**Инструкция пользователя**

**GEFEST** — open-source фреймворк генеративного проектирования физических объектов, взаимодействующих со сплошными средами. Фреймворк позволяет создавать геометрические объекты и оптимизировать их эволюционными методами для решения поставленных задач. Последовательность выполнения алгоритма GEFEST представлена на рисунке 1. Области применения:

1. Генерация ряда структур геометрических объектов, решающих задачи из различных инженерных сфер (гидродинамика, проектирование волнозащитных сооружений и др.).
2. Восстановление формы объектов по малоинформативному сигналу, находящихся в звуковом поле.
3. Выполнение многокритериальной оптимизации, ряда задач, не решаемых градиентными методами.
4. Нахождения оптимального маршрута и расположения объектов на местности.



Рисунок 1 – Подход GEFEST к генеративному проектированию физических объектов, включая кодирование полигонов, создание набора инструментов и генеративный дизайн.

На рисунке 2 приведена общая схема фреймворка. Она включает в себя блоки, предназначенные для решения задач генеративного дизайна.



Рисунок 2 – Схема реализации фреймворка GEFEST. Основные блоки для решения задачи: конфигурация Домена геометрических структур, конфигурация инструментов и блок проектирования.

Для решения задач генеративного дизайна физических объектов требуется определить три основных блока фреймворка в соответствии с постановкой задачи и ее граничными условиями.

Блок инициализации домена геометрических структур заключается в выполнении нескольких этапов. В первую очередь задается подблок Geometry, который описывает возможные трансформации, применимые к элементам структуры (полигонам и точкам), а также правило генерации полигонов в структуре: их число, размер, форму. В зависимости от постановки задачи, необходимо определить и задать разрешенные и запрещенные области для генерации геометрических структур.

Далее необходимо описать логику блока инструментов. Для решения различных задач часто необходимо описать различную логику инструментария. Таким образом, в блоке Toolkit описываются подблоки Sampler, Estimator, Optimizer.

Подблок Sampler задает правила инициализации начальной популяции геометрических структур. В нем описывается размер популяции и метод генерации структур (стандартный алгоритм, или на основе нейронных сетей).

Estimator определяет алгоритм или программу-симулятор, в котором рассчитываются характеристики рассматриваемой постановки задачи для каждой сгенерированной геометрической структуры. Optimizer описывает методы оптимизации задачи. В зависимости от условий задачи задается эволюционный алгоритм, определяющий логику оптимизации. Так, например, при многокритериальной постановке задачи, выбирается SPEA2 находящий множество оптимальных по Парето решений.

 Таким образом, GEFEST реализует итеративное выполнение вышеперечисленных блоков, в результате которого находится решение поставленной задачи.

**Пример работы с фреймворком для решения задачи оптимизации волнозащитных сооружений.**

* Шаг 1. Установка фреймворка и импорт необходимых библиотек:

pip install <https://github.com/ITMO-NSS-team/GEFEST/archive/master.zip>

import timeit

import pickle

from types import SimpleNamespace

import matplotlib.pyplot as plt

import numpy as np

from gefest.core.geometry.geometry\_2d import Geometry2D

from gefest.tools.estimators.simulators.swan.swan\_interface import Swan

from gefest.core.opt.setup import Setup

from gefest.tools.optimizers.optimizer import Optimizer

from gefest.core.structure.domain import Domain

from gefest.core.structure.structure import Structure

from gefest.core.opt.analytics import EvoAnalytics

from gefest.core.viz.struct\_vizualizer import StructVizualizer

from gefest.tools.estimators.estimator import Estimator

from gefest.tools.samplers.standard.standard import StandardSampler

from gefest.tools.samplers.sampler import Sampler

from gefest.tools.optimizers.SPEA2.SPEA2 import SPEA2

from gefest.core.opt.operators.operators import default\_operators

from gefest.core.opt.gen\_design import design

from gefest.tools.estimators.simulators.swan import swan\_model

from gefest.core.structure.prohibited import create\_prohibited

* Шаг 2. Описание домена (области генерации структур):

grid\_resolution\_x = 83 # Number of points on x-axis

grid\_resolution\_y = 58 # Number of points on y-axis

coord\_X = np.linspace(0, 2075, grid\_resolution\_x + 1) # X coordinate for spatial grid

coord\_Y = np.linspace(0, 1450, grid\_resolution\_y + 1) # Y coordinate for spatial grid

X, Y = np.meshgrid(coord\_X, coord\_Y) # Two dimensional spatial grid

grid\_target\_X = 25 # X-grid coordinate of your target

grid\_target\_Y = 25 # Y-grid coordinate of your target

grid = [grid\_resolution\_x, grid\_resolution\_y]

targets = [[grid\_target\_X, grid\_target\_Y]]

fixed\_area = [

[[471, 5], [1335, 2], [1323, 214], [1361, 277], [1395, 327], [1459, 405], [1485, 490], [1449, 521], [1419, 558],

 [1375, 564], [1321, 469], [1248, 318], [1068, 272], [921, 225], [804, 231], [732, 266], [634, 331], [548, 405],

 [485, 482], [424, 569], [381, 625], [310, 662], [271, 684], [244, 706], [203, 708], [182, 647], [214, 638],

 [234, 632], [275, 588], [346, 475], [427, 366], [504, 240], [574, 166], [471, 5]],

[[652, 1451], [580, 1335], [544, 1253], [468, 1190], [439, 1170], [395, 1150], [378, 1115], [438, 1070],

 [481, 1059], [508, 1076], [539, 1133], [554, 1183], [571, 1244], [594, 1305], [631, 1366], [657, 1414],

 [671, 1449], [652, 1451]]

]

fixed\_targets = [[coord\_X[26], coord\_Y[49]], [coord\_X[37], coord\_Y[11]], [coord\_X[60], coord\_Y[5]]]

fixed\_poly = [

 [[878, 1433], [829, 1303], [739, 1116], [619, 995], [447, 962], [306, 1004], [254, 1092], [241, 1184],

 [269, 1244],

 [291, 1338], [370, 1450]],

 [[878, 1433], [829, 1303], [739, 1116], [619, 995], [447, 962], [274, 868], [180, 813], [126, 717], [146, 580],

 [203, 480], [249, 469], [347, 471]]

]

# Creation of prohibited structure consist of targets, lines, areas

prohibited\_structure = create\_prohibited(

 targets=fixed\_targets,

 fixed\_area=fixed\_area,

 fixed\_points=fixed\_poly

)

fixed\_points = [[[1000, 50], [700, 600], [800, 800]],

 [[1900, 540], [1750, 1000]]]

is\_closed = False

geometry = Geometry2D(is\_closed=is\_closed)

domain = Domain(allowed\_area=[(min(coord\_X), min(coord\_Y)),

 (min(coord\_X), max(coord\_Y)),

 (max(coord\_X), max(coord\_Y)),

 (max(coord\_X), min(coord\_Y))],

 geometry=geometry,

 max\_poly\_num=3,

 min\_poly\_num=1,

 max\_points\_num=10,

 min\_points\_num=2,

 prohibited\_area=prohibited\_structure,

 fixed\_points=fixed\_points,

 is\_closed=is\_closed)

task\_setup = Setup(domain=domain)

* **Шаг 3**. Инициализация модели-симулятора волн SWAN

path = swan\_model.\_\_file\_\_[:-11]

swan = Swan(path=path,

 targets=targets,

 grid=grid,

 domain=domain)

max\_length = np.linalg.norm(np.array([max(coord\_X) - min(coord\_X), max(coord\_Y) - min(coord\_Y)]))

* **Шаг 4**. Описании функции определяющей приспособленность популяции – стоимость построения структуры и высоту волн. Инициализация Estomatorа.

def cost(struct, estimator):

 max\_length = np.linalg.norm(

 np.array([max(coord\_X) - min(coord\_X),

 max(coord\_Y) - min(coord\_Y)]))

 lengths = 0

 for poly in struct.polygons:

 if poly.id != 'fixed':

 length = geometry.get\_length(poly)

 lengths += length

 \_, hs = estimator.estimate(struct)

 loss = [hs, 2 \* lengths / max\_length]

 return loss

estimator = Estimator(estimator=swan, loss=cost)

* **Шаг 5**. Инициализация Sampler.

sampler = Sampler(sampler=StandardSampler(), domain=domain)

* **Шаг 6**. Инициализация Optimizer.

pop\_size = 10

n\_steps = 10

params = SPEA2.Params(pop\_size=pop\_size,

 crossover\_rate=0.6,

 mutation\_rate=0.6,

 mutation\_value\_rate=[])

spea2\_optimizer = SPEA2(params=params,

 evolutionary\_operators=default\_operators(),

 task\_setup=task\_setup)

* **Шаг 7**. Запуск расчета.

start = timeit.default\_timer()

optimized\_pop = design(n\_steps=n\_steps,

 pop\_size=pop\_size,

 estimator=estimator,

 sampler=sampler,

 optimizer=spea2\_optimizer)

spend\_time = timeit.default\_timer() - start

По итогу выполнения расчета, алгоритм сохранит результаты оптимизации в папке “HistoryFiles”. После чего можно провести визуализацию решений, полученных фреймворком, использую следующий код:

with open(f'HistoryFiles/performance\_{n\_steps-1}.pickle', 'rb') as f:

 performance = pickle.load(f)

with open(f'HistoryFiles/population\_{n\_steps-1}.pickle', 'rb') as f:

 population = pickle.load(f)

performance\_sum = [sum(pair) for pair in performance]

idx\_of\_best = performance\_sum.index(min(performance\_sum))

visualiser = StructVizualizer(task\_setup.domain)

plt.figure(figsize=(7, 7))

best = performance[idx\_of\_best]

info\_optimized = {

 'spend time': f'{spend\_time:.2f}',

 'fitness': f'[{best[0]:.3f}, {best[1]:.3f}]',

 'type': 'prediction'}

visualiser.plot\_structure(

 [domain.prohibited\_area, population[idx\_of\_best]],

 ['prohibited structures', info\_optimized],

 [':', '-'])

**Компонент размещен в репозитории:**

* <https://github.com/aimclub/GEFEST/tree/main>