.

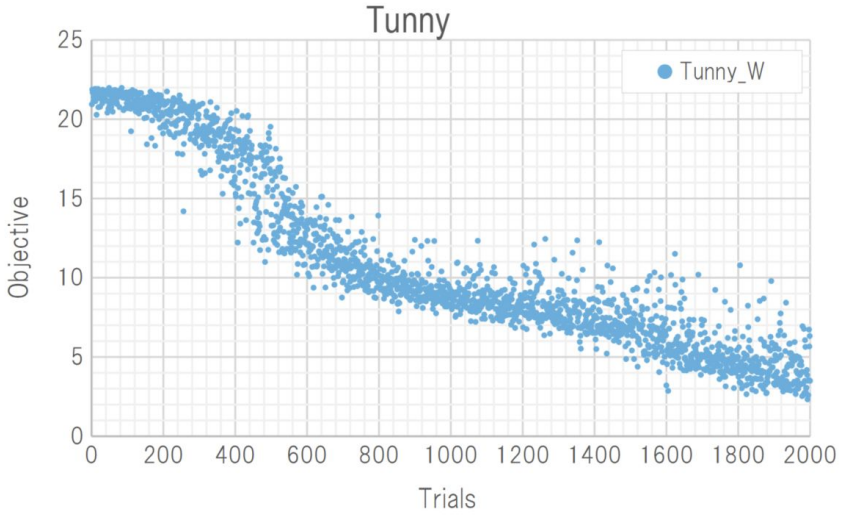




Comparison with Wallacei: Adjusting Settings

When Tunny's settings are adjusted to match Wallacei's configuration, both tools achieve comparable results.

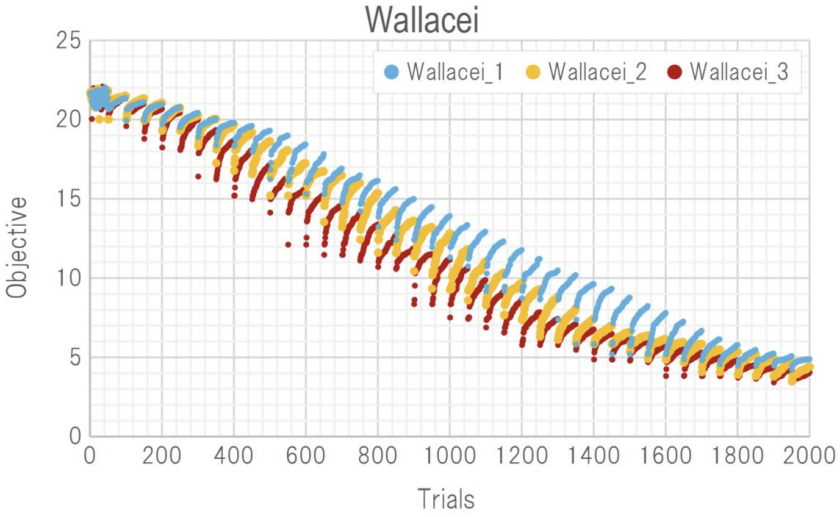


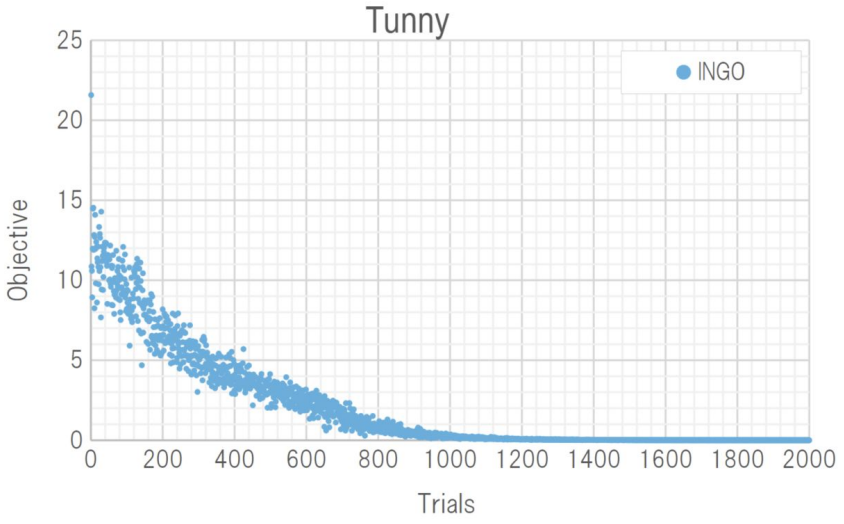


Comparison with Wallacei: Using Alternative Algorithms

Wallacei is limited to NSGA-II as its only algorithm.

In contrast, Tunny offers a wide selection of methods. For instance, using INGO (an advanced evolution strategy), Tunny converges to optimal solutions significantly faster than Wallacei.

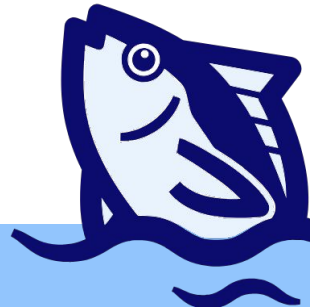




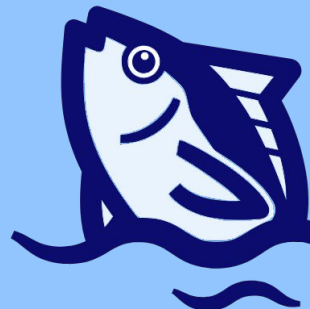
Comparison with Wallacei:Benefits of Multiple Algorithm Options

Key Takeaways:

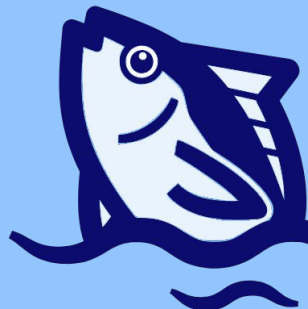
- ➔ **Verified Performance:** Both tools work correctly with the same algorithm and settings
- ➔ **Flexibility Matters:** Tunny offers multiple algorithms optimized for different problem types
- ➔ **Better Results, Faster:** Advanced algorithms like INGO can converge significantly faster than NSGA-II
- ➔ **But there's a tradeoff:** Choosing the right algorithm isn't always straightforward
- ➔ **Wallacei's approach is also valid:** Providing one robust, reliable algorithm simplifies the user experience



Hands-on Session



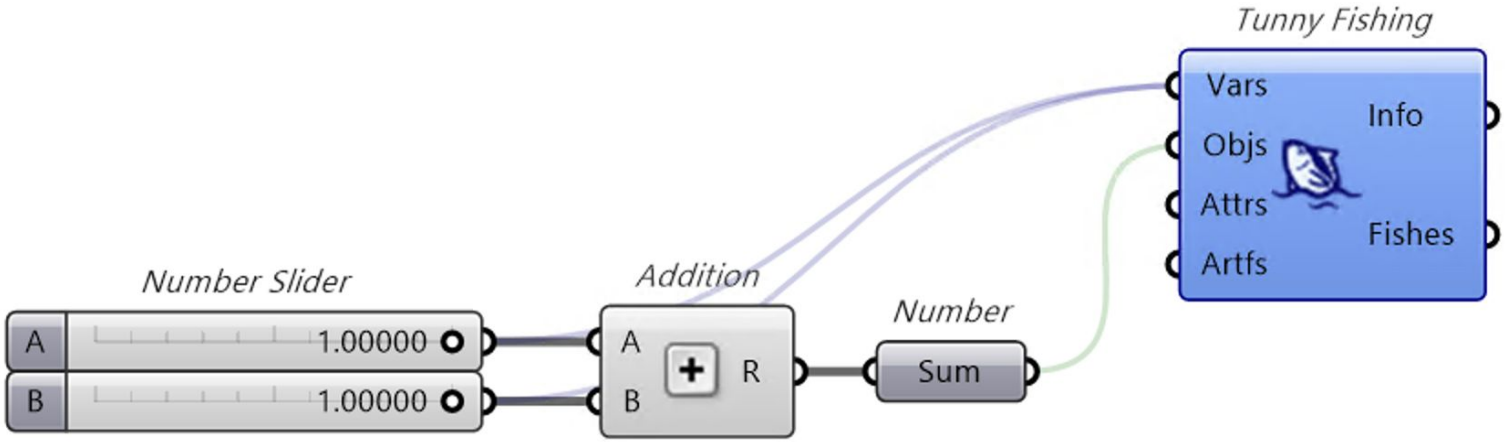
Getting Started with Tunny



Optimizing a Simple Problem: A+B

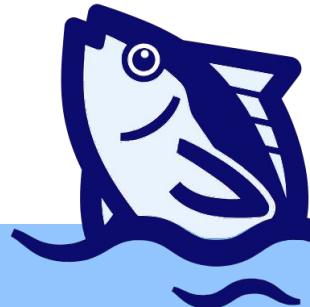
We'll start with a simple addition to see how Tunny works.

Use "01_addition.gh" from the provided files.



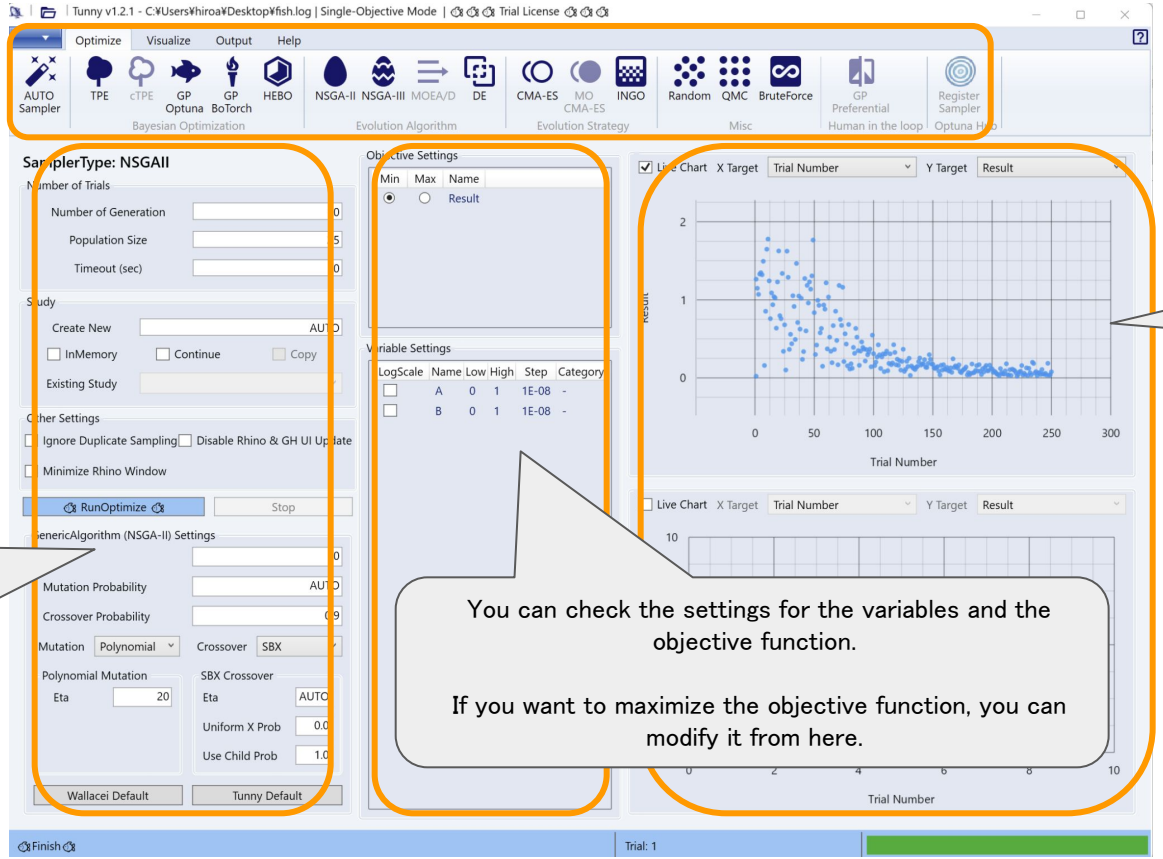
🐟Hands-On Practice🐟

- ➔ Basic Operations
- ➔ Tunny UI Overview
- ➔ Restoring Optimal Solutions (Reinstate Slider)
- ➔ Understanding Feature Importance
  - ◆ A+B doesn't mean equal importance (1:1)
  - ◆ Watch out for sampling bias
- ➔ Convergence Differences by Algorithm
  - ◆ Auto Sampler: Instant convergence
  - ◆ NSGA-II: Requires more time



Tunny UI Overview

This ribbon allows you to select the optimization method to use.



Optimization results are shown in real time.

Configure optimization settings.

Details will be introduced on the following slides and beyond.

You can check the settings for the variables and the objective function.

If you want to maximize the objective function, you can modify it from here.

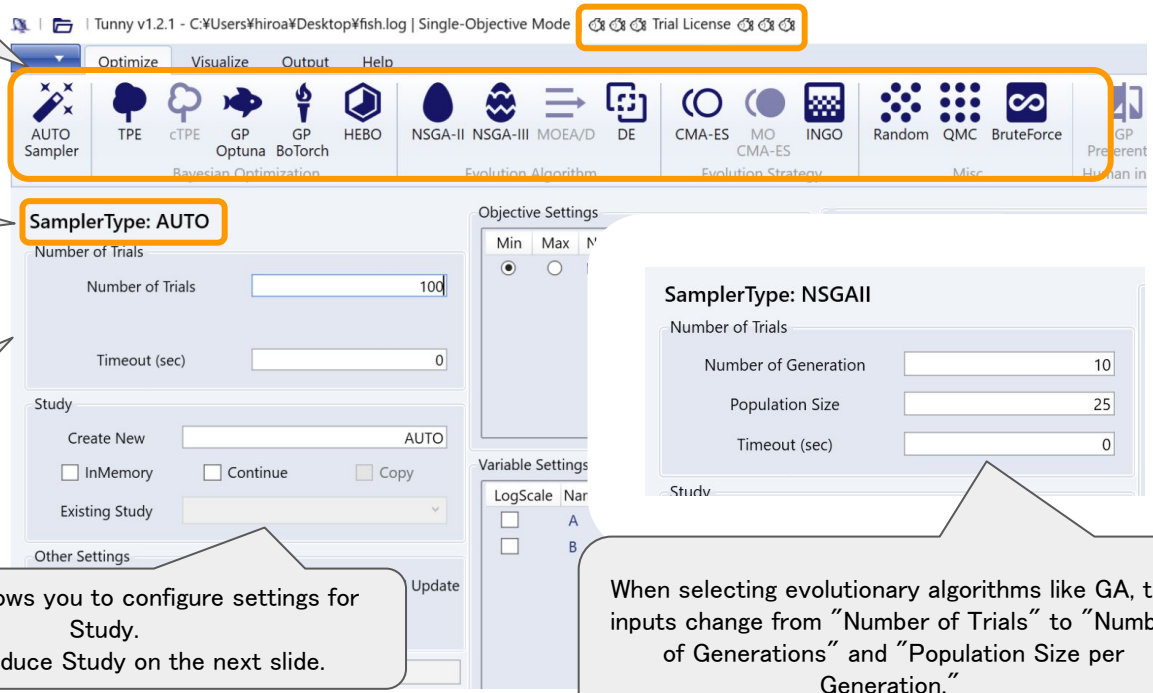
Tunny UI Overview : Ribbon & Trial

You can check whether you have obtained a license here.
If it displays "Community," you don't have one—please obtain one.

Only sampling methods available based on the number of objective functions and the presence of constraints will be enabled.

Displays the currently selected sampler. Choose AUTO.

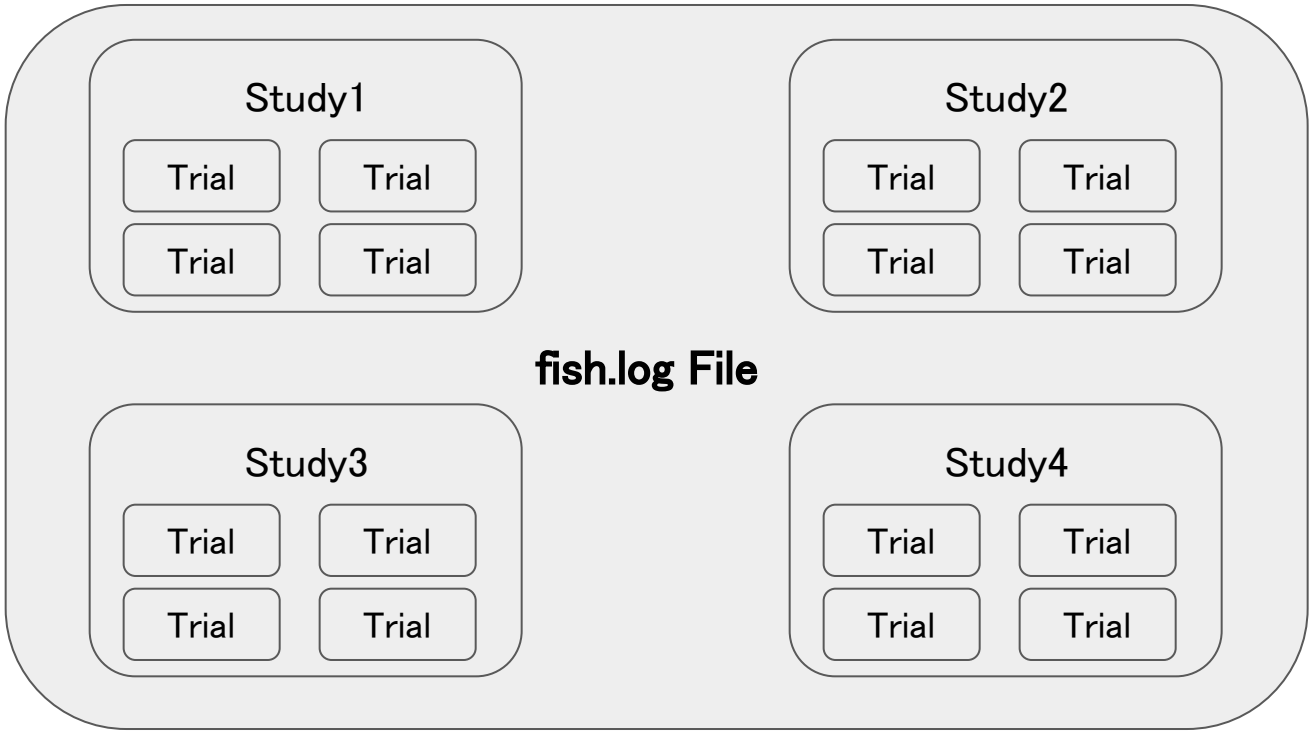
It provides inputs for determining both the Number of Trials and how long the optimization should run in seconds.



This field allows you to configure settings for Study.
I will introduce Study on the next slide.

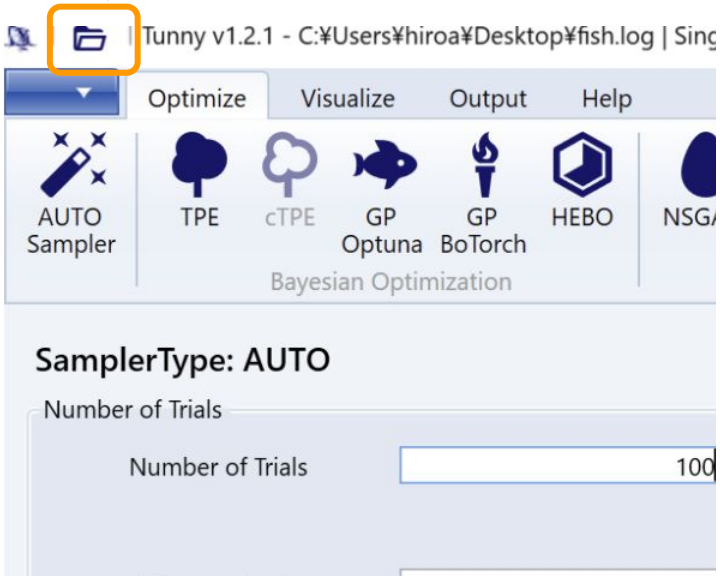
When selecting evolutionary algorithms like GA, the inputs change from "Number of Trials" to "Number of Generations" and "Population Size per Generation."

What is “Study” means : Result Save Format

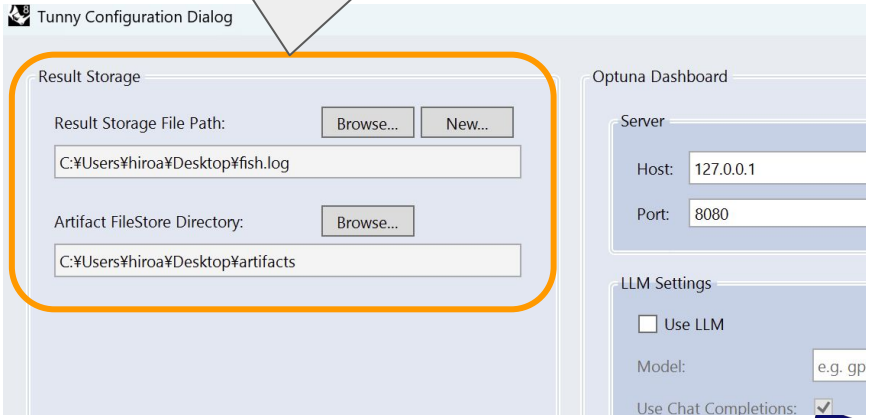


How to Change Result File

You can change the destination folder for result files from the folder icon at the top of Tunny UI.



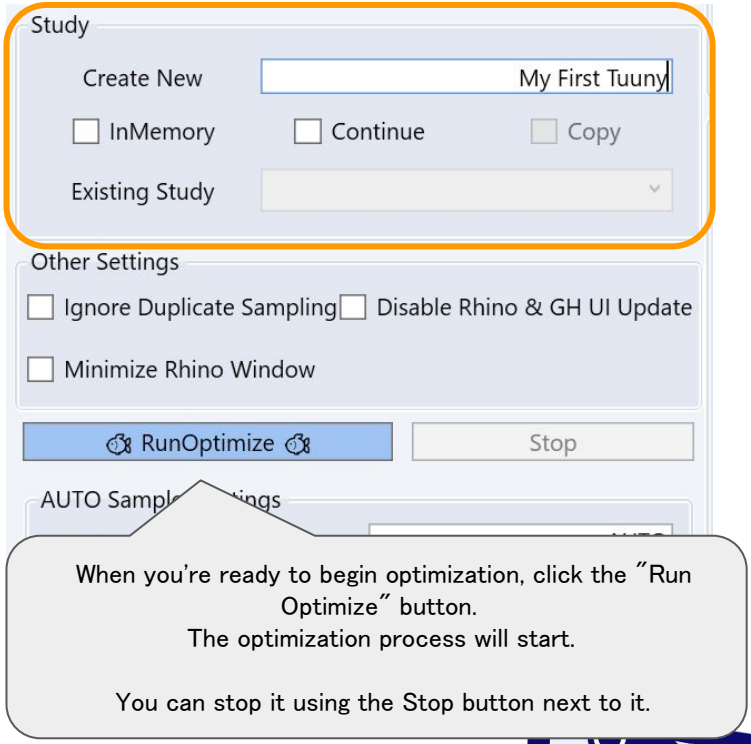
This is the window that appears when you click on a folder icon. From this window's "Result Storage" field, you can create new result files or open existing ones.



Tunny UI Overview : Study Settings

The ″Study″ field allows you to configure how optimization results are handled.

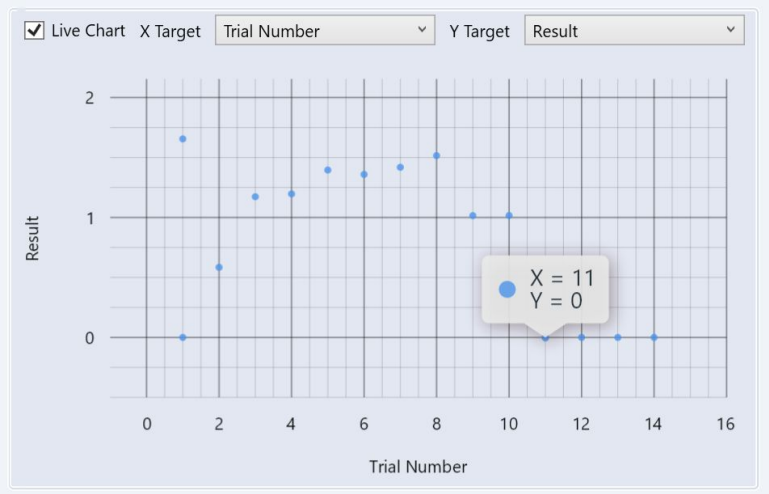
- ➔ In the ″Create New″ section, specify the name of the new study to be created.
  - ◆ Since this is our first optimization, let's call it ″**My First Tunny**″
  - ◆ Entering ″AUTO″ will automatically assign an ID.
- ➔ Checking ″Continue″ allows you to restart an existing study.
  - ◆ Select the study you wish to restart from the ″Existing Study″ combo box.
- ➔ Checking ″InMemory″ improves optimization speed by preventing results from being saved to files during the intermediate stages.
  - ◆ Note that results won't be saved if Rhino crashes or encounters other issues



Restoring Optimal Solutions (Reinstate Slider)

After optimization is complete, let's check the results.
The AUTO Sampler is extremely powerful—it may reach the optimal value of 0 before completing the set number of trials, so be sure to stop it when appropriate.

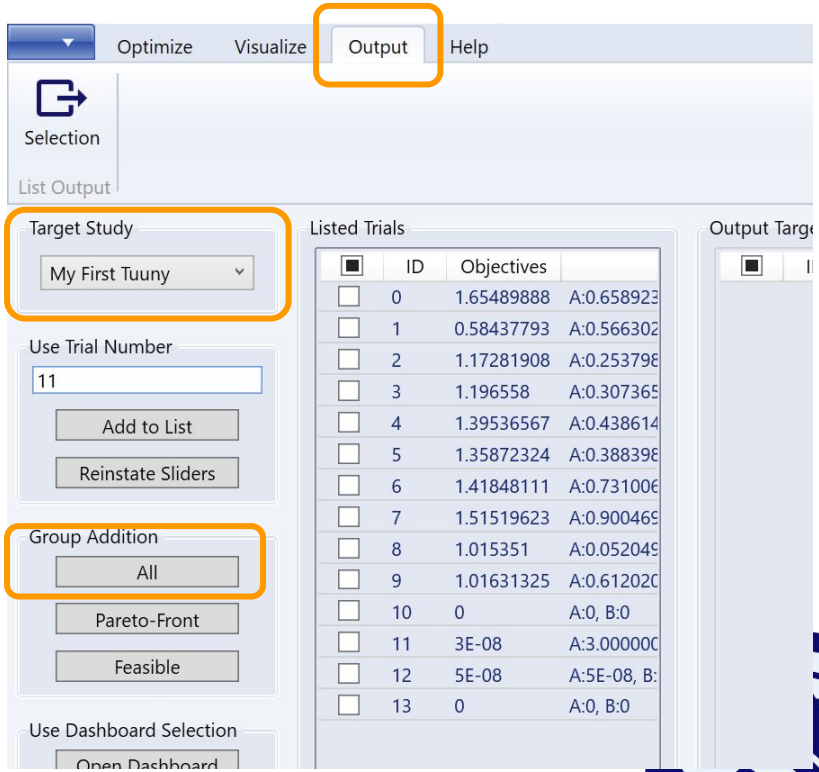
In the example below, the value reached the minimum of 0 at the 11th trial.



Restoring Optimal Solutions (Reinstate Slider)

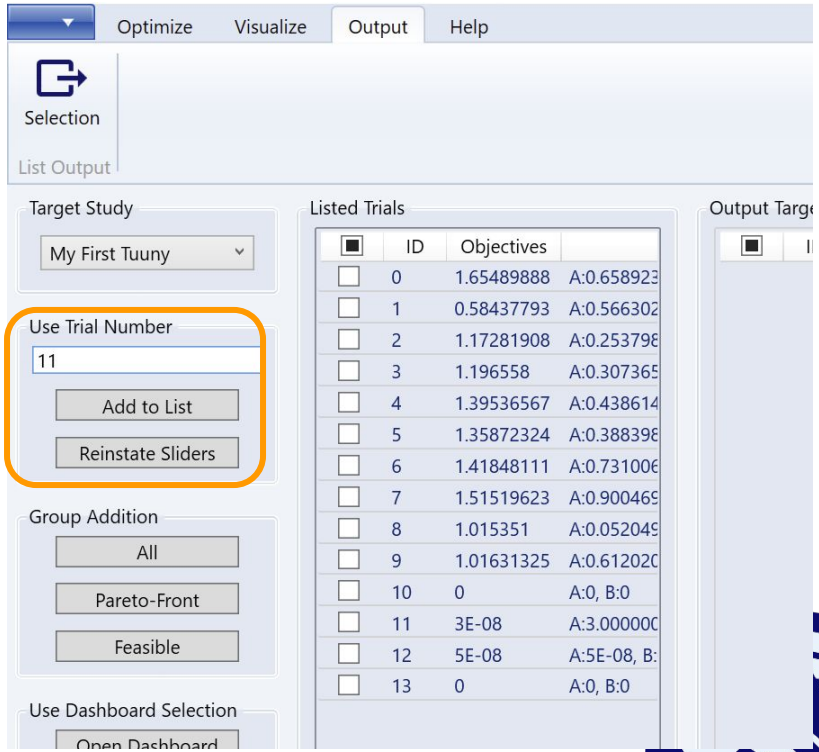
To view the results in Tunny UI, select the "Selection" option in the "Output tab".

Ensure that the Target Study is set to "My First Tunny" and select "All" for "Group Addition" to display all trials in the Listed Trials table.



Restoring Optimal Solutions (Reinstate Slider)

To apply specific results to a Grasshopper Number Slider, enter the ID of the result you want to apply in the "Use Trial Number" field, then click the "Reinstate Sliders" button.



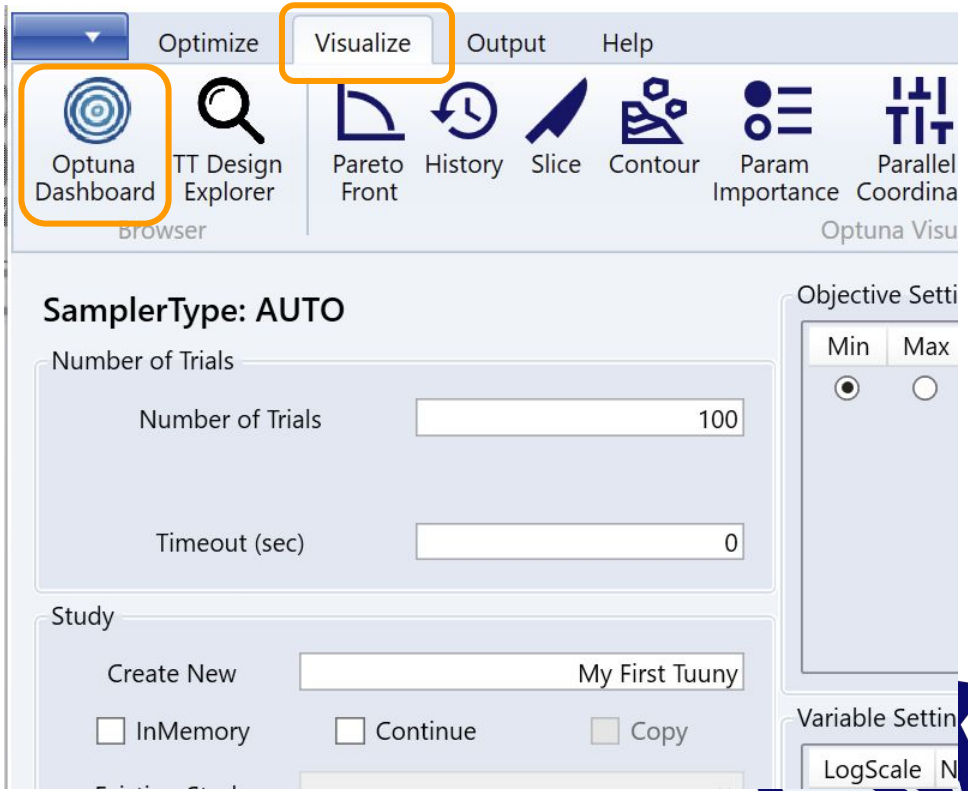
Checking Results in Optuna Dashboard

For more detailed result analysis, use the "Optuna Dashboard."

You can launch the Optuna Dashboard by clicking the "Optuna Dashboard button" in the "Visualize tab."

Clicking the button will launch your browser where you can view the results.

If the results don't appear immediately, it means the launch process is taking time – please refresh the page after a short wait.

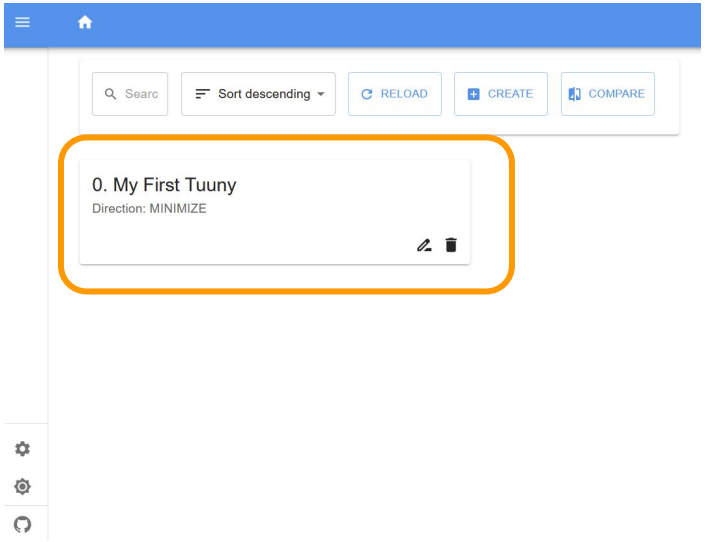


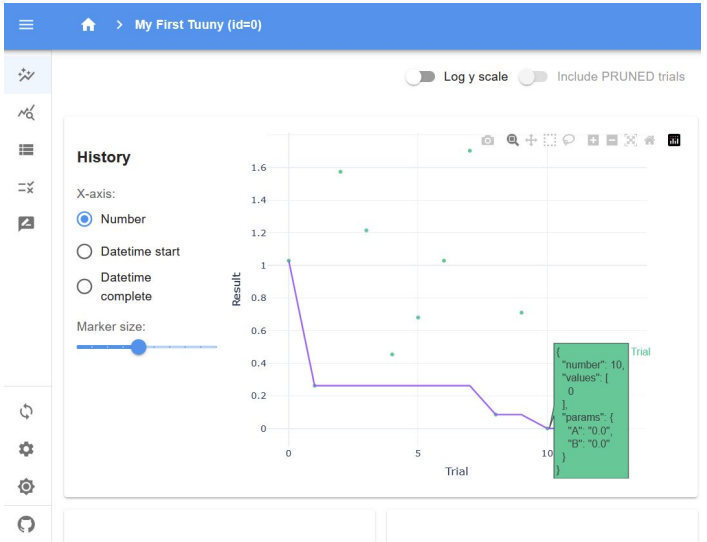
Checking Results in Optuna Dashboard

The dashboard's main screen displays all your created studies. For this example, let's select "My First Tunny."

When selected, you can view the details of a Study. You can view various information graphs about the Study on this dashboard.

On the first page where you select Study, you can review historical data about how the objective function values changed during optimization



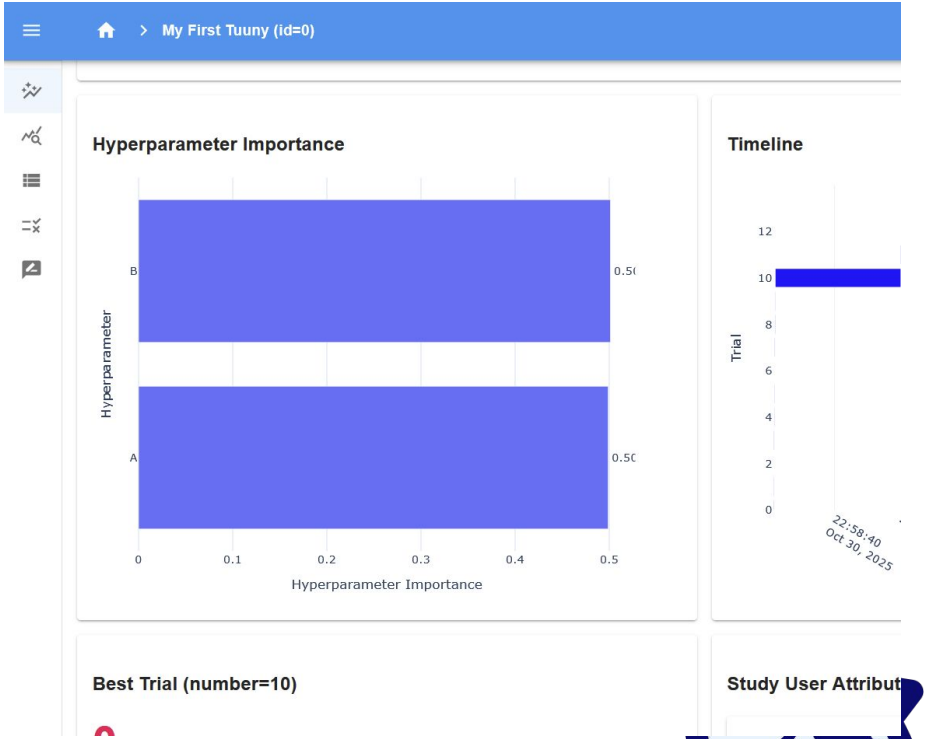


Understanding Feature Importance

One particularly useful feature is "Hyperparameter Importance," which visualizes how much each variable (hyperparameter) affects the objective function.

In our problem, we're optimizing A+B, and both variables contribute equally to the objective function.

Examining the graph reveals that variables A and B are both represented by bar charts showing 0.5, clearly demonstrating that each contributes 0.5 to the objective function value.



Convergence Differences by Algorithm

This is a very simple problem of A + B, but the number of trials required for convergence can vary dramatically depending on the optimization algorithm.

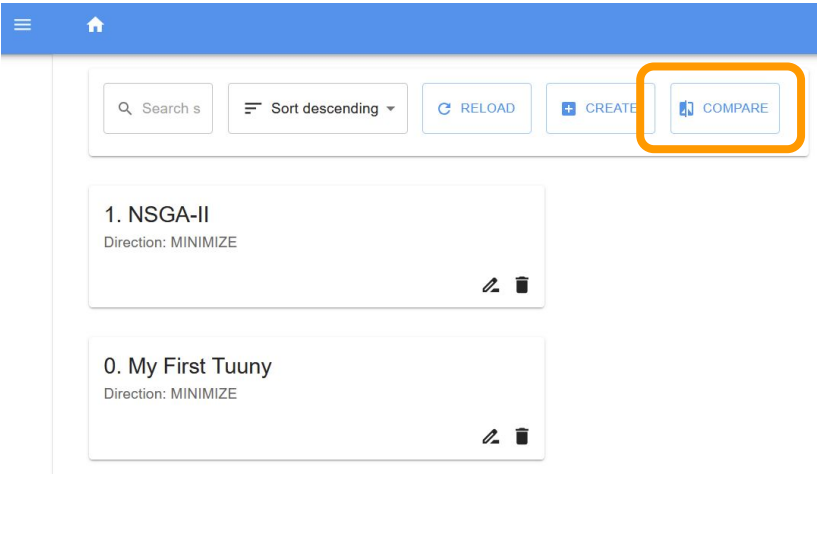
I'll allocate about 15 minutes of time, so please experiment with how different methods affect convergence using your Grasshopper environment.

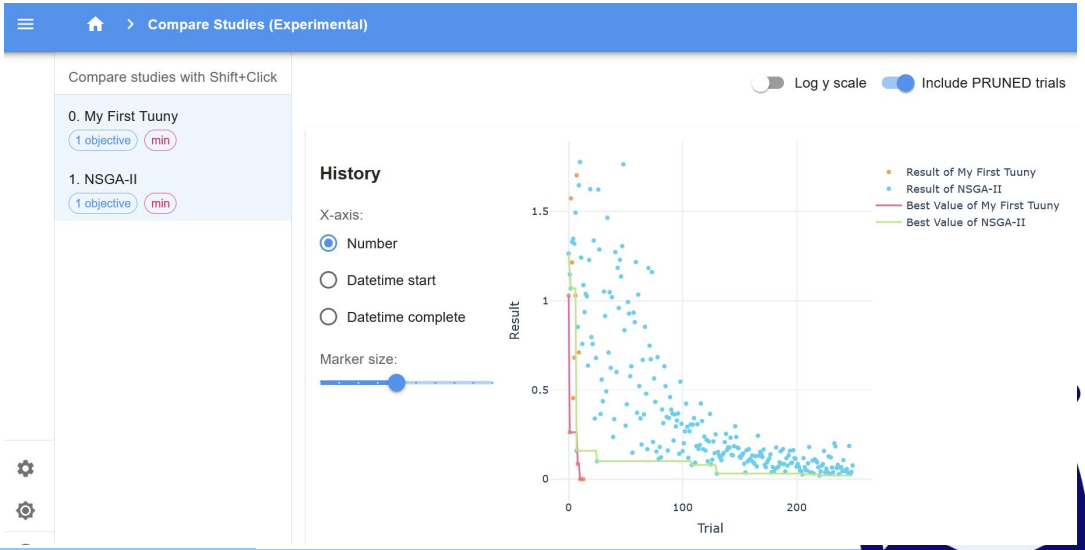
Since this is single-objective optimization, you can test 13 different methods. To facilitate comparison later via the dashboard, it's good practice to name your Study after its method.

Compare Optimization Results

Using Dashboard is also convenient when comparing different methods.

On the top-right corner of the dashboard's main page, you'll find the " COMPARE" button. Clicking this button will take you to the comparison page.
By holding down the "Shift" key while clicking on Study names on the left side, you can include multiple results in the single graph.





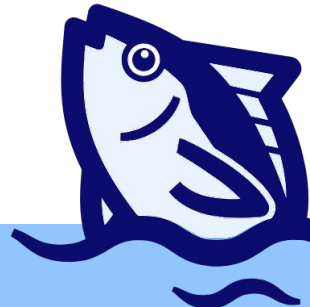
Practice Time

To reiterate, even an extremely simple problem like A+B can vary significantly depending on the optimization method.

Please experiment with different approaches yourself to truly appreciate the need for multiple methods.

Keep in mind that the most appropriate method varies by problem, so these results aren't necessarily applicable to all cases.

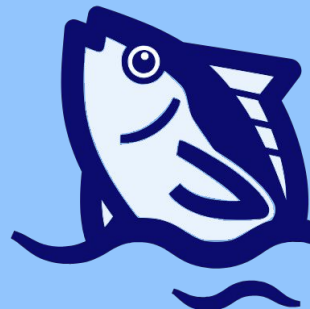
Take about 15 minutes to optimize on your own and experiment with how different approaches affect each method.



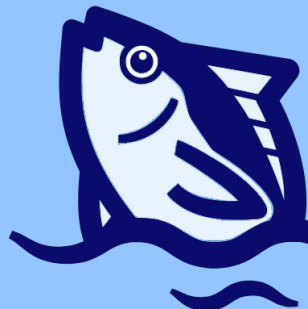
Practice Time

15-minute

12:00(EST) start next session



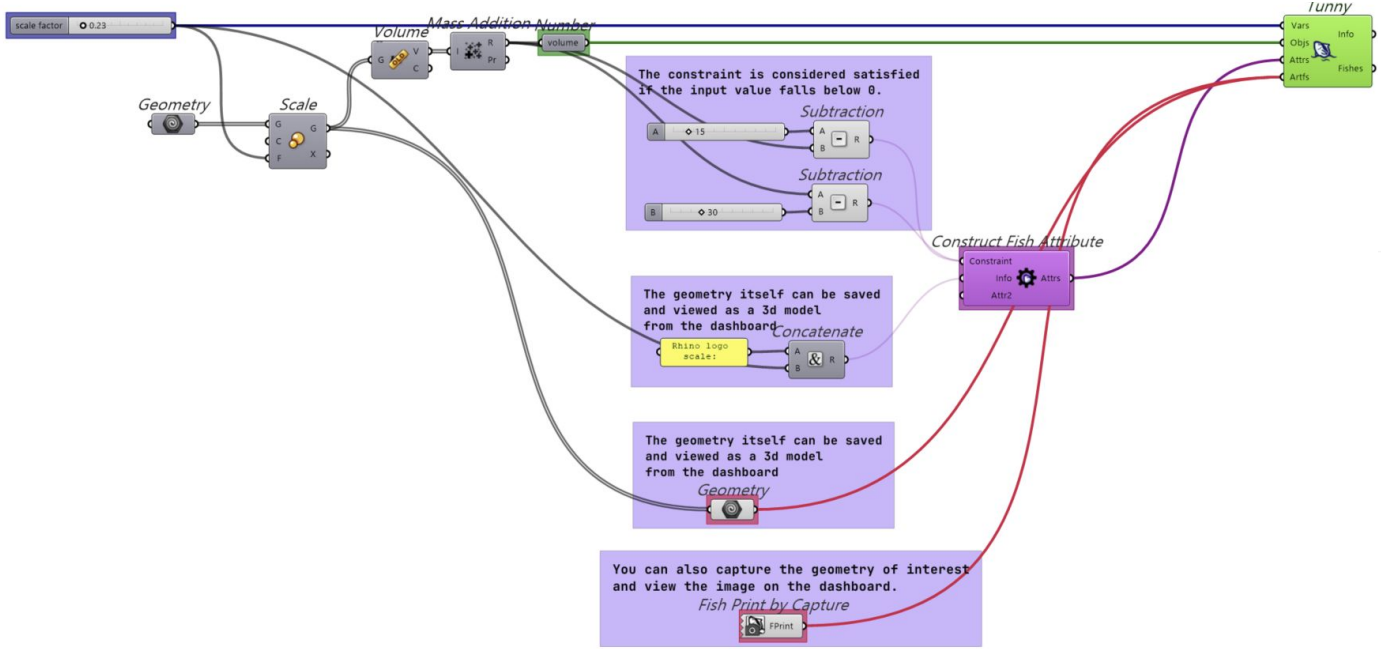
Exploring Attributes, Artifacts at Dashboard

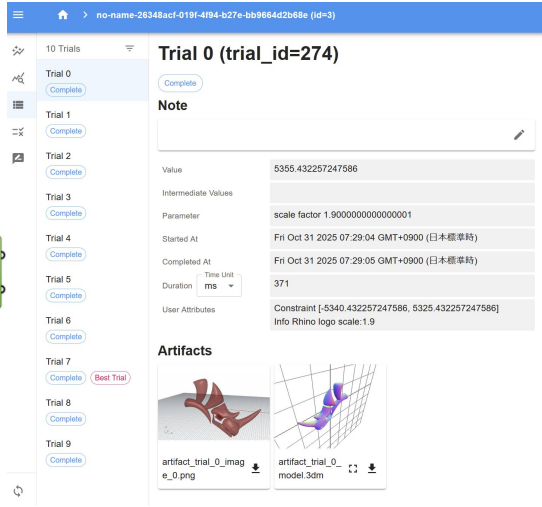


Exploring Attributes, Artifacts at Dashboard

Let's explore some of Tunny's distinctive features: constraints, Attributes, Artifacts.

Open the file "02_attr_and_artf.gh" from the workshop materials.





What is "Soft Constraint"

In Tunny, constraints are implemented as **"soft constraints"** rather than hard constraints.

This means that solutions violating the constraints can still appear in the optimization process—this is intentional, not a bug or misconfiguration.

Why soft constraints?

- ➔ The optimizer learns from constraint violations to understand the boundaries of feasible regions
- ➔ Exploring infeasible solutions helps the algorithm find better feasible solutions nearby
- ➔ Hard constraints can make optimization more difficult or even impossible in some cases

What you'll see:

Don't be alarmed if you see solutions that violate your constraints during optimization. The algorithm is actively learning where the feasible region is, and over time, it will converge toward constraint-satisfying solutions.

🐟Hands-On Practice🐟

Dashboard Basics

➔ Click on any data point in the History or Pareto Front plots to jump directly to that trial

Working with Attributes and Artifacts

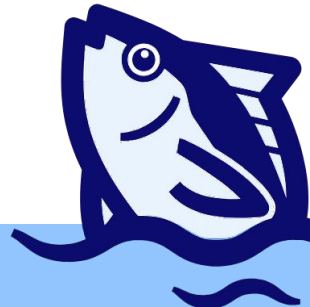
➔ Artifacts: Accept any file type via file path
➔ Attributes: Automatically visualized in Dashboard charts—add any values you want to track

3D Model Best Practices

➔ Convert geometry to mesh format before adding to Artifacts for optimal performance

Trial Smart Selection

➔ Compare Artifacts side-by-side
➔ View multiple 3D models simultaneously



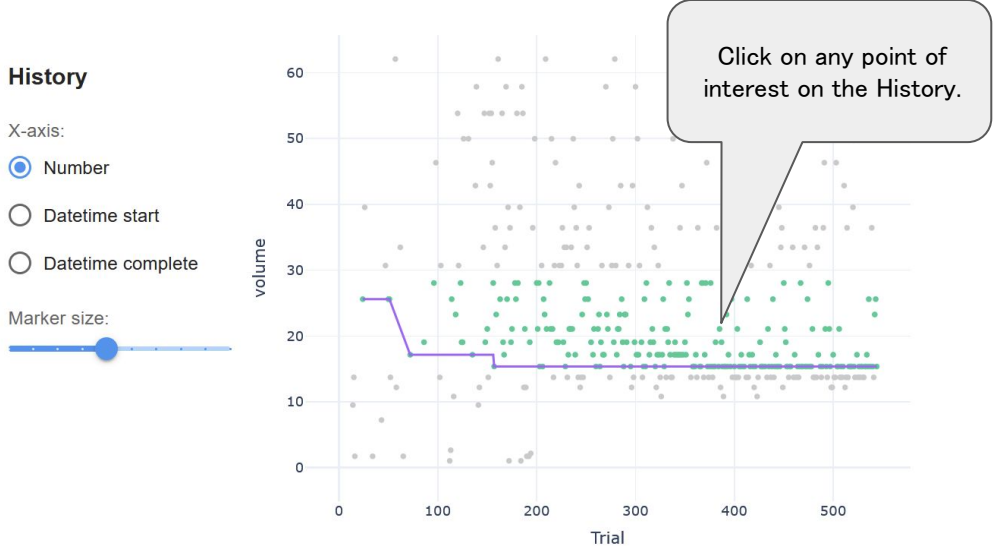
Dashboard Basics

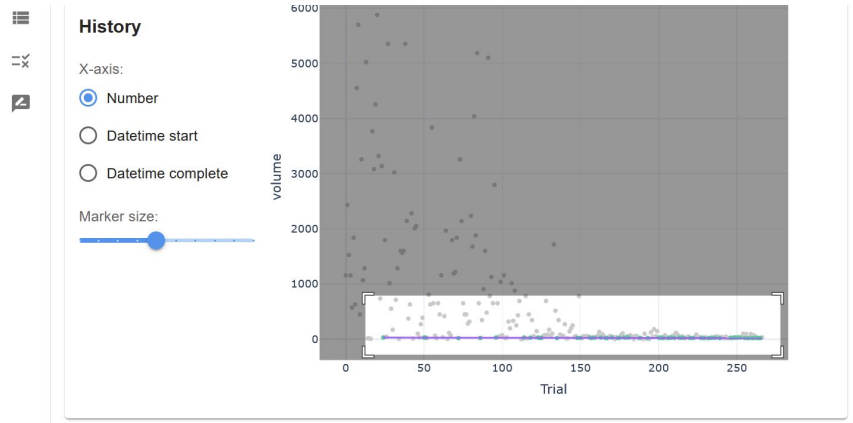
Many plots with gray colors represent infeasible solutions that do not satisfy the constraints.

The Dashboard is highly interactive.
All charts can be manipulated using the mouse.

For instance, with the History plot, you can:

- ➔ Adjust the display range by dragging and dropping with the mouse
- ➔ Directly set the range by double-clicking on axis endpoints
- ➔ Click on any point of interest to jump to its detailed view
- ➔ Shows point details when hovered over





Working with Attributes and Artifacts

The trial details page appears as follows:

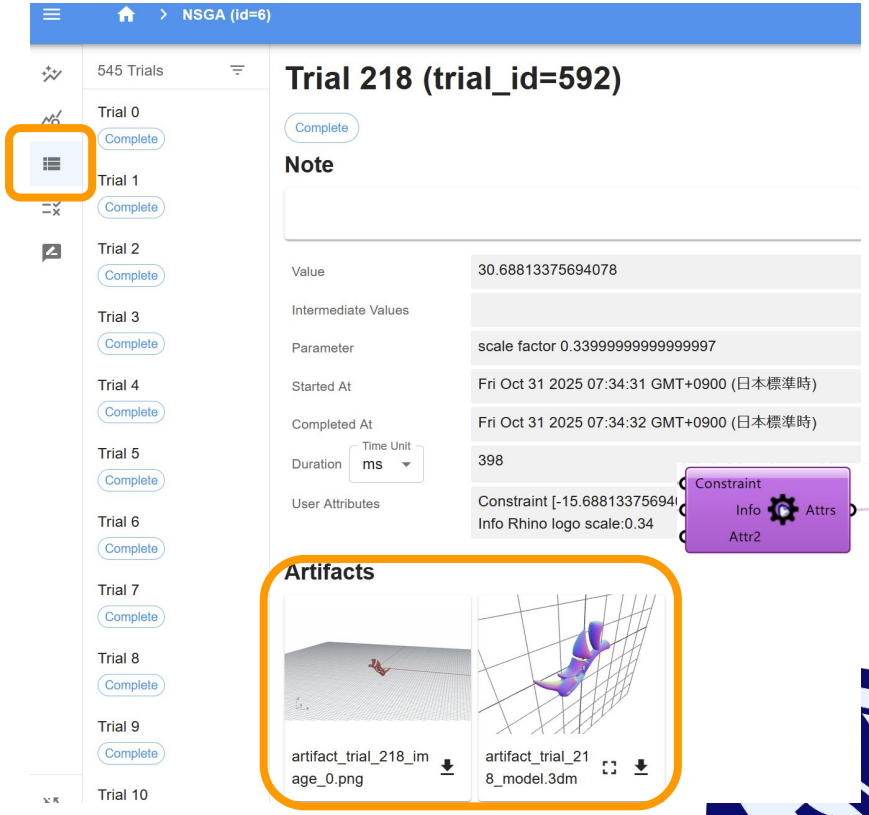
You can view this information not only by clicking on plot points but also by pressing the ″ Trials (List)″ button on the left side of the screen.

This display shows all relevant information about the trial.

The ″Attributes″ section displays the values set for the Fish Attribute in Grasshopper.

The ″Artifacts″ section can display various file information.
In addition to images, when you input a Brep or mesh into Artf, you can view and manipulate the object in 3D space directly in the browser.

you can associate any file such as a CSV file

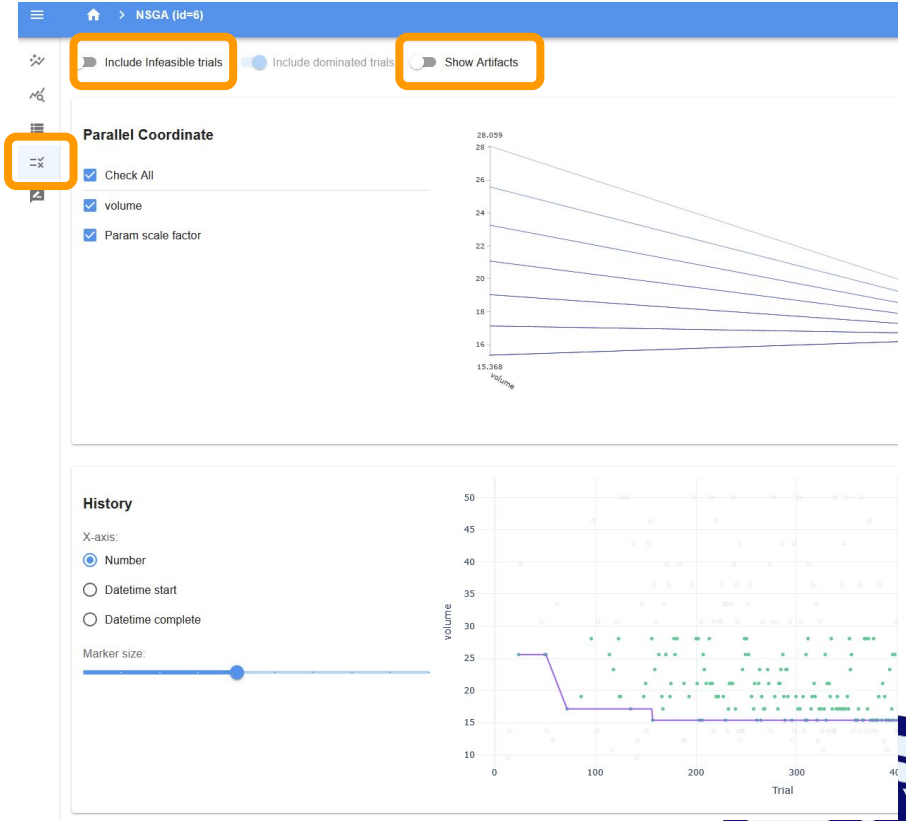


Trial Smart Selection (Top)

For selecting trials, use the Smart Selection method. Below I explain how selection works.

At the top of the page, you'll find options for specifying the target selection criteria.

1. The first option is whether to restrict selection to only feasible solutions that satisfy all constraints.
2. The second option is whether to display dominated individuals during multi-objective optimization.
3. The third option is whether to display artifacts registered through the Artifact feature mentioned earlier.

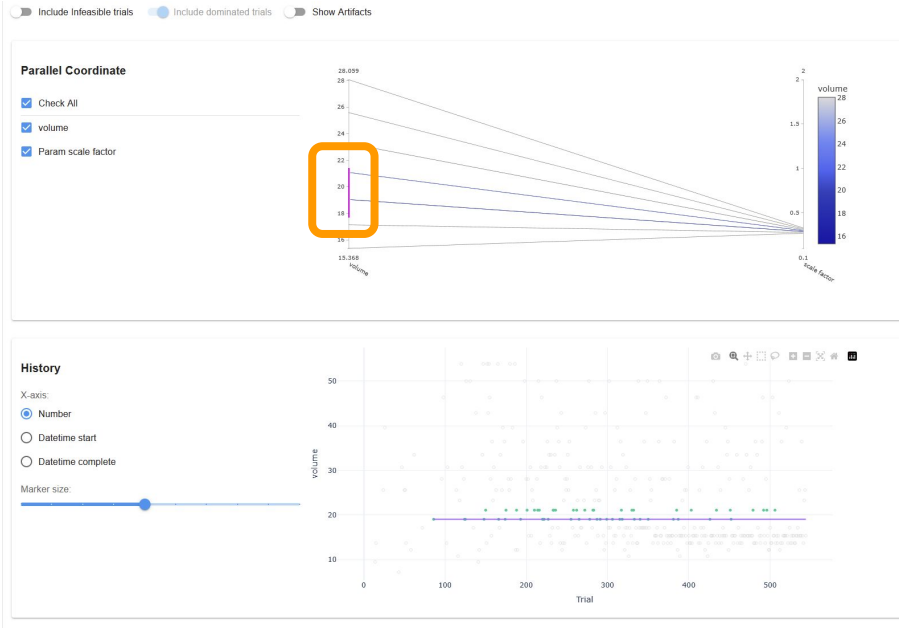


Trial Smart Selection (Filtering)

On the top Parallel Coordinate Plot, you can filter which trials to display on the page.

Selection can be made by dragging along the axis.

In History, for example, selected trials appear as colored points, while unselected trials appear as unfilled circles.



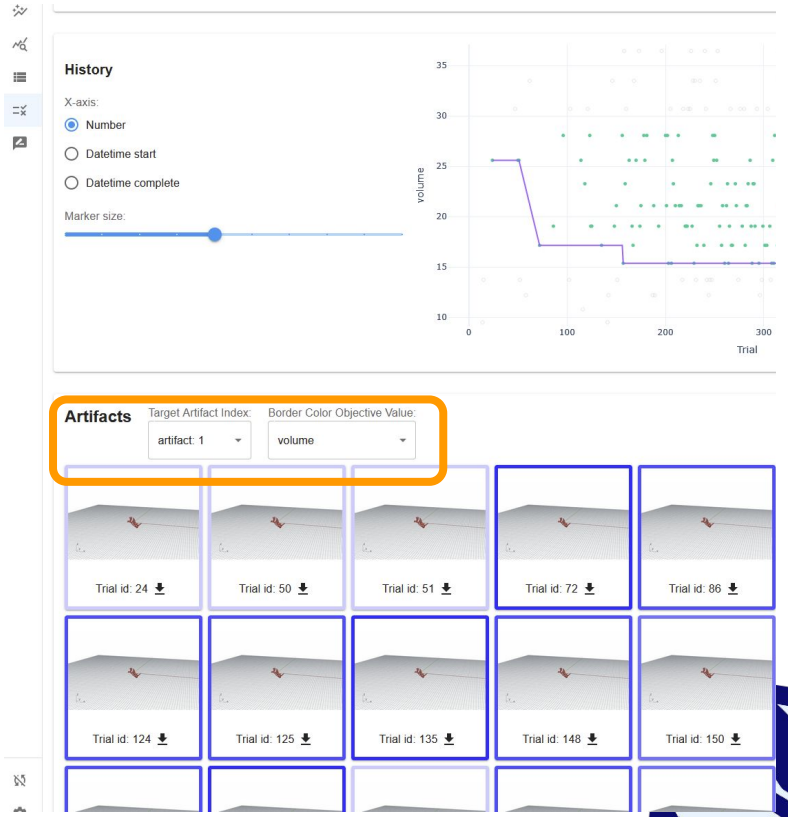
Trial Smart Selection (Artifact)

When Artifacts are enabled, you can display a list of artifacts at the bottom of the page.

Only selected items are displayed

The border color indicates the degree of objective function evaluation—darker colors represent better objective function values. For minimization problems, this indicates smaller values.

When there are multiple artifacts or objective functions, you can select which one to choose from the dropdown list.



Trial Smart Selection (Download CSV)

At the very bottom of the page, you'll find a table containing detailed information about your selected trial.

Here you can view all detailed values.

If you would like to perform more in-depth processing, you can download this table in CSV format.

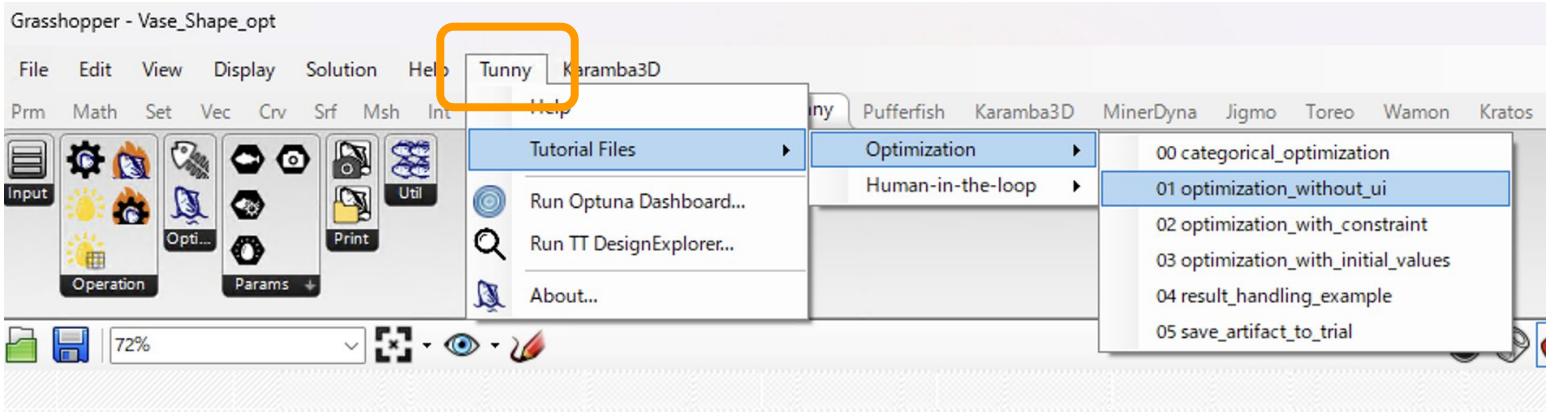
To download it, please click the "DOWNLOAD CSV FILE" button.

Using the Toolbar

Next, let's examine the functions available in the toolbar at the top of the Grasshopper interface.

Here, you can access help documentation and tutorial files.

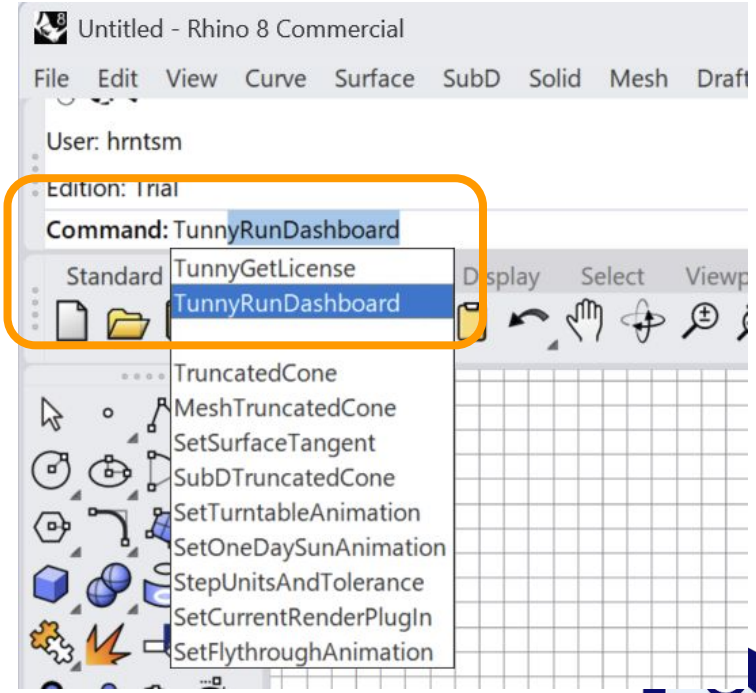
You can also launch browser-based optimization analysis tools such as the Dashboard and Design Explorer from this location.



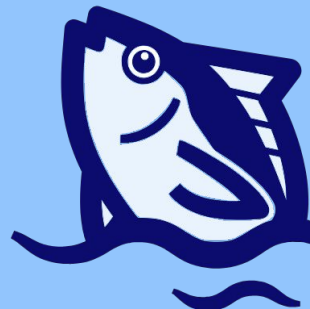
Open Dashboard in Rhino

When you're just looking to review results, constantly waiting for Grasshopper to launch can be somewhat tedious.

Tunny allows you to run the Dashboard in Rhino by executing the "**TunnyRunDashboard**" command, eliminating the need to launch Grasshopper itself.



LLM Supported Result Analysis



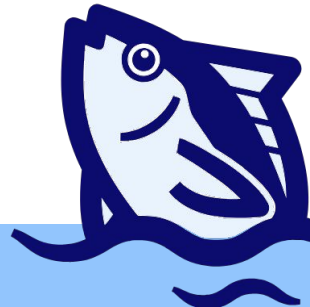
LLM-Based Optimization Analysis

Tunny supports **Optuna Dashboard's LLM integration** , enabling you to analyze optimization results using natural language.

Key Capabilities:

- ➔ **Natural language queries** : Ask questions about your optimization in plain English
- ➔ **Intelligent data filtering** : Control Dashboard views through conversational AI
- ➔ **Interactive analysis** : Get insights without writing complex queries or code

This integration brings the **power of AI-assisted analysis** directly to your design optimization workflow, making it easier to extract meaningful insights from your results.



Handling LLM Integration in Tunny

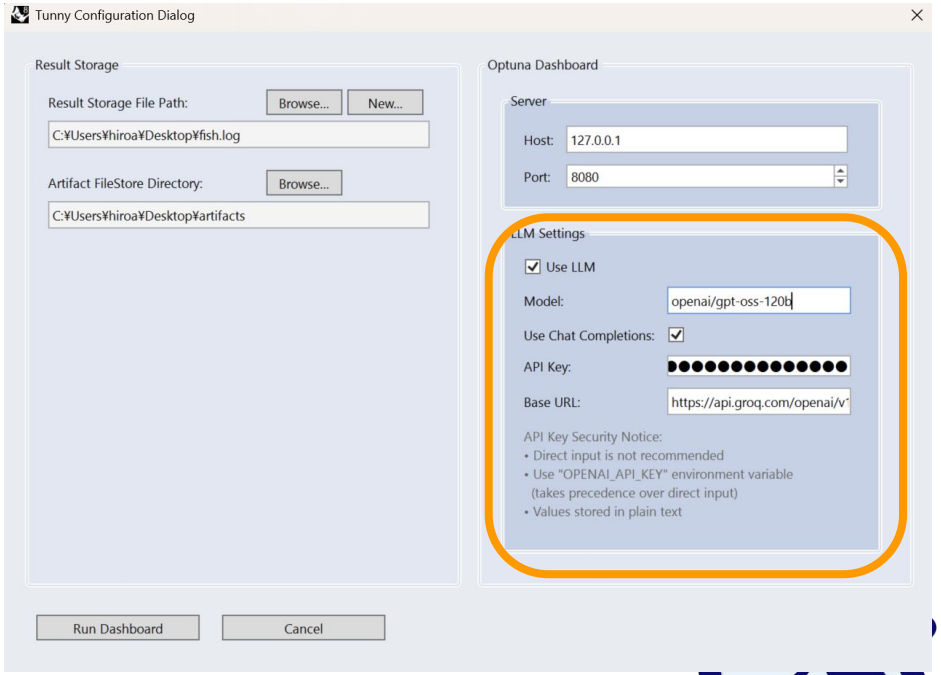
This is a form for configuring the storage location for results after introducing Study.

On the right side of this form, you'll find settings for integrating with LLMs.

Enter the desired LLM model, API Key, and API URL here, then launch Optuna Dashboard to enable the LLM integration functionality.

For detailed information about this feature, please refer to the documentation page for Optuna Dashboard.

[Tutorial: LLM Integration]



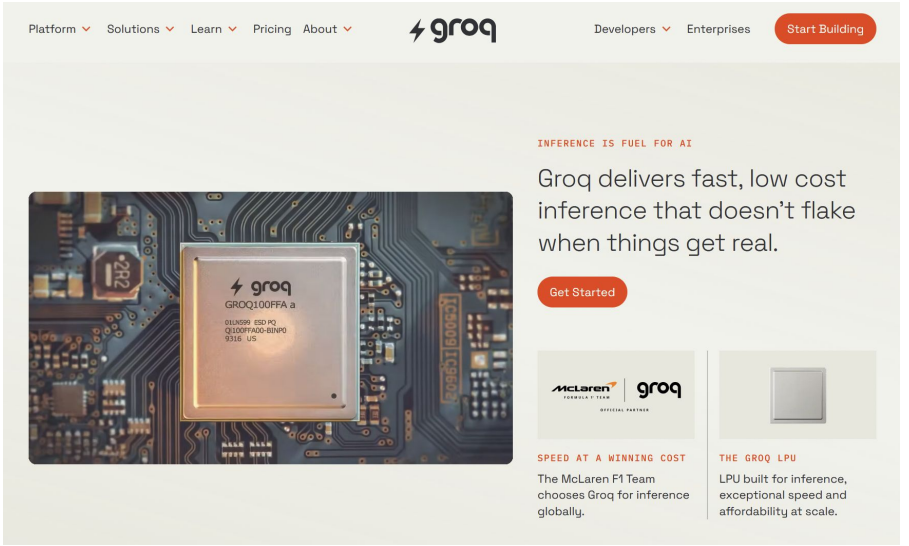
OpenAI Compatible API Services

If you're interested in trying this LLM integration feature, I recommend using the **"groq"** service.

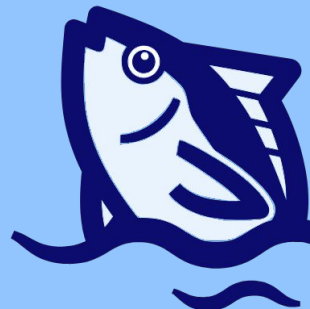
Why groq?

- ➔ **Free to use** : While OpenAI's API services are paid, groq currently offers a fully compatible service for free
- ➔ **Fast and affordable** : groq provides very fast responses at an extremely affordable price
- ➔ **Easy to start** : Simply register an account and obtain an API Key

Recommended model : Use "openai/gpt-oss-120b" or a model with higher performance. Models with lower capabilities often make analysis errors.



Exploring Different Objectives and Algorithms



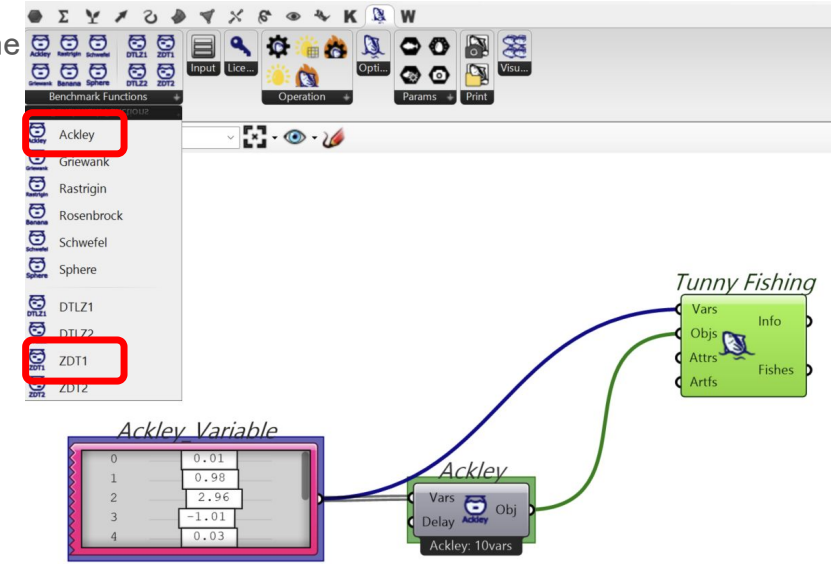
How Objectives and Algorithms Differ

Let's experiment with Tunny's various optimization algorithms using the Ackley function—the same benchmark we used when comparing with wallacei earlier.

In the earlier Addition example, Auto Sampler (GP) converged very quickly. Will it perform as well on this more challenging problem?

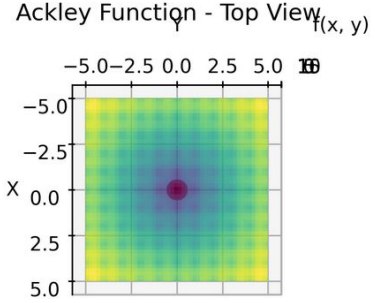
For advanced users: Try the multi-objective ZDT1 function to see how different algorithms handle multiple objectives.

Pro tip: Use Dashboard's Compare feature to visualize performance differences between algorithms.

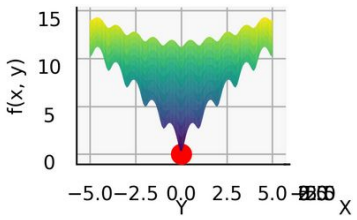


Ackley Function

Ackley Function - Top View

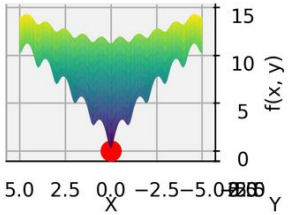


Ackley Function - Front View

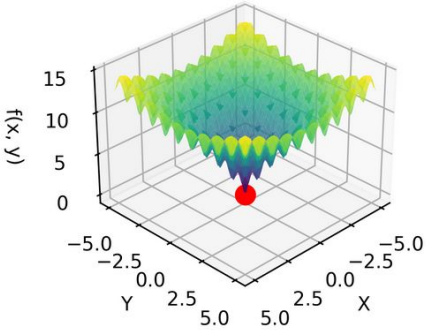


$$f(x) = -20\exp\left(-0.2 \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$$

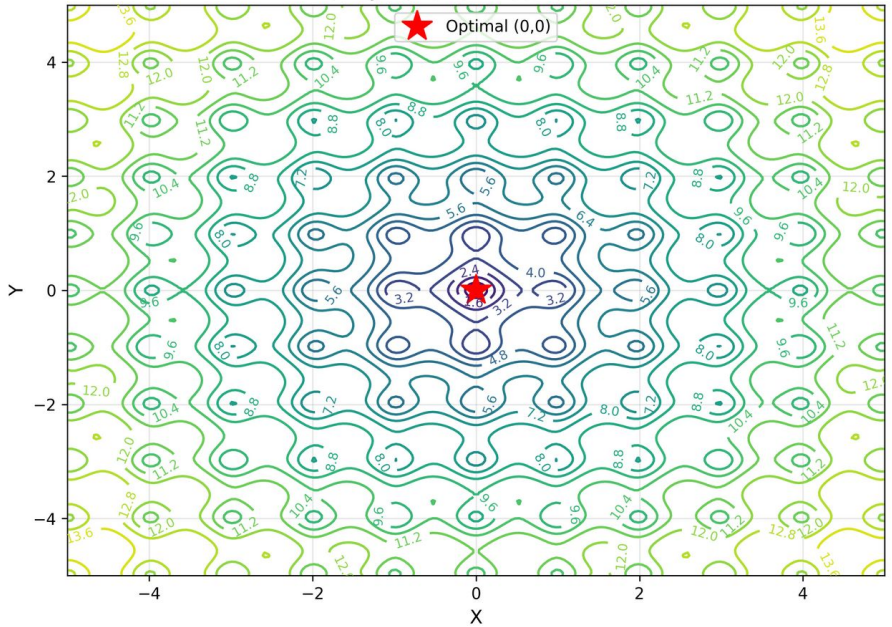
Ackley Function - Side View



Ackley Function - Perspective View



Ackley Function - Contour Plot



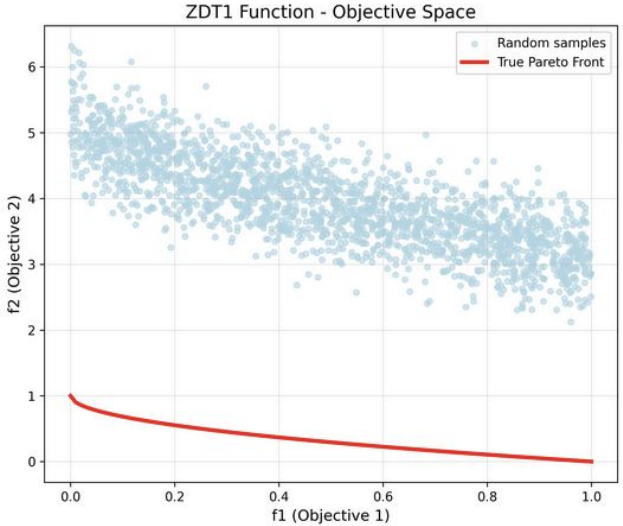
ZDT1 Function

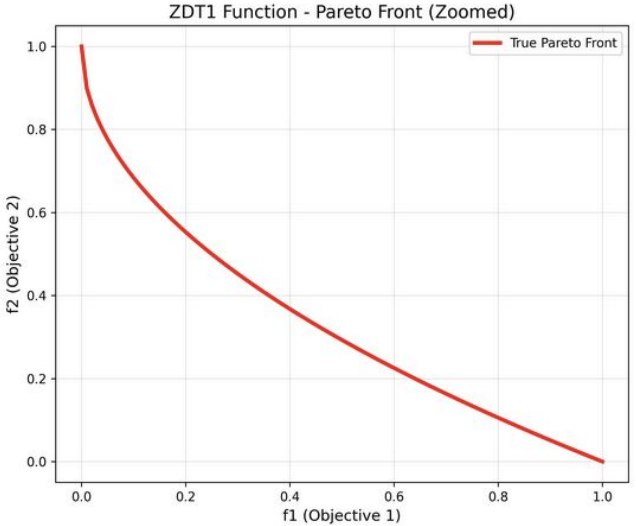
$$f_1(x) = x_1$$

$$f_2(x) = g(x)\left[1 - \frac{x_1}{g(x)}\right]$$

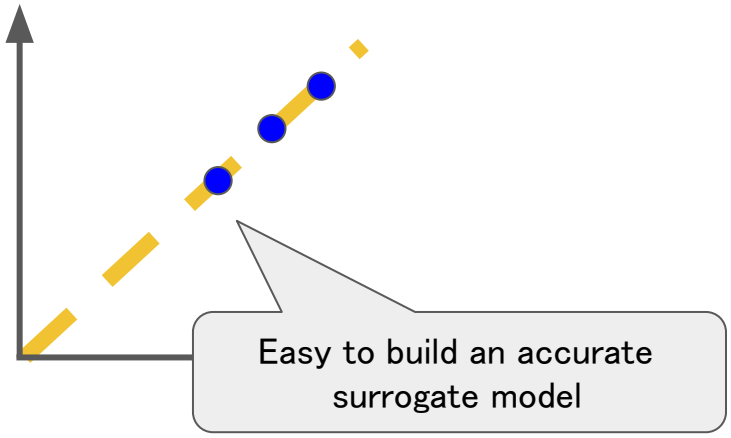
$$\text{where } g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$$

$$\text{and } x_i \in [0,1] \text{ for } i = 1, 2, \ldots, n$$

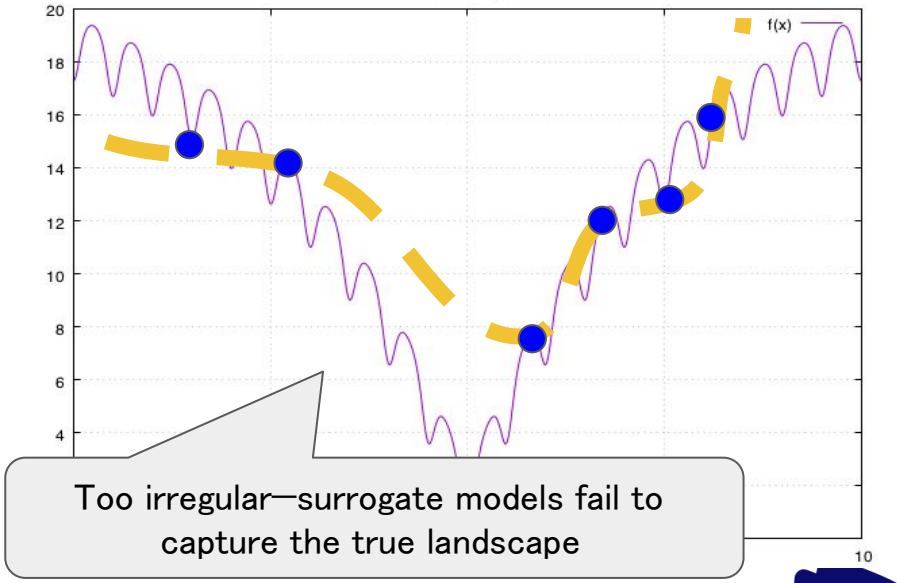




Why Does GP Struggle with the Ackley Function?



Case: Addition



Case: Ackley Function

Practice Time

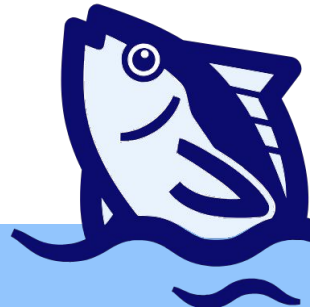
Apply what you've learned so far to perform optimization.

Which sampler performs best with optimization?

The number of objective functions also affects results—**experience it for yourself!**

If you want to experiment with other benchmark functions or review the flow for selecting samplers as demonstrated in the presentation, you can also check the Tunny documentation site.

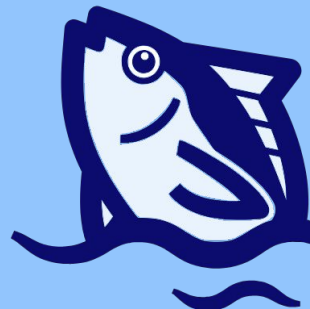
Please refer to that resource as well.
Benchmark Functions | Tunny docs



Practice Time

15-minute

13:00(EST) start next session



Method Selection in Facade Optimization (Research Introduction)

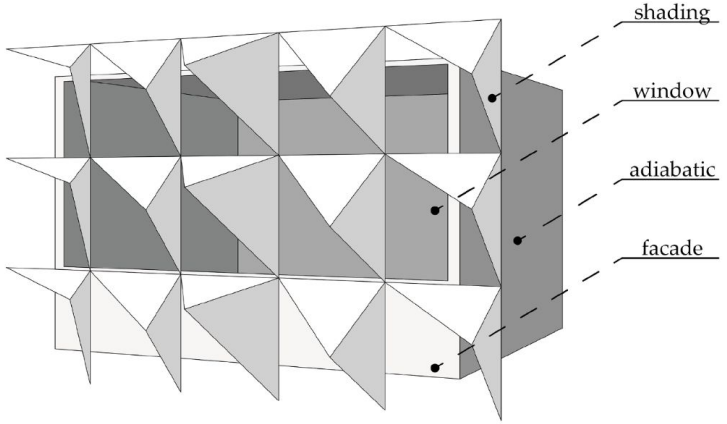
In actual architectural design problems, which approach would be most effective?

As an example, let me introduce a research paper.

This research uses both Opossum and Tunny for analysis, and focuses on the facade design and environmental performance of office buildings.

Since environmental analysis of facades requires significant time for evaluation, this paper recommends using SMBO-based models like TPE or RBFOpt as appropriate approaches.

For detailed information, please refer to the original paper.

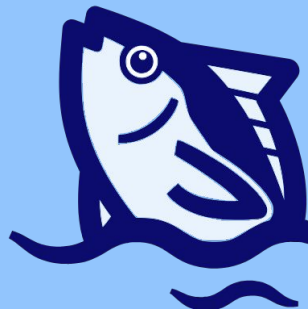


Max Zorn, Luisa Claus, Christian Frenzel, Thomas Wortmann,

"Optimizing an expensive multi-objective building performance problem: Benchmarking model-based optimization algorithms against metaheuristics with and without surrogates",

Energy and Buildings, Volume 336, 2025

Human in the Loop



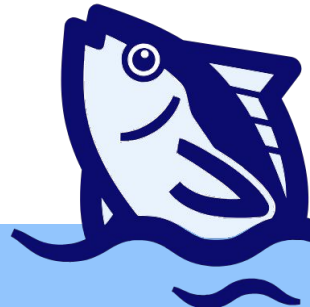
Optimize Person Ratings

Until now, optimization has primarily focused on **problems that can be evaluated computationally using Grasshopper.**

However, in real-world design challenges, there are often situations where we want to **incorporate human perceptions** —such as beauty, comfort, or usability—**into the design process.**

Optimization techniques that use such **human evaluations as objective functions** are known as **Human-in-the-Loop approaches.**

Tunny supports Human-in-the-Loop optimization, allowing you to integrate subjective human judgment directly into your parametric design workflow.



Challenges in Human-in-the-Loop Optimization

While Human-in-the-Loop optimization is powerful, it presents unique challenges:

➔ **Challenge 1: Relative vs. Absolute Evaluation**

- Humans naturally evaluate by comparison, not absolute metrics
- ″This design is better than that one″ is easier than ″This design scores 7.3 out of 10″
- Consistent absolute scoring across many trials is extremely difficult

➔ **Challenge 2: Evaluation Fatigue**

- Traditional optimization requires hundreds or thousands of evaluations
- Humans quickly become tired and inconsistent when rating many designs
- Decision quality deteriorates over time

The Problem: Standard optimization algorithms expect fast, consistent evaluations—but human judgment is slow and can vary with fatigue.

Preferential Bayesian Optimization: PBO

The solution to Human-in-the-Loop challenges is Preferential Bayesian Optimization (PBO).

How PBO Solves the Challenges:

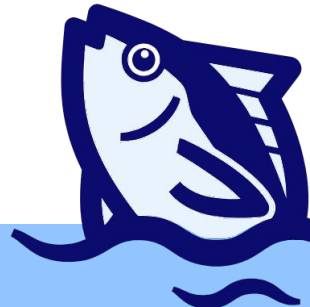
➔ **1. Preference-Based Evaluation**

- Instead of absolute scores, you simply choose which design you prefer
- "A is better than B" is much easier than scoring each design numerically
- Works naturally with how humans actually make decisions

➔ **2. Efficient Sample Selection**

- PBO intelligently selects which designs to compare, minimizing the number of evaluations needed
- Uses Bayesian optimization to learn from your preferences and quickly converge to optimal designs
- Dramatically reduces evaluation fatigue by requiring far fewer comparisons

Result: PBO makes Human-in-the-Loop optimization practical by aligning with human cognitive strengths.

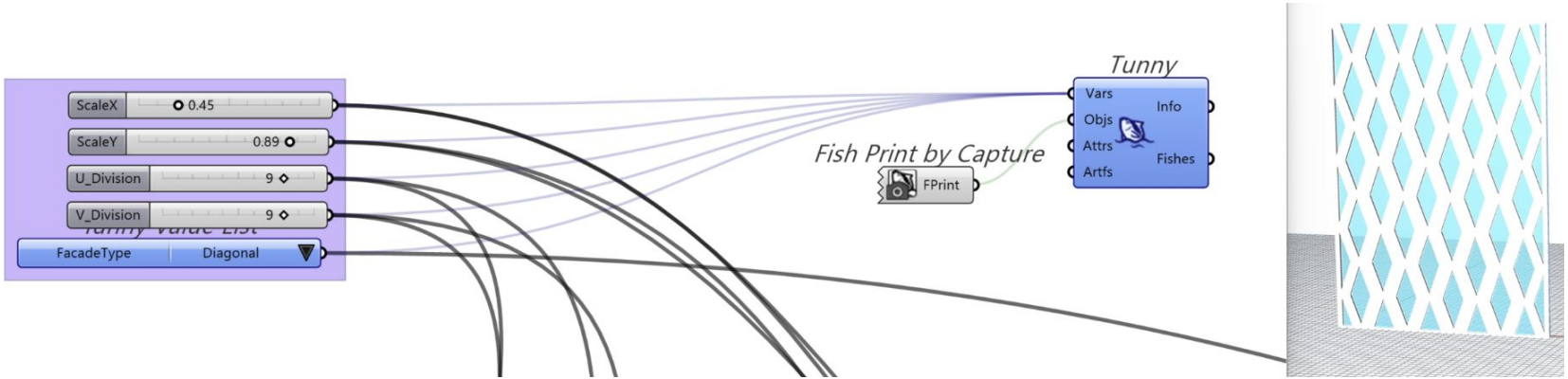


Human-in-the-Loop with Tunny

Now, let's actually perform a Human-in-the-Loop process using Tunny.

Open the file "**03_human_in_the_loop.gh**"

This file contains a single surface designed for facade, controlling the orientation of the louvers (whether diagonal or vertical) as well as their number and thickness.



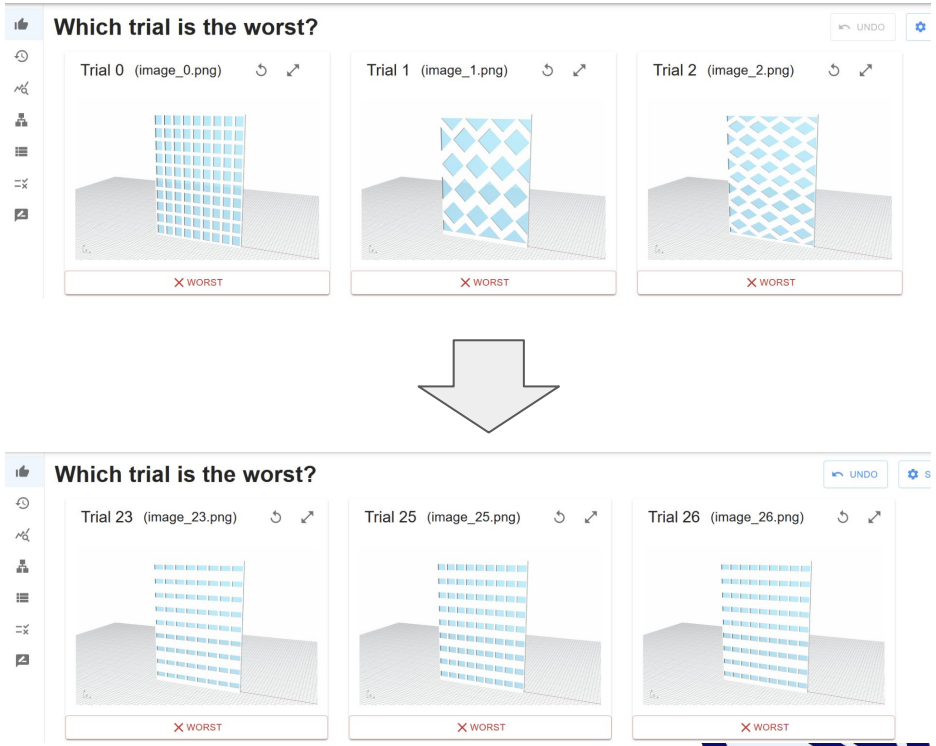
Building facade optimization

A practical example of PBO applied to building facade design.

How it works:

- ➔ The interface asks: "Which trial is the worst?"
- ➔ You select the design you like least
- ➔ The algorithm learns your preferences and converges toward your taste

Result: After multiple selections, the optimization learned I prefer horizontally-emphasized patterns—discovered entirely through comparative judgments, no scoring needed.



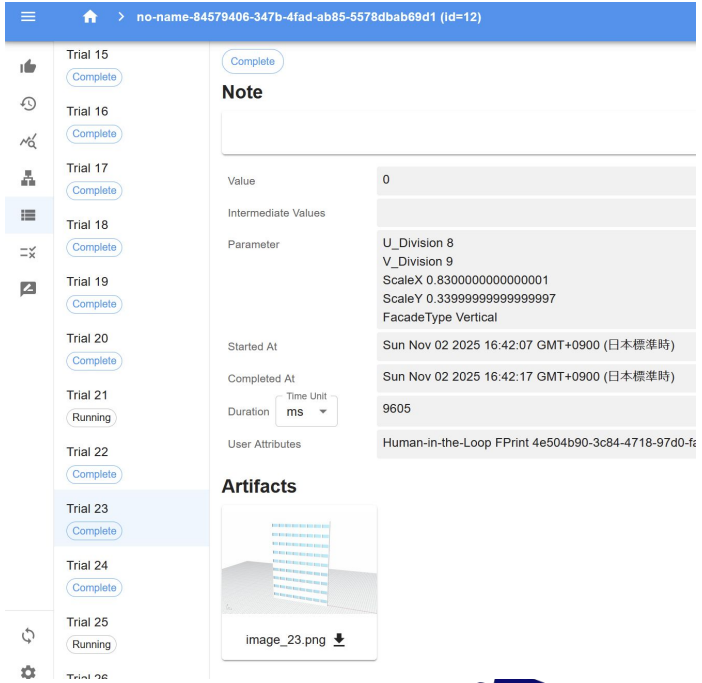
Building facade optimization: analysis

Since this is optimization in Tunny, you can do more than just select images—you can analyze the underlying parameters just like any other optimization.

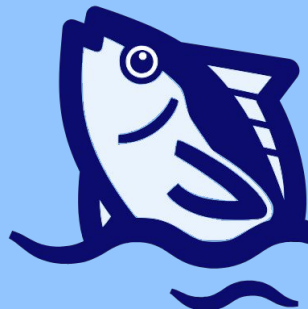
Optimization data available:

- ➔ View the parameter values that generated each design
- ➔ Download artifacts for further exploration
- ➔ Track optimization progress over time

This means you're not just picking favorite images—**you're discovering the design rules** behind your preferences.



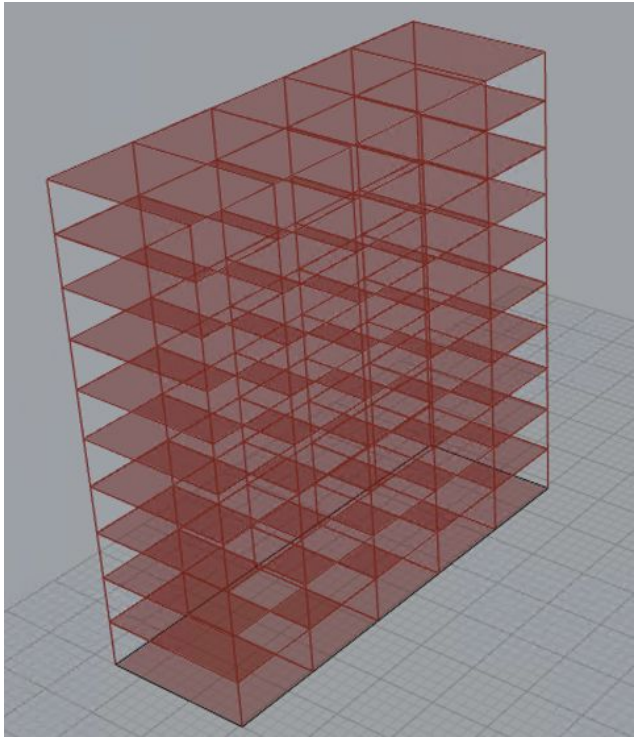
Exploration of Design Space



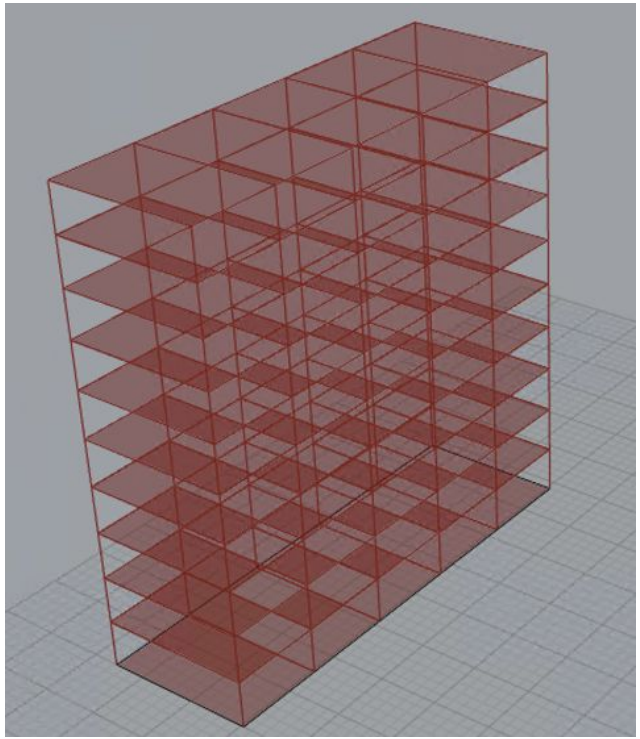
Non-optimized Usage: Problem

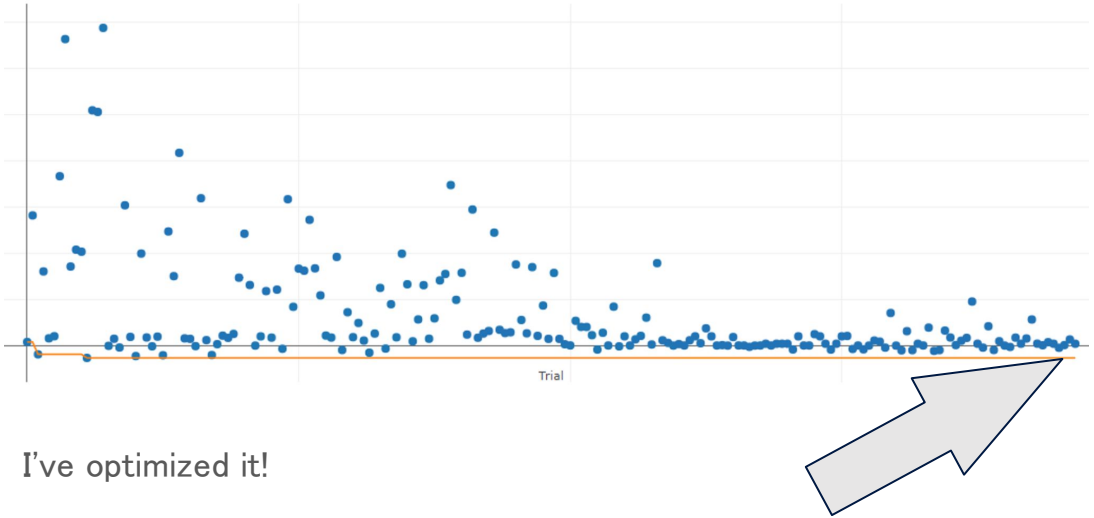
We are in the early stages of design: 1 span x 5 spans, 11 story building,

What are the member cross-sections?



Non-optimized Usage: Optimize





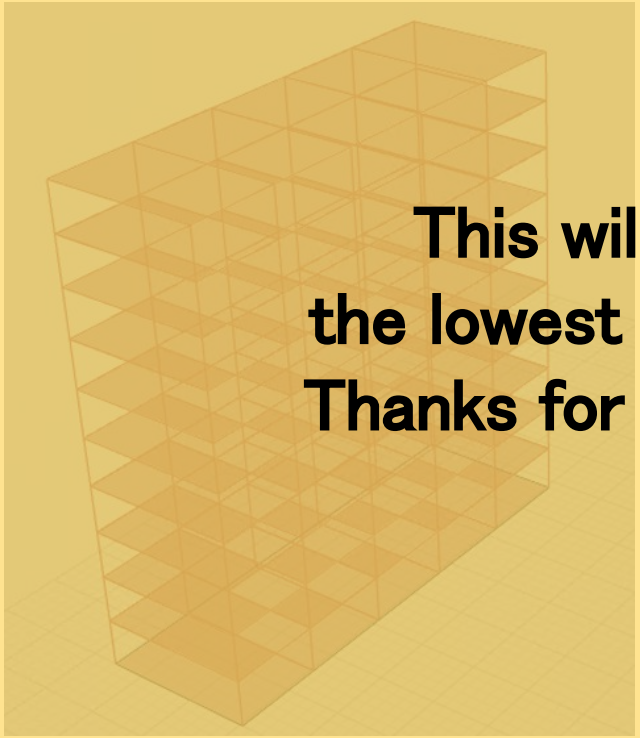
I've optimized it!

It's a basic plan, and I'll start with a uniform

➔ Beam: I −900x300x16x28
➔ Column: □−700x700x12

is optimal, so let's go with the structure 👍

Ideal timized Usage: Optimize



This will allow me to propose
the lowest amount of steel framing!
Thanks for the computer 💖 love it!

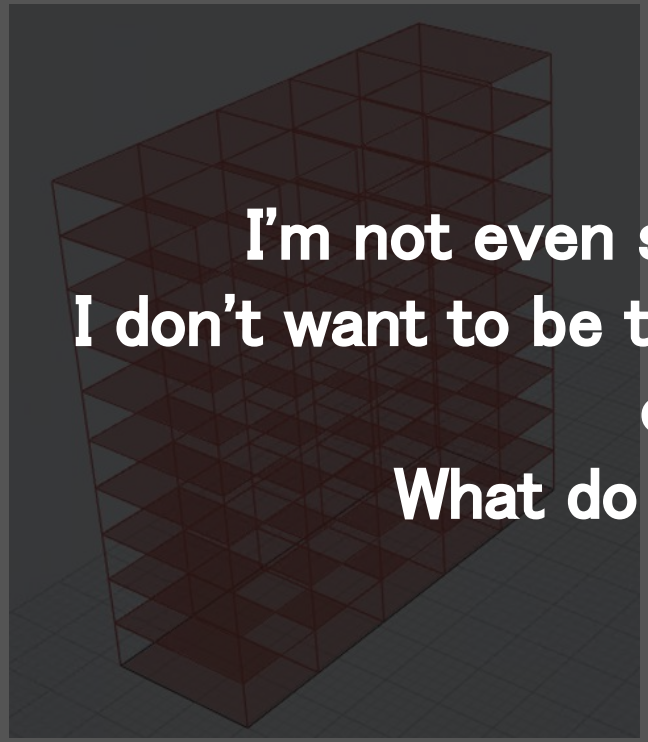
I've optimized it!

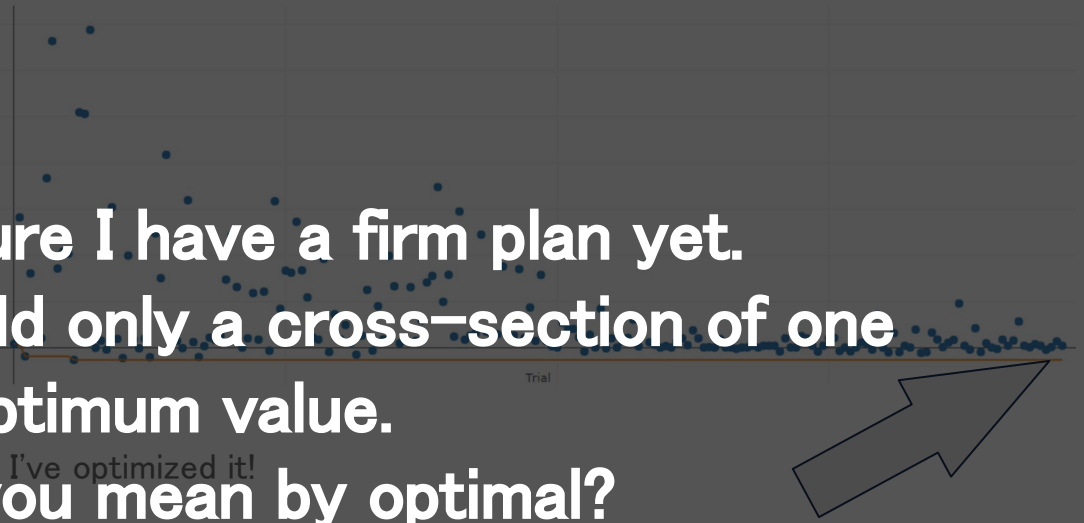
It's a basic plan, and I'll start with a uniform

➔ Beam: I -900x300x16x28
➔ Column: □-700x700x12

is optimal, so let's go with the structure 👍

Real ptimized Usage: Optimize





I'm not even sure I have a firm plan yet.
I don't want to be told only a cross-section of one optimum value.
What do you mean by optimal?

I've optimized it!
It's a basic plan, and I'll start with a uniform

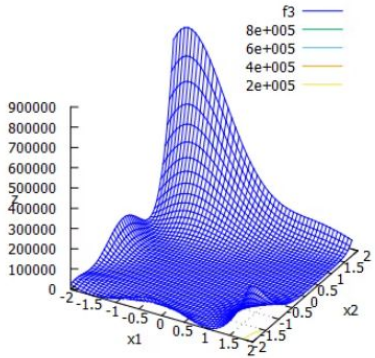
➔ Beam: I -900x300x16x28
➔ Column: □-700x700x12

is optimal, so let's go with the structure 👍

Considering the Design Space

What shape is this function?

$$f(x_1, x_2) = \Big(1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\Big)$$
$$\Big(30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\Big)$$



Design space is "the range in which f(x1, x2) solutions can exist".

For example, the area shown in blue on the left is design space of the above function

So what is the form of this function that you want to check in this design?

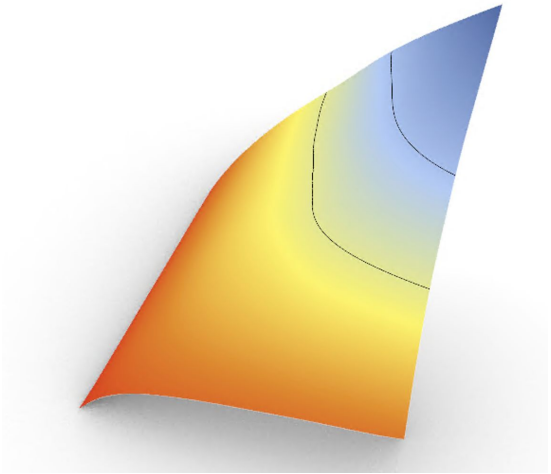
$$f(column, beam) = ???$$

Understanding DesignSpace makes it a useful tool

Once the design space is revealed, it can be tooled.

Tunny also supports design space search functionality (Quasi-Monte Carlo method)

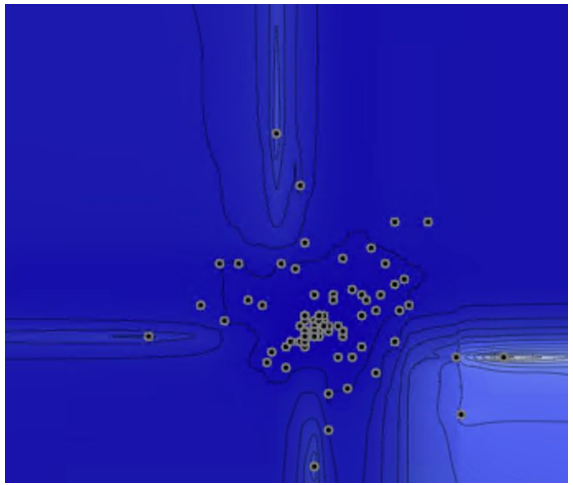
$$f(column, beam) =$$



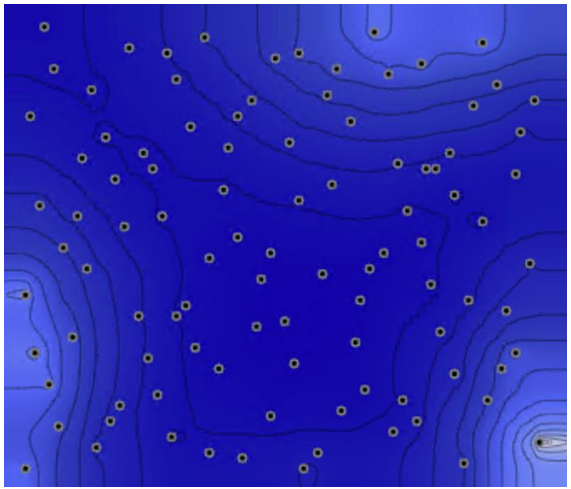
Why not conventional optimization tools?

Two methods were explored for the same objective function.
Optimization algorithms search for the best possible solution. This approach doesn't provide any information about how the solution compares to other potential points.

On the other hand, uniformly sampling approaches like QMC provide a comprehensive view of the entire space, making it particularly useful for understanding the characteristics of the DesignSpace.



Optimization Algorithm



Quasi-Monte Carlo

Tunny
Lecture & Hands-on