Overview

This repository contains the results of the SausageDog hack at AECTech Barcelona 2024.

This repository demonstrates how to create a web interface that allows users to interact with a Grasshopper script. The Grasshopper script is structured to be able to create multiple options for various applications such as project designs, recipes, and more. The interface is designed with ease of use in mind, focusing only on developer-selected inputs.

Key Components

Grasshopper Script:

- Utilizes sliders to generate multiple options.
- Outputs numerical scores for each option to indicate performance or suitability.

Web Interface:

- Developed using ShapeDiver, a React-based library that simplifies the creation of web interfaces capable of running Grasshopper scripts.
- Includes a 3D viewer powered by three.js, enhancing the interactive experience by visualizing script outputs.

Optimization Algorithm:

- Implements the NSGA-II (Non-dominated Sorting Genetic Algorithm II), a multi-objective optimization technique. FInd the original code byIbnu Athaillah here
- Capable of recursively generating populations of options and selecting the optimal ones.
- Unlike other optimization applications which run scripts internally, ShapeDiver efficiently manages script execution recursively, providing a more elegant solution.

Purpose

This setup allows users to easily interact with complex algorithms through a simple graphical interface, streamlining the process of exploring and selecting optimal solutions across various use cases.

Example Implementations

To demonstrate the capabilities of this setup, two Grasshopper scripts have been implemented, each showcasing different optimization scenarios:

1. Hotdog Optimizer

User Interaction:

Users can select from a list to decide the ingredients for their hotdog.

Optimization Objectives:

Prioritize by maximizing tastiness, minimizing calories, or maximizing cost based on user-selected weights.

Iterative Feedback:

At the end of optimization cycles, the interface displays data for the option that is nearest to the Pareto front and closest to the user's priority weight vector.

Post-Optimization:

Users can adjust priority weights post-optimization to explore other Pareto-efficient options.

2. Barcelona Blocks Optimizer

Script Functionality:
The script models urban structures with block layouts similar to those in Barcelona.
User Inputs:
Users can iterate over dimensional variables such as block size, courtyard size, and the angle of street intersections.
Optimization Objectives:
Set weights to minimize facade area, maximize street size, and maximize courtyard dimensions.
Iterative Feedback:
The application provides visualizations of data for options on the Pareto front that best match the priority vector at the end of the iterations.
Post-Optimization:
Allows users to modify the weights and view different Pareto-efficient options available.
These examples illustrate how users can directly influence the optimization process through simple interface elements, making complex algorithmic adjustments accessible and interactive.