

RENATO GODOI DA CRUZ

arquiteto urbanista

contato

renatogcruz@hotmail.com
(19) 9 9716-5302

endereço

rua mário pinto sobrinho, 87
santa mônica
uberlândia | mg. barsil

ÁREAS DE CONHECIMENTO E ATUAÇÃO

- análise de projeto arquitetônico
- projeto arquitetônico
- projeto arquitetônico em aço
- bim
- legislação urbanística

HABILIDADES

modelagem bim



edição de imagens



design gráfico



programação textual



programação visual



EDUCAÇÃO E EXPERIÊNCIA ACADÊMICA

- 2020 - atual especialização em building information modeling
pontifícia universidade católica - puc mg
- 2016 - 2018 mestrado profissional em construção metálica
universidade federal de ouro preto - ufop mg
título - design generativo como ferramenta e metodologia projetual na construção metálica
- 2010 - 2015 bacharel em arquitetura e urbanismo.
universidade federal de ouro preto - ufop mg
título - mapas contemporâneos: os mapas colaborativos como meio de produção, apropriação e planejamento das cidades

FORMAÇÃO COMPLEMENTAR

- 2017 - 2017 estudo de impacto de vizinhança - eiv
ministério das cidades - 30 h - 2017-2017
- 2015- 2015 gestão de projetos urbanos
ministério das cidades - 24 h - 2015-2015
- 2015 - 2015 workshop tecnopolítica, democracia e urbanismo tático
coletivo micrópolis + indisciplinar | ufmg
- 2013- 2013 reabilitação urbana com foco em áreas centrais
ministério das cidades 28 h - 2013-2013

EXPERIÊNCIAS PROFISSIONAIS

- 2019 - atual analista de desenvolvimento urbano prefeitura uberlândia - mg
Analista de desenvolvimento urbano na Diretoria de Acessibilidade e Mobilidade Reduzida (DAMR) da Secretaria de Planejamento Urbano (Seplan) da Prefeitura de Uberlândia - PMU (MG). Atuações: 1 - Análise de projeto de acessibilidade e mobilidade reduzida¹. 2 - Vistoria/Fiscalização de acessibilidade e mobilidade reduzida¹.
¹NBR9050/2015, Lei N° 13.146/2015 (Lei Brasileira de Inclusão da Pessoa com Deficiência) e legislações municipais.
- 2016 - atual arquiteto urbanista
projeto arquitetônico, desenvolvimento de projeto arquitetônico, projeto de acessibilidade (nbr9050/2020, lei brasileira de inclusão).
- 2014 - 2014 estágio virtual arquitetura & ambiente passos - mg
levantamento arquitetônico, desenvolvimento de projeto de arquitetura, desenho arquitetônico, projeto de as-built
- 2014 - 2014 estágio bruno lopes studios ribeirão preto - sp
levantamento arquitetônico, desenvolvimento de projeto de arquitetura, desenho arquitetônico
- 2014 - 2014 estágio renata lemos paisagismo passos - mg
levantamento arquitetônico, desenvolvimento de projeto paisagístico, desenho arquitetônico paisagístico

sumário

portfólio

problema: **xxxxxx**

ambiente: python



03 - algoritmos genéticos

04 - inteligência artificial

05 - algoritmo evolutivo

06 - biblioteca osmnx python

03 - **xxxxxxxxxxxx**

03 - **xxxxxxxxxxxx**

03 - **xxxxxxxxxxxxxx**

03 - **xxxxxxxxxxxxxx**

algoritmos genéticos

Estudos dedicados aos algoritmos genéticos (GA) com Python. Algoritmo genético é uma heurística de busca inspirada na teoria da evolução natural de Darwin. Estes algoritmos refletem o processo de seleção natural, onde os indivíduos mais aptos são selecionados para reprodução, a fim de produzir descendentes mais aptos para a próxima geração.

problema: Algoritmo genético criado para reproduzir a frase "cargo DAM-4". Para isso, o algoritmo precisa de um conjunto de 'genes' para usar na construção de palpites (geneSet). Também precisa de um objetivo (target). Isso parece bobagem, se conhecemos o resultado desejado, por que escrever um algoritmo? A resposta é simples: este é um exercício.

Referência: Genetic algorithms with Python, de Clinton Sheppard.

ambiente: python

cargo_dam4.py

```
import random
import datetime

def get_fitness(guess):
    return sum(1 for expected, actual in zip(target, guess)
              if expected == actual)

def display(guess):
    timeDiff = datetime.datetime.now() - startTime
    fitness = get_fitness(guess)
    print("{0}\t{1}\t{2}".format(guess, fitness, str(timeDiff)))

def generate_parent(length):
    genes = []
    while len(genes) < length:
        sampleSize = min(length - len(genes), len(geneSet))
        genes.extend(random.sample(geneSet, sampleSize))
    return ''.join(genes)

def mutate(parent):
    index = random.randrange(0, len(parent))
    childGenes = list(parent)
    newGene, alternate = random.sample(geneSet, 2)
    childGenes[index] = alternate \
        if newGene == childGenes[index] \
        else newGene
    return ''.join(childGenes)

random.seed()
geneSet = "1234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!,-"
target = "cargo DAM-4"
startTime = datetime.datetime.now()

bestParent = generate_parent(len(target))
bestFitness = get_fitness(bestParent)
display(bestParent)
while True:
    child = mutate(bestParent)
    childFitness = get_fitness(child)

    if bestFitness >= childFitness:
        continue
    display(child)
    if childFitness >= len(bestParent):
        break
    bestFitness = childFitness
    bestParent = child
```

Out: [

4jTigCf;Dy /	0	0:00:00.000549
4aTigCf;Dy /	1	0:00:00.001553
caTigCf;Dy /	2	0:00:00.002553
caTigCfADy /	3	0:00:00.004553
caTigCfAD-/	4	0:00:00.004553
caTigCfAM-/	5	0:00:00.004553
caTigCDAM-/	6	0:00:00.007549
caTioCDAM-/	7	0:00:00.010547
carioCDAM-/	8	0:00:00.013545
carioCDAM-4	9	0:00:00.022541
cargoCDAM-4	10	0:00:00.023540
cargo DAM-4	11	0:00:00.025539

[Finished in 2.8s]

]

inteligência artificial

problema: otimização de viagens. com a hipótese de 6 candidatos participarem de uma etapa da seleção para a vaga CAM-4 na cidade de Belo Horizonte, este exercício propõem a otimização das viagens de ônibus, no qual os seis candidatos chegarão na rodoviária de Belo Horizonte com a intenção de compartilhar um mesmo transporte. Nesse cenário, foi utilizado quatro algoritmos (pesquisa aleatória, subida da encosta (hill climb), têmpora simulada (recozimento simulado ou simulated annealing) e algoritmos genéticos) para encontrar os melhores horários de ida e de volta para cada pessoa, com o objetivo de reduzirmos os custos com as passagens e também o tempo de espera que cada pessoa ficará na rodoviária.

legenda: RUB - rod. de uberlândia (mg); RSP - rod. de são paulo (sp); RVI - rod. vitória (es); RRJ - rod. rio de janeiro (rj); RMC - rod. montes claros (mg) e RPA - rod. paracatu (mg)

ambiente: python

otimizacao_viagens.py

```
import time
import random
import math

pessoas = [('Candidato_A', 'RUB'),
           ('Candidato_B', 'RSP'),
           ('Candidato_C', 'RVI'),
           ('Candidato_D', 'RRJ'),
           ('Candidato_E', 'RMC'),
           ('Candidato_F', 'RPA')]

destino = 'RBH'

voos = {}
for linha in open('linhas.txt'):
    #print(linha)
    _origem, _destino, _saida, _chegada, _preco = linha.split(',')
    voos.setdefault((_origem, _destino), [])
    voos[( _origem, _destino)].append(( _saida, _chegada, int(_preco)))

# [1,4, 3,2, 7,3, 6,3, 2,4, 5,3]
def imprimir_agenda(agenda):
    id_voo = -1
    for i in range(len(agenda) // 2):
        nome = pessoas[i][0]
        origem = pessoas[i][1]
        id_voo += 1
        ida = voos[(origem, destino)][agenda[id_voo]]
        id_voo += 1
        volta = voos[(destino, origem)][agenda[id_voo]]
        print('%10s%10s %5s-%5s R$%3s %5s-%5s R$%3s' %
              (nome, origem, ida[0], ida[1], ida[2],
               volta[0], volta[1], volta[2]))

    agenda = [1,4, 3,2, 7,3, 6,3, 2,4, 5,3]
    print("Agenda")
    imprimir_agenda(agenda)
    print("\n")
    [...]
```

out: [

Agenda

Candidato_A	RUB 7:39-10:24 R\$219 12:31-14:02 R\$234
Candidato_B	RSP 11:01-12:39 R\$260 9:11-10:42 R\$172
Candidato_C	RVI 17:07-20:04 R\$291 11:08-14:38 R\$262
Candidato_D	RRJ 15:27-17:18 R\$151 10:33-12:03 R\$ 74
Candidato_E	RMC 9:08-12:12 R\$364 12:20-16:34 R\$500
Candidato_F	RPA 13:40-15:38 R\$137 10:32-13:16 R\$139

Solução Randômica

Candidato_A	RUB 18:12-20:17 R\$242 8:04-10:59 R\$136
Candidato_B	RSP 15:58-18:40 R\$173 6:03- 8:43 R\$219
Candidato_C	RVI 17:07-20:04 R\$291 6:33- 9:14 R\$172
Candidato_D	RRJ 17:11-18:30 R\$108 9:58-11:18 R\$130
Candidato_E	RMC 10:30-14:57 R\$290 6:09- 9:49 R\$414
Candidato_F	RPA 18:35-20:28 R\$204 6:58- 9:01 R\$238

Solução Subida de Encosta

Candidato_A	RUB 16:51-19:09 R\$147 15:07-17:21 R\$129
Candidato_B	RSP 9:42-11:32 R\$169 7:50-10:08 R\$164
Candidato_C	RVI 11:28-14:40 R\$248 6:33- 9:14 R\$172
Candidato_D	RRJ 20:17-22:22 R\$102 10:33-12:03 R\$ 74
Candidato_E	RMC 20:07-23:27 R\$473 9:49-13:51 R\$229
Candidato_F	RPA 8:27-10:45 R\$139 13:37-15:33 R\$142

Solução Tempera Simulada

Candidato_A	RUB 9:15-12:03 R\$ 99 9:31-11:43 R\$210
Candidato_B	RSP 9:42-11:32 R\$169 10:33-13:11 R\$132
Candidato_C	RVI 15:34-18:11 R\$326 9:25-12:46 R\$295
Candidato_D	RRJ 17:11-18:30 R\$108 10:33-12:03 R\$ 74
Candidato_E	RMC 13:54-18:02 R\$294 9:49-13:51 R\$229
Candidato_F	RPA 15:23-17:25 R\$232 13:37-15:33 R\$142

Solução Algoritmo Genético

Candidato_A	RUB 15:03-16:42 R\$135 11:07-13:24 R\$171
Candidato_B	RSP 15:58-18:40 R\$173 10:33-13:11 R\$132
Candidato_C	RVI 15:34-18:11 R\$326 12:37-15:05 R\$170
Candidato_D	RRJ 17:11-18:30 R\$108 10:33-12:03 R\$ 74
Candidato_E	RMC 13:54-18:02 R\$294 10:51-14:16 R\$256
Candidato_F	RPA 15:23-17:25 R\$232 10:32-13:16 R\$139

[Finished in 34.2s]

algoritmo evolutivos

problema: otimização de mudança. Baseado no exercício anterior, este exercício propõem a otimização da mudança do candidato selecionado para a vaga. A empresa contratada para fazer a mudança possui um caminhão com espaço limitado de armazenamento (3 m^3). Este algoritmo consegue gerar a melhor combinação dos objetos que devem ser transportados, levando em consideração o fato de que o candidato pretende levar os produtos mais valiosos e ocupar o máximo de espaço disponível no caminhão.

ambiente: python

```
in: [ objeto           volume   valor
      "Geladeira dakô",    0.751,   999.90
      "Cama box casal",   1.11,    1099.08
      "TV 55",            0.400,   4345.99
      "Fogão Consul 4 bocas", 0.30,    899.10
      "Lava e seca Samsung", 0.51,    3899.00
      "Notebook Asus",     0.527,   3999.00
      "Ventilador Panasonic", 0.496,   199.90
      "Sofá 3 lugares",     1.77,    4915.91
      "Microondas LG",      0.0544,   429.90
      "Geladeira Brastemp",  0.635,   849.00
      "Guarda-roupa 6 portas", 1.686,   599.90 ]
```

otimizacao_mudanca.py

```
from random import random

class Produto():
    """Cria a classe Produtos"""
    def __init__(self, nome, espaco, valor):
        self.nome = nome
        self.espaco = espaco
        self.valor = valor

    #Classe indivíduo I
    #(indivíduos representam as soluções)
    #(um conj. de indivíduos formam uma população)
    #(O cromossomo representa uma solução)
    #(O indiv. pode ser o próprio cromossomo ou pode conter o
    #cromossomo como atributo [dependendo do caso])
    #(gelad. = 0, iphone = 1, notebook = 0, ... (formam o cromosso-
    #mo '0, 1, 0, ...' que nada mais é do que um conjunto de string")
```

```
class Individuo():
    """Cria a classe Individuo"""
    def __init__(self, espacos, valores, limite_espacos, geracao=0):
        self.espacos = espacos
        self.valores = valores
        self.limite_espacos = limite_espacos
        self.nota_avaliacao = 0
        [...]
```

out: [

```
G:0 -> Valor: 13772.89 Espaço: 2.2874 Cromossomo: ['0', '0', '1', '1', '1', '1', '0', '1', '0', '0']
G:1 -> Valor: 13772.89 Espaço: 2.2874 Cromossomo: ['0', '0', '1', '1', '1', '1', '1', '0', '1', '0', '0']
G:2 -> Valor: 13343.07 Espaço: 2.5470000000000006 Cromossomo: ['0', '1', '1', '0', '1', '1', '0', '0', '0', '0', '0']
G:3 -> Valor: 13590.8 Espaço: 2.7344 Cromossomo: ['0', '0', '1', '0', '1', '0', '1', '1', '0', '0', '0']
G:4 -> Valor: 13772.97 Espaço: 2.6014000000000004 Cromossomo: ['0', '1', '1', '0', '1', '1', '0', '0', '1', '0', '0']
G:5 -> Valor: 13772.97 Espaço: 2.6014000000000004 Cromossomo: ['0', '1', '1', '0', '1', '1', '0', '0', '1', '0', '0']
G:6 -> Valor: 14160.0 Espaço: 2.997 Cromossomo: ['0', '0', '1', '1', '0', '1', '0', '1', '0', '0', '0']
G:n -> Valor: n Espaço: n Cromossomo: [n]
G:98 -> Valor: 14621.89 Espaço: 2.9223999999999997 Cromossomo: ['0', '0', '1', '1', '1', '1', '1', '0', '1', '1', '0']
G:99 -> Valor: 14621.89 Espaço: 2.9223999999999997 Cromossomo: ['0', '0', '1', '1', '1', '1', '1', '0', '1', '1', '0']
G:100 -> Valor: 14621.89 Espaço: 2.9223999999999997 Cromossomo: ['0', '0', '1', '1', '1', '1', '1', '0', '1', '1', '0']
```

```
Melhor solução -> G: 18 Valor: 14672.07 Espaço: 2.9014 Cromossomo: ['0', '1', '1', '1', '1', '1', '0', '0', '1', '0', '0']
Nome: Cama box casal R$ 1099.08
Nome: TV 55 R$ 4345.99
Nome: Fogão Consul 4 bocas R$ 899.1
Nome: Lava e seca Samsung R$ 3899.0
Nome: Notebook Asus R$ 3999.0
Nome: Microondas LG R$ 429.9
[Finished in 0.7s]
```

]

osmnx

OSMnx é um pacote Python que permite baixar geometrias espaciais e modelar, projetar, visualizar e analisar redes de ruas do mundo real a partir de APIs do OpenStreetMap. Pode-se trabalhar facilmente com amenidades / pontos de interesse, projeções de edifícios, dados de elevação, posições / orientações de ruas, velocidade / tempo de viagem e rotas de rede.

Referência: Geoff Boeing.

Ambiente: python + jupyter notebook

obtenha um gráfico para alguma cidade

OSMnx apresenta demonstração

Obtenha redes de ruas e outros dados espaciais em qualquer lugar do mundo a partir do OpenStreetMap e analise-os e visualize-os.

in: 'Belo Horizonte, BR'

out: [



in: qual o tamanho da área que nossa rede cobre em metros quadrados?

out: 375219445.8146904

in: mostre algumas estatísticas básicas sobre a rede

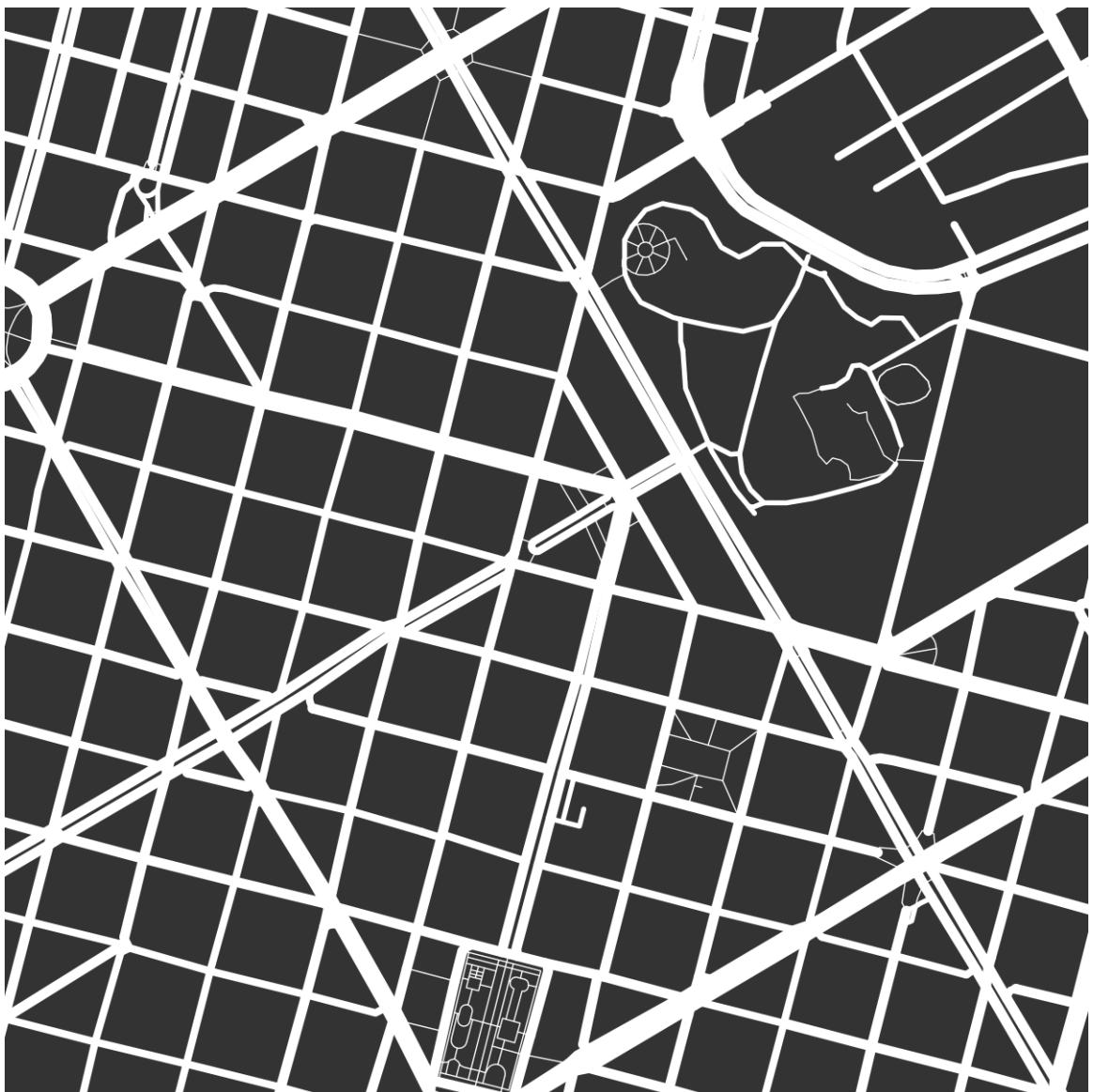
out: {'n': 30349,
'm': 81084,
'k_avg': 5.343438004547102,
'intersection_count': 27997,
'streets_per_node_avg': 3.1308115588652017,
'streets_per_node_counts': {0: 0,
1: 2352,
2: 158,
3: 19347,
4: 8181,
5: 283,
6: 27,
7: 1},
'streets_per_node_proportion': {0: 0.0,
1: 0.0774984348742957,
2: 0.005206102342746054,
3: 0.6374839368677716,
4: 0.2695640713038321,
5: 0.009324854196184389,
6: 0.0008896504003426802,
7: 3.295001482750667e-05},
'edge_length_total': 8438061.147000043,
'edge_length_avg': 104.06567444872037,
'street_length_total': 4901437.207000002,
'street_length_avg': 103.70339385155725,
'street_segments_count': 47264,
'node_density_km': 80.88333464195898,
'intersection_density_km': 74.61500279979326,
'edge_density_km': 22488.336468487156,
'street_density_km': 13062.854981723614,
'circuitry_avg': 1.0380205725208402,
'self_loop_proportion': 0.0010482955946919245,
'clean_intersection_count': 22207,
'clean_intersection_density_km': 59.18403283119652}

diagramas *figure-ground* da rede de ruas

Use o OSMnx para baixar redes de ruas de cidades de milhas quadradas e visualizá-las como diagramas *figure-ground*

in: (-19.925781, -43.937147)

out: [

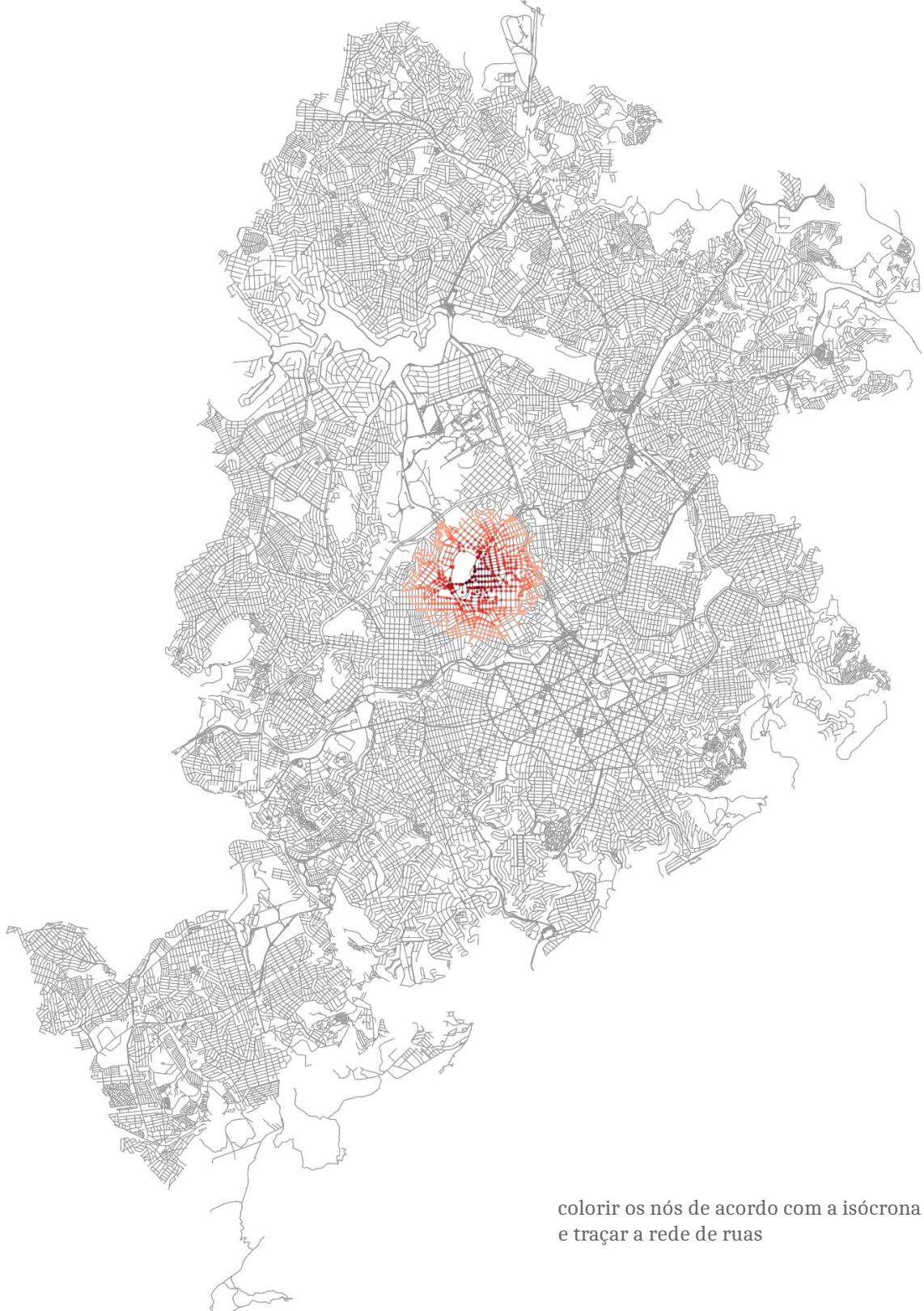


]

desenhe um mapa de isócronas com OSMnx

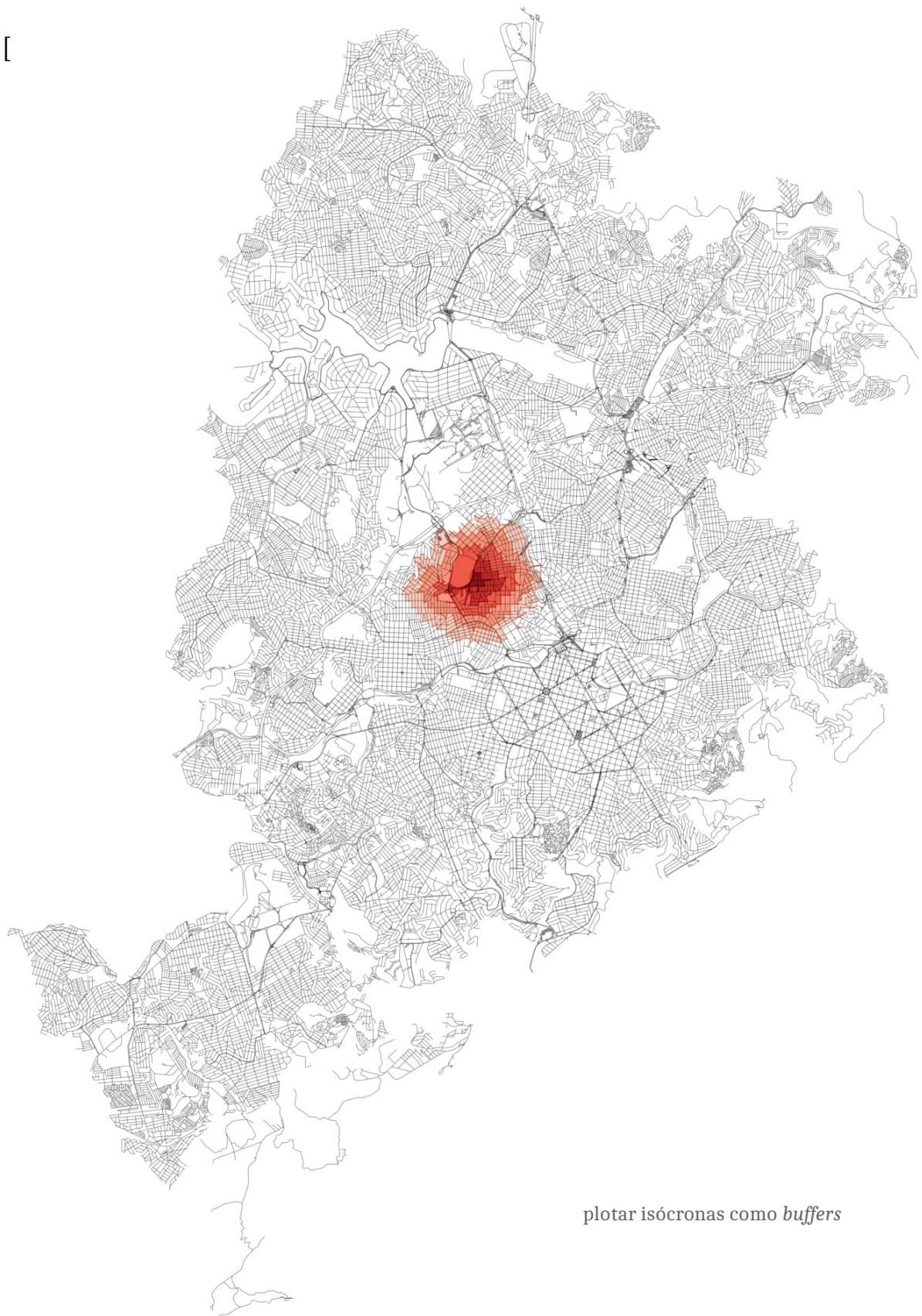
Quão longe você pode andar em 5, 10, 15, 20 e 25 minutos do nó de origem? Usar o NetworkX para induzir um subgrafo de cada distância, com base no tempo de viagem e na velocidade de viagem.

out: [



] colorir os nós de acordo com a isócrona
e traçar a rede de ruas

out: [



] plotar isócronas como *buffers*

qgis + ors tools

OSMnx é um pacote Python que permite baixar geometrias espaciais e modelar, projetar, visualizar e analisar redes de ruas do mundo real a partir de APIs do OpenStreetMap. Pode-se trabalhar facilmente com amenidades / pontos de interesse, projeções de edifícios, dados de elevação, posições / orientações de ruas, velocidade / tempo de viagem e rotas de rede.

Problema: Algoritmo genético criado para reproduzir a frase “cargo DAM-4”. Para isso, o algoritmo precisa de um conjunto de ‘genes’ para usar na construção de palpites (geneSet). Também precisa de um objetivo (target). Isso parece bobagem, se conhecemos o resultado desejado, por que escrever um algoritmo? A resposta é simples: este é um exercício.

Referência: Geoff Boeing.

Ambiente: qgis

qgis + ors tools

OSMnx é um pacote Python que permite baixar geometrias espaciais e modelar, projetar, visualizar e analisar redes de ruas do mundo real a partir de APIs do OpenStreetMap. Pode-se trabalhar facilmente com amenidades / pontos de interesse, projeções de edifícios, dados de elevação, posições / orientações de ruas, velocidade / tempo de viagem e rotas de rede.

Problema: Algoritmo genético criado para reproduzir a frase “cargo DAM-4”. Para isso, o algoritmo precisa de um conjunto de ‘genes’ para usar na construção de palpites (geneSet). Também precisa de um objetivo (target). Isso parece bobagem, se conhecemos o resultado desejado, por que escrever um algoritmo? A resposta é simples: este é um exercício.

Referência: Geoff Boeing.

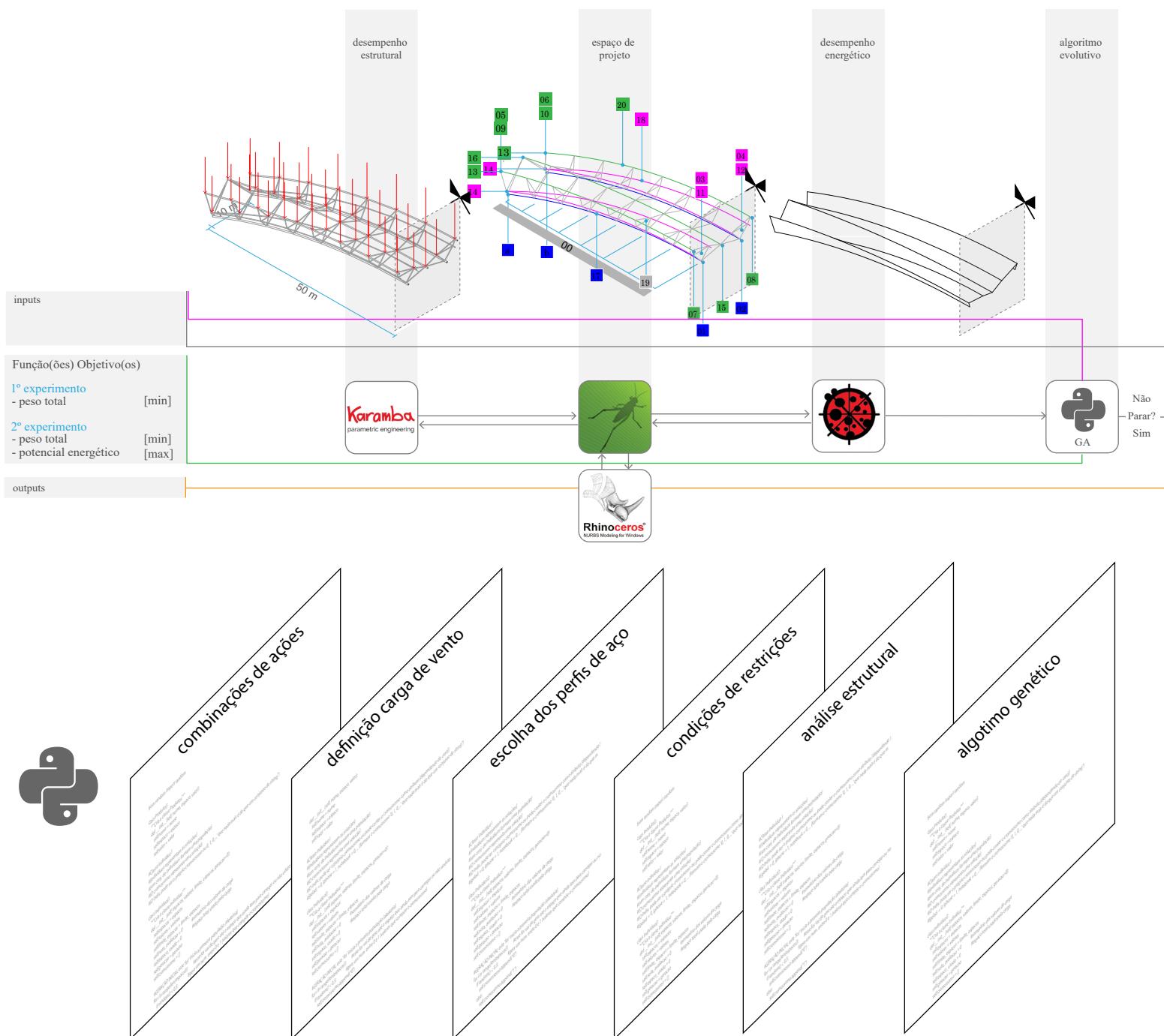
Ambiente: qgis

otimização estrutural

Descrição O presente trabalho descreve a construção de um sistema que combina estratégias de modelagem paramétrica e algoritmos genéticos para otimização do peso total de uma cobertura em estrutura metálica e de sua superfície como potencial área de geração de energia fotovoltaica

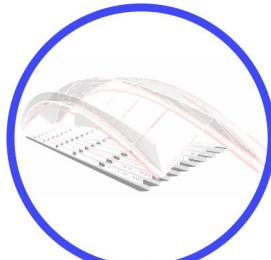
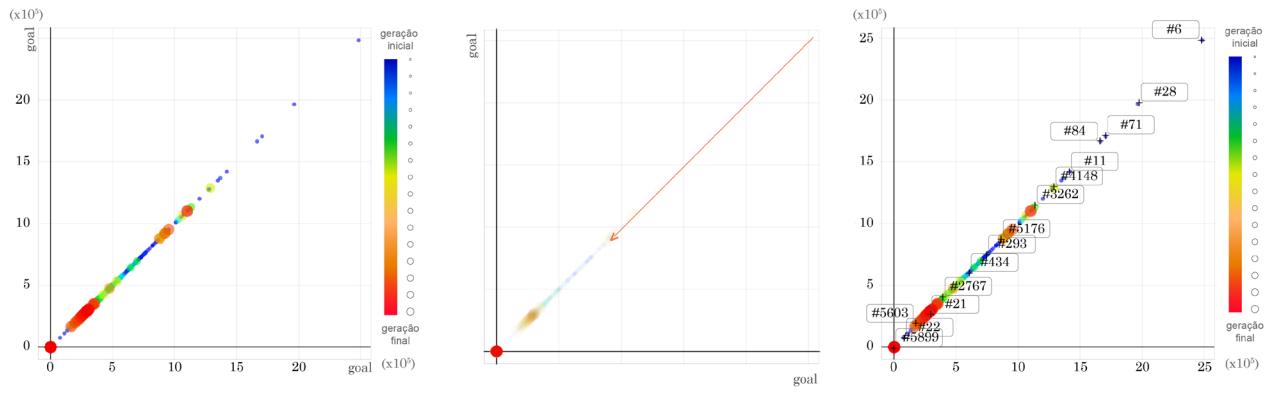
Nos dois experimentos foram gerados um conjunto de 10.800 opções. A tarefa neste estágio foi filtrar os conjuntos de dados pelas pontuações e selecionar as alternativas com maiores desempenhos

Ambiente: grasshopper 3d + lady bug + karamba 3d + python

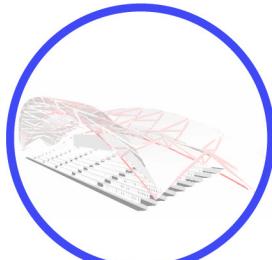


out: [

14



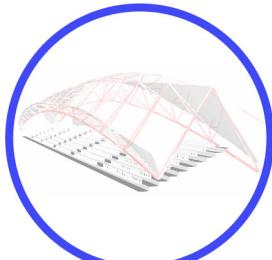
design 6
2483355



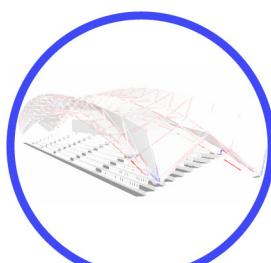
design 11
1420456



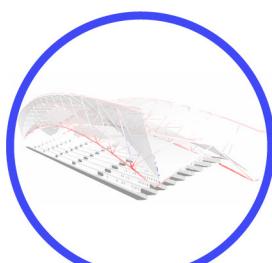
design 21
1199710



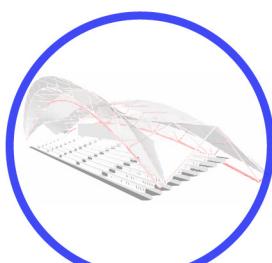
design 22
75539.6



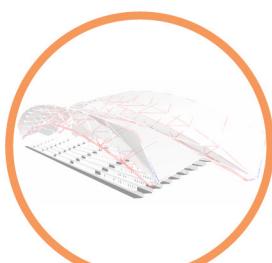
design 28
62.9



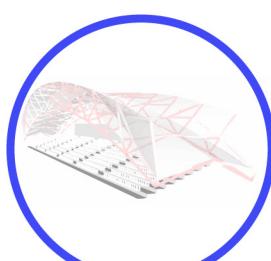
design 71
1704756



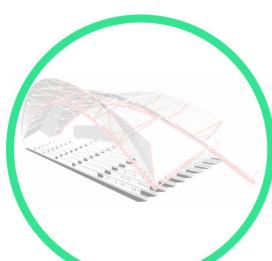
design 84
1664486



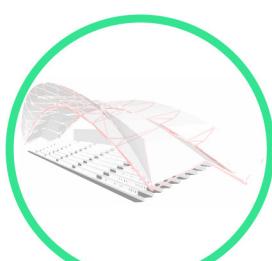
design 293
724964.7



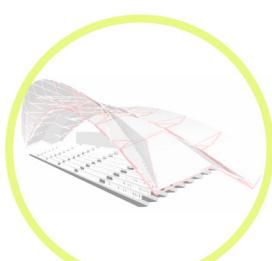
design 434
754851.4



design 2767
393597.4



design 3262
1132236



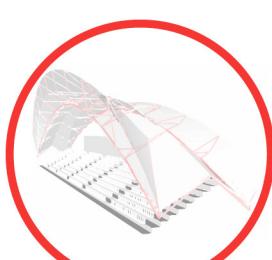
design 4148
1288046



design 5176
878905.2



design 5603
10.0



design 5899
3.44

LEGENDA

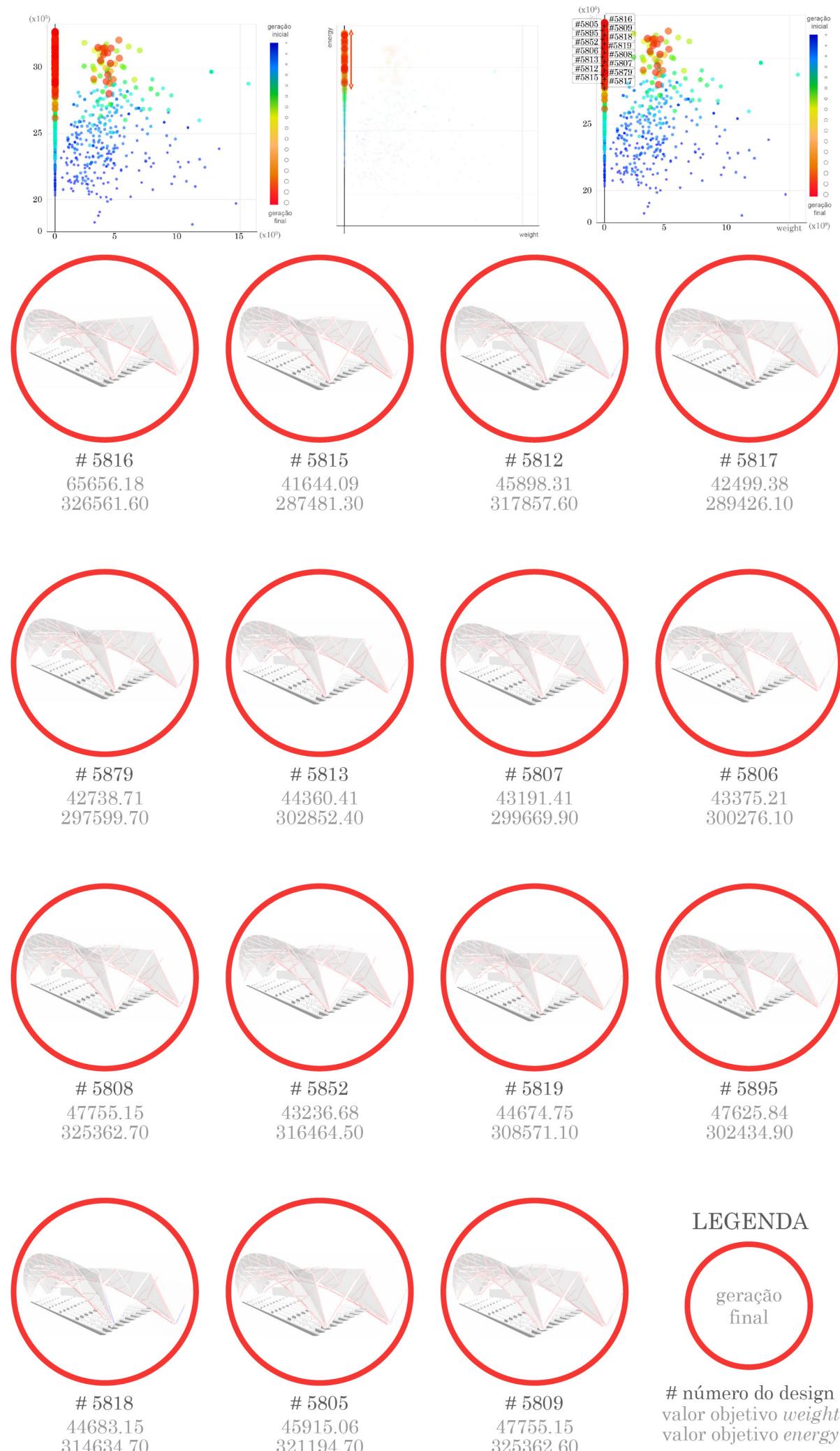


número do design
valor objetivo

]

out: [

15

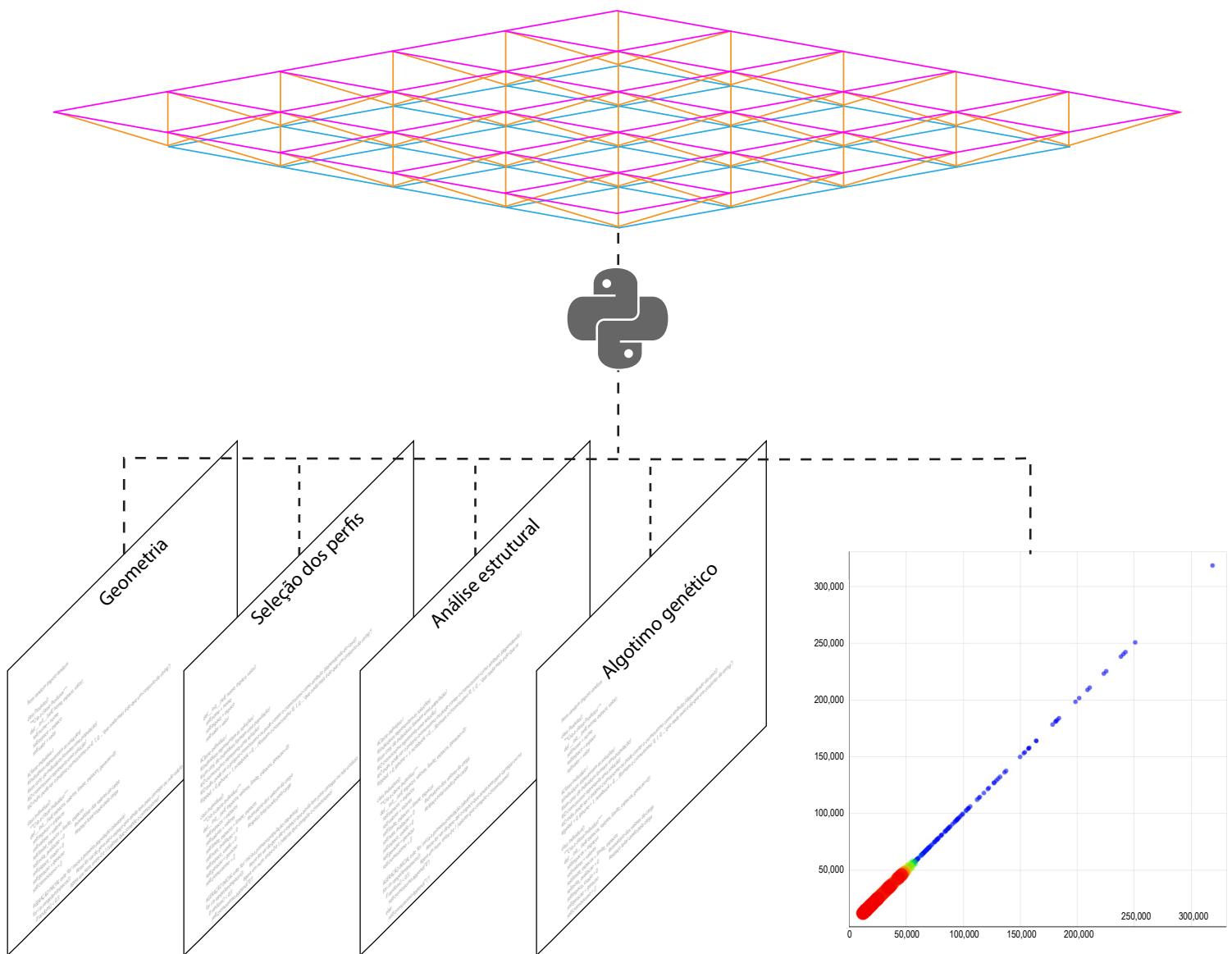


]

interação grasshopper + python

Descrição A maioria dos softwares de modelagem e computação gráfica oferecidos atualmente incorporam linguagens de scripts e uma vez que se tem o domínio sobre esses recursos, podemos transcender quaisquer limitações que o software pode nos impor como designer. Dessa forma, este estudo foca na potência dos algoritmos num exercício de programação onde o modelo, a análise estrutural, a seleção de perfis metálicos e a otimização estrutural foram modelados em python.

Ambiente: grasshopper + python



beagle - algoritmo genético

descrição

Os algoritmos genéticos são uma importante área da Inteligência Artificial que são responsáveis pela resolução de problemas complexos, tendo como base encontrar soluções para problemas de otimização e busca. Existem várias aplicações práticas deste tipo de algoritmo, as quais podem ser aplicadas na resolução de problemas em cenários reais do dia a dia. O objetivo principal deste algoritmo é servir de ferramenta computacional para disseminar a utilização de algoritmo genético para solucionar problemas, ou ainda, automatizar tarefas entre arquitetos e engenheiros projetistas, assim como auxiliar o aprendizado de estudantes na área de design computacional.

Situação: 80% pronto

Autor: Renato Godoi da Cruz

ambiente: grasshopper 3d + python

GA_Beagle.py

```
import rhinoscriptsyntax as rs
from random import random, randint

class Parametro():
    def __init__(self, nome, minimo, maximo): #o valor entra
nesse algoritmo
        self.nome = nome
        self.minimo = minimo
        self.maximo = maximo

class Individuo():
    def __init__(self, minimos, maximos, geracao=0):
        self.minimos = minimos
        self.maximos = maximos
        self.nota_avaliacao = 0
        self.geracao = geracao
        self.cromossomo = []

    for i in range(len(lista_parametros)):
        if (type(minimos[i]) == float) and (type(maximos[i]) ==
float):
            self.cromossomo.append(random.uniform(minimos[i],
maximos[i]))
        else:
            self.cromossomo.append(randint(minimos[i], maxi-
mos[i]))

    def avaliacao(self):
        nota = randint(1000, 2222)
        self.nota_avaliacao = nota

    def crossover(self, outro_individuo):
        corte = int(round(random() * len(self.cromossomo)))
        filho1 = outro_individuo.cromossomo[0:corte] + self.cro-
mossomo[corte::]
        filho2 = self.cromossomo[0:corte] + outro_individuo.cro-
mossomo[corte::]

        filhos = [Individuo(self.minimos, self.maximos, self.geracao
+ 1),
                  Individuo(self.minimos, self.maximos, self.geracao +
```

```
def __init__(self, tamanho_populacao):
    self.tamanho_populacao = tamanho_populacao
    self.populacao = []
    self.geracao = 0
    self.melhor_solucao = 0

def inicializa_populacao(self, minimos, maximos):
    for i in range(self.tamanho_populacao):
        self.populacao.append(Individuo(minimos, maximos))
    self.melhor_solucao = self.populacao[0]

def ordena_populacao(self):
    self.populacao = sorted(self.populacao,
                           key = lambda populacao: populacao.nota_aval-
iacao,
                           reverse = True)

def melhor_individuo(self, individuo):
    if individuo.nota_avaliacao > self.melhor_solucao.nota_aval-
iacao:
        self.melhor_solucao = individuo

def soma_avaliacoes(self):
    soma = 0
    for individuo in self.populacao:
        soma += individuo.nota_avaliacao
    return soma

def seleciona_pai(self, soma_avaliacao):
    pai = -1
    valor_sorteado = random() * soma_avaliacao
    soma = 0
    i = 0
    while i < len(self.populacao) and soma < valor_sorteado:
        soma += self.populacao[i].nota_avaliacao
        pai += 1
        i += 1
    return pai

def visualiza_geracao(self): #DADOS DO GA(raspar para uma
planilha Excel)
    melhor = self.populacao[0]
    [...]
```

data-capture

descrição:

Algoritmo para capturar dados produzidos durante o processo de otimização GA. Esse recurso funciona capturando viewports Rhino (.png) e dados usados como 'genomas' e 'valor_fitness' durante o processo de otimização de Galápagos em um arquivo .CSV, atribuindo seu nome e localização da mesma maneira descrita para as imagens.

Este algoritmo faz parte do projeto Beagle e deverá servir para capturar os dados de seus processos de otimização

ambiente: grasshopper 3d + python

_scraper.py

```
import scriptcontext as sc
import Rhino as rc
import os
import System

def checkOrMakeFolder():

    if ghdoc.Path:
        folder = os.path.dirname(ghdoc.Path)
        ghDef = ghenv.LocalScope.ghdoc.Name.strip("*")
        captureFolder = folder + "\\\" + str(ghDef)
        if not os.path.isdir(captureFolder):
            os.makedirs(captureFolder)
        return captureFolder

def makeFileName():

    fileName = str(goalValue)
    return fileName

def captureActiveViewToFile(width,height,path):

    sc.doc = rc.RhinoDoc.ActiveDoc
    activeView = sc.doc.Views.ActiveView
    imageDim = System.Drawing.Size(width,height)
    try:
        imageCap = rc.Display.RhinoView.CaptureToBitmap(activeView,imageDim)
        System.Drawing.Bitmap.Save(imageCap,path)
        rc.RhinoApp.WriteLine(path)
        return path
    except:
        raise Exception(" Capture failed, check the path")

if Toggle:
    capFolder = checkOrMakeFolder()
```

```
fileText = os.path.join(capFolder + ".csv")
with open(fileText, 'a') as file_object:
    fileName = makeFileName()
    count = 0
    try:
        path = os.path.join(capFolder,fileName + ".png")
        Path = captureActiveViewToFile(width,height,path)
    except:
        raise Exception("Capture failed, save the GH definition")
    converte = ', '.join(map(str, data))
    concatenatedData = converte + ", " + fileName + "\n"
    file_object.write(concatenatedData)
```

out:[



.png



]

design code

Design Computacional

programação como meio de projeto e métodos criativas relacionadas modelagem geométrica de projeto e modelagem paramétrica

ambiente: grasshopper + python

truss_tensegrity.py

```
import rhinoscriptsyntax as rs

list = [[0,0,0],[y,0,x],[y,0,-x],[y+4,0,x],[y+4,0,-x],[y+6,0,0]]

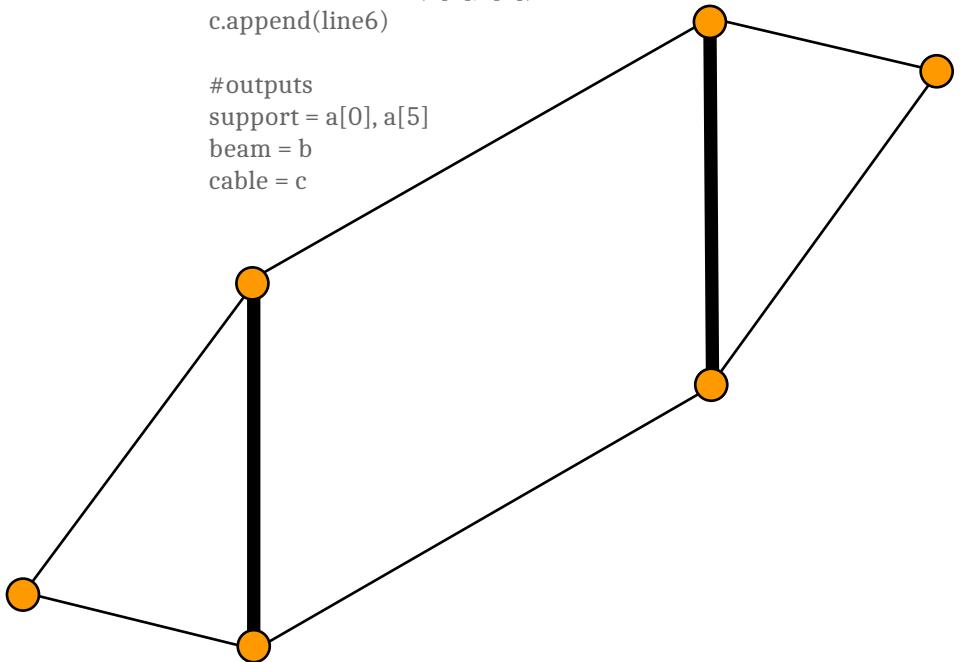
cont = 0
a = []
for i in range(6):
    points = rs.AddPoint(list[cont])
    a.append(points)
    cont += 1

cont = 1
b = []
for i in range(2):
    bars = rs.AddLine(a[cont], a[cont + 1])
    b.append(bars)
    cont += 2

c = []
line1 = rs.AddLine(a[0],a[1])
c.append(line1)
line2 = rs.AddLine(a[1],a[3])
c.append(line2)
line3 = rs.AddLine(a[3],a[5])
c.append(line3)
```

```
line4 = rs.AddLine(a[5],a[4])
c.append(line4)
line5 = rs.AddLine(a[4],a[2])
c.append(line5)
line6 = rs.AddLine(a[2],a[0])
c.append(line6)

#outputs
support = a[0], a[5]
beam = b
cable = c
```



tensegrity_3.py

```

import rhinoscriptsyntax as rs
import Rhino as rh
import math

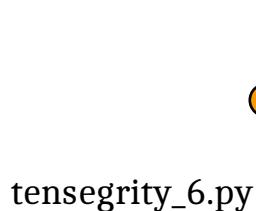
pointList = []
curAngle = 0
delAngle = 360/numberSides

print('Tensegrity simple')

while (curAngle <= 360):
    x = centerPt.X + math.cos(math.radians(curAngle)) * radius
    y = centerPt.Y + math.sin(math.radians(curAngle)) * radius
    z = centerPt.Z
    pt = rh.Geometry.Point3d(x, y, z)
    pt2 = rh.Geometry.Point3d(x, y, altura)
    curAngle += delAngle
    pointList.append(pt)
    pointList.append(pt2)

bars = []
numPoint = (len(pointList) - 2)
cont = 0

```



tensegrity_6.py

```

import rhinoscriptsyntax as rs
import Rhino as rh
import math

pointList = []
curAngle = 0
delAngle = 360/numberSides

print('Tensegrity simple')

while (curAngle <= 360):
    x = centerPt.X + math.cos(math.radians(curAngle)) * radius
    y = centerPt.Y + math.sin(math.radians(curAngle)) * radius
    z = centerPt.Z
    pt = rh.Geometry.Point3d(x, y, z)
    pt2 = rh.Geometry.Point3d(x, y, altura)
    curAngle += delAngle
    pointList.append(pt)
    pointList.append(pt2)

a = pointList
print(a)

bar1 = rs.AddLine(pointList[0], pointList[9])
bar2 = rs.AddLine(pointList[2], pointList[11])

```

```

while cont < numPoint:
    bar = rs.AddLine(pointList[cont], pointList[cont + 3])
    bars.append(bar)
    cont += 2

```

```

lineV = []
numPoint2 = (len(pointList) - 2)
cont2 = 0

```

```

while cont2 < numPoint2:
    lv = rs.AddLine(pointList[cont2], pointList[cont2 + 1])
    lineV.append(lv)
    cont2 += 2

```

```

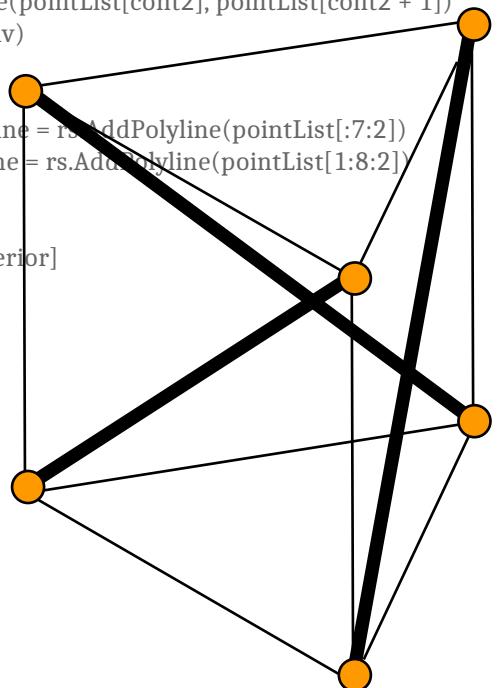
superior = polyline = rs.AddPolyline(pointList[7:2])
inferior = polyline = rs.AddPolyline(pointList[1:8:2])

```

```

b = bars
c = [superior,inferior]
d = lineV

```



```

bar3 = rs.AddLine(pointList[4], pointList[1])
bar4 = rs.AddLine(pointList[6], pointList[3])
bar5 = rs.AddLine(pointList[8], pointList[5])
bar6 = rs.AddLine(pointList[10], pointList[7])

```

```

interstitials = []
numPoint = (len(pointList) - 2)
cont = 0

```

```

while cont < numPoint:
    interstitial = rs.AddLine(pointList[cont], pointList[cont + 3])
    interstitials.append(interstitial )
    cont += 2

```

```

inferior = polyline = rs.AddPolyline(pointList[:13:2])
superior = polyline = rs.AddPolyline(pointList[1:14:2])

```

```

a = interstitials
b = [bar1, bar2, bar3, bar4, bar5, bar6]
c = pointList
d = [superior, inferior]

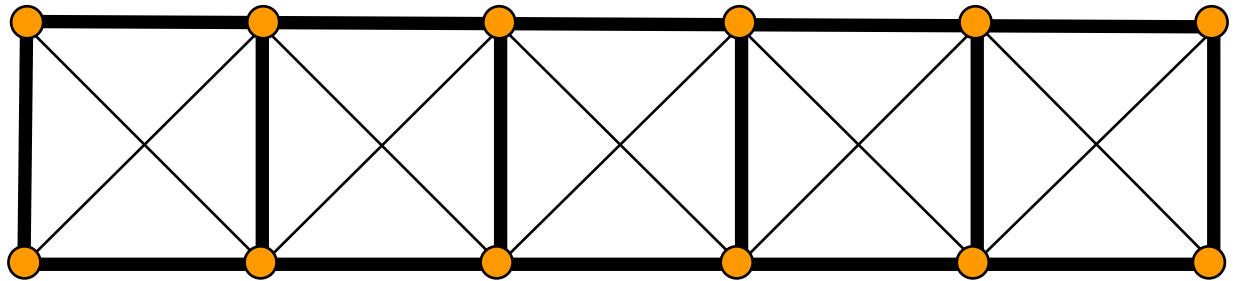
```

tensegrity_structures

Estudos de estruturas de autotensão

Estudos de estruturas autotensionáveis modelados em python compostas por estruturas rígidas e cabos, com forças de tração e compressão, que formam um todo integrado.

Ambiente: grasshopper + python



simple_truss.py

Drawing a simple trellis.

Environment - rhinoceros + grasshopper

Language - python

What is a simple trellis?

In structural engineering, a lattice is a structure composed of triangular units constructed with straight elements whose ends are connected at points known as knots. External forces and reactions are considered, in a simplified way, applied in these same nodes.

INPUTS

spacing - Item acess {type him = ghdoc rhinoscriptsyntax};
zNum - Item acess {type him = ghdoc rhinoscriptsyntax};
yNum - Item acess {type him = ghdoc rhinoscriptsyntax}.

OUTPUTS

out - the execution information;
bars - output bars {boxes};
braces - outputs braces {diagonals};
supports - outputs supports {point of connections};
load - outputs load {point of load}.

Reference - Danil NAGY

```
import Rhino.Geometry as rh
```

```
points = []
```

```
for y in range(int(yNum+1)):  
    points.append([])
```

```
for y in range(int(yNum+1)):  
    points.append([])  
    for z in range(int(zNum+1)):  
        points[-1].append(rh.Point3d(0, y*spacing, z*spacing))  
  
bars = []  
braces = []  
  
for i in range(len(points)):  
    for j in range(len(points[i])):  
        if i < len(points) - 1:  
            bars.append(rh.Line(points[i][j], points[i+1][j]))  
  
        if j < len(points[i]) - 1:  
            bars.append(rh.Line(points[i][j], points[i][j+1]))  
  
        if i < len(points) - 1 and j < len(points[i]) - 1:  
            braces.append(rh.Line(points[i][j], points[i+1][j+1]))  
  
        if i > 0 and j < len(points[i]) - 1:  
            braces.append(rh.Line(points[i][j], points[i-1][j+1]))  
  
supports = points[:][0]  
load = points[-1][-1]
```

truss_simple_2.py

```
import rhinoscriptsyntax as rs
import math
```

```
start = 0,0,0
end = (length, 0,0)
pt1 = rs.coerce3dpoint(start)
pt2 = rs.coerce3dpoint(end)
```

```
span = rs.Distance(pt1, pt2)
segments = int(segments)
length_increment = span/segments
```

```
slope_perc = slope
slope_ang = math.atan(slope/100)
```

```
height_increment = (span/segments)*math.tan(slope_ang)
n = height
height_step = height_increment
length_step = length_increment
```

```
a = []
b = []
c = []
d = []
e = []
```

```
slope_perc = slope
slope_ang = math.atan(slope/100)
```

```
j = 0
for i in range(segments+1):
    j = j + length_step
    bot = rs.AddPoint(j-length_step,0,0)
    b.append(bot)
```

```
j = 0
for i in range(segments+1):
    if i <= segments/2:
        n = n + height_step
        j = j + length_step
        top = rs.AddPoint(j - length_step,0,n - height_step)
    else:
        n = n - height_step
        top = rs.AddPoint(j,0,n - height_step)
        j = j + length_step
    a.append(top)
```

```
pt_top = a
pt_bot = b
j = 0
for i in range(segments):
    diagonal = rs.AddLine(pt_bot[j],pt_top[j+1])
    j = j + 1
    c.append(diagonal)
```

```
j = 0
for i in range(segments):
    diagonal = rs.AddLine(pt_bot[j+1],pt_top[j])
    j = j + 1
    d.append(diagonal)
```

```
bot_ch = rs.AddLine(pt_bot[0],pt_bot[segments])
top_ch = [rs.AddLine(pt_top[0],pt_top[int(segments/2)]),
          rs.AddLine(pt_top[int(segments/2)],pt_top[segments])]
```

```
#generation of verticals
j = 0
for i in range(segments+1):
    vertical = rs.AddLine(pt_bot[j],pt_top[j])
    j = j + 1
    e.append(vertical)
```

```
#assigning variables
dia_1 = c
dia_2 = d
vert = e
```

```
#Warren
if truss_type == 0:
    if segments/2 % 2 == 1:
        diagonals = dia_2[slice(1, int(segments/2),2)] +
                    dia_1[slice(int(segments/2+1), int(segments),2)]
        diagonals = dia_1[slice(0, int(segments/2),2)] +
                    dia_2[slice(int(segments/2), int(segments),2)]
    else:
        diagonals = dia_2[slice(1, int(segments/2),2)] +
                    dia_1[slice(int(segments/2), int(segments),2)]
        diagonals = dia_1[slice(0, int(segments/2),2)] +
                    dia_2[slice(int(segments/2+1), int(segments),2)]
#Warren Flipped
if truss_type == 1:
    if segments/2 % 2 == 1:
        diagonals = dia_2[slice(0, int(segments/2),2)] +
                    dia_1[slice(int(segments/2), int(segments),2)]
        diagonals = dia_1[slice(1, int(segments/2),2)] +
                    dia_2[slice(int(segments/2+1), int(segments),2)]
    else:
        diagonals = dia_2[slice(0, int(segments/2),2)] +
                    dia_1[slice(int(segments/2+1), int(segments),2)]
        diagonals = dia_1[slice(1, int(segments/2),2)] +
                    dia_2[slice(int(segments/2), int(segments),2)]
#Howe
if truss_type == 2:
    if edge_diagonals < segments/2:
        diagonals = dia_1[slice(0, int(edge_diagonals))] +
                    dia_2[slice(int(segments-edge_diagonals), int(segments))]
        diagonals = dia_2[slice(0, int(edge_diagonals))] +
                    dia_1[slice(int(segments-edge_diagonals), int(segments))]
    else:
        diagonals = dia_1[slice(0, int(segments/2))] +
                    dia_2[slice(int(segments/2), int(segments))]
#Pratt
if truss_type == 3:
    if edge_diagonals < segments/2:
        diagonals = dia_2[slice(0, int(edge_diagonals))] +
                    dia_1[slice(int(segments-edge_diagonals), int(segments))]
        diagonals = dia_2[slice(int(edge_diagonals), int(segments/2))] +
                    dia_1[slice(int(segments/2), int(segments-edge_diagonals))]
    else:
        diagonals = dia_2[slice(0, int(segments/2))] +
                    dia_1[slice(int(segments/2), int(segments))]
```

space_truss.py

"""

Drawing a space truss

Environment - rhinoceros + grasshopper

Language - python

What is a space truss?

See here: https://en.wikipedia.org/wiki/Space_frame

INPUTS

side - Item access {type hint = float};

index - Item access {type hint = int};

xNum - Item access {type hint = int};

yNum - Item access {type hint = int};

height - Item access {type hint = float};

OUTPUTS

out - the execution information;

 0 - Side dimension: {float}

 1 - Number of modules: {int/unit}

 2 - Modular dimensions: {float}

upper - outputs upper bars;

below - outputs below bars;

diagonal - outputs diagonals bars;

support - outputs support points;

a - outputs upper points.

Autor - Renato Godoi da Cruz

email - renatogcruz@hotmail.com

"""

```
import Rhino.Geometry as rh
```

```
#size [variable] * num [variable] == side [immutable]
```

```
x = side / index
```

```
a = side
```

```
size = x
```

```
#points
```

```
points = []
```

```
#carga = [] #PROVISÓRIO
```

```
for y in range (int(yNum + 1)):
```

```
    points.append ([])
```

```
    for x in range (int(xNum +1)):
```

```
        points[-1].append (rh.Point3d(x * size,y * size, height))
```

```
#    carga.append((x * size,y * size, height)) #PROVISÓRIO
```

adiciona pontos de cargas

```
points2 = []
```

```
for y in range (int(yNum)):
```

```
    points2.append ([])
```

```
    for x in range (int(xNum)):
```

```
        points2[-1].append (rh.Point3d(x * size/2,y * size/2, height/2))
```

```
size/2, 0))
```

```
below = []
```

```
for i in range(len(points2)):
```

```
    for j in range(len(points2[i])):
```

```
        if i < len(points2) - 1:
```

```
            below.append(rh.Line(points2[i][j], points2[i+1][j]))
```

```
        if j < len(points2[i]) - 1:
```

```
            below.append(rh.Line(points2[i][j], points2[i][j+1])))
```

```
upper = []
```

```
for i in range(len(points)):
```

```
    for j in range(len(points[i])):
```

```
        if i < len(points) - 1:
```

```
            upper.append(rh.Line(points[i][j], points[i+1][j]))
```

```
        if j < len(points[i]) - 1:
```

```
            upper.append(rh.Line(points[i][j], points[i][j+1])))
```

```
diagonal = []
```

```
for i in range (len(points2)):
```

```
    for j in range (len(points)):
```

```
        if i < len(points2) and j < len(points) - 1 :
```

```
            diagonal.append(rh.Line(points2[i][j], points[i][j])))
```

```
if i >= 0 and j < len(points)-1 :
```

```
    diagonal.append(rh.Line(points2[i][j], points[i+1][j])))
```

```
if i >= 0 and j < len(points)-1 :
```

```
    diagonal.append(rh.Line(points2[i][j], points[i+1][j+1])))
```

```
support = (points2[0][0], points2 [0][-1], points2[-1][0],  
points2[-1][-1])
```

```
a = points
```

```
print ('Side dimension: %.2f m' %(size * xNum))
```

```
numModules = index * index
```

```
print ('Number of modules: %s unid.' %numModules)
```

```
print ('Modular dimensions: %.2f m' %x)
```

