Optimización topológica en el diseño estructural mediante entornos de programación visual y algoritmos genéticos.

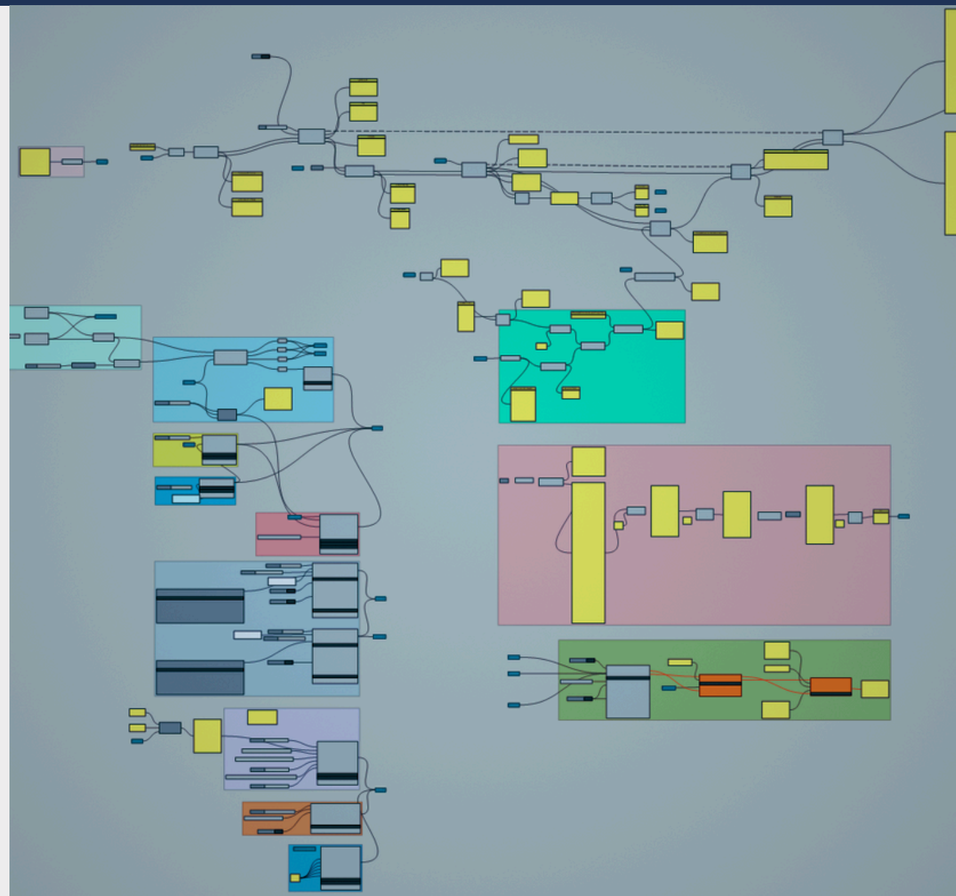
Trabajo de Fin de Grado.

Presentado por:
Sara Trinidad Quiñones

Tutor:
Fernando Medina Reguera

Universidad LOYOLA

ÍNDICE

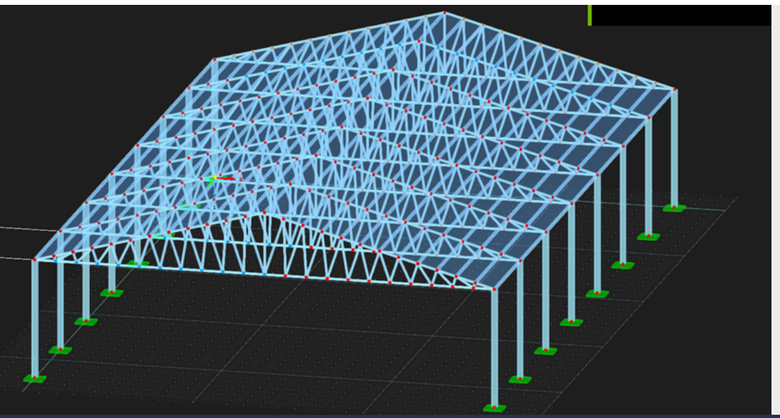


Contexto y Motivación

- Innovación continua y eficiencia.
- ¿Podemos realmente ser participes de esta evolución?
- Intereses propios:
  - Diseño estructural, automatización de procesos, programación dinámica y visual, entornos gráficos.
- Gran potencial de Grasshopper.
- Facilitar el trabajo de otros.

INTRODUCCIÓN

Planteamiento del problema

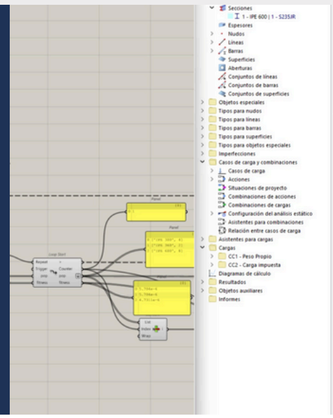


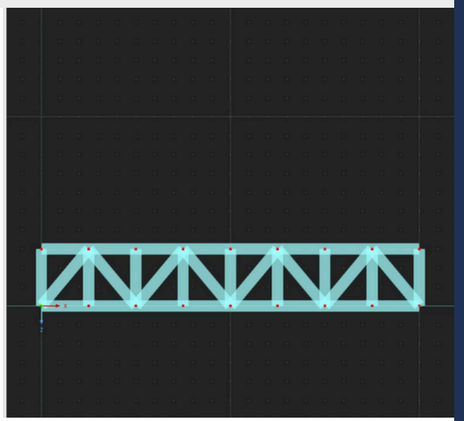
Problema

El diseño y cálculo de estructuras como lo conocemos consume tiempo y no siempre garantiza la mejor solución.

Solución presentada

Automatizar el diseño estructural con algoritmos genéticos y Grasshopper para optimizar la eficiencia y precisión.





Imágenes propias

INTRODUCCIÓN

Objetivos del proyecto

Comprender los fundamentos de los AG y programación visual.	Integrar 3 softwares distintos: RFEM, Grasshopper y Python.	Adecuada implementación de Algoritmos genéticos.
Automatizar el diseño estructural de vigas simples y celosías.	Validar la precisión y eficiencia de la aplicación.	Fomentar la innovación en este campo.

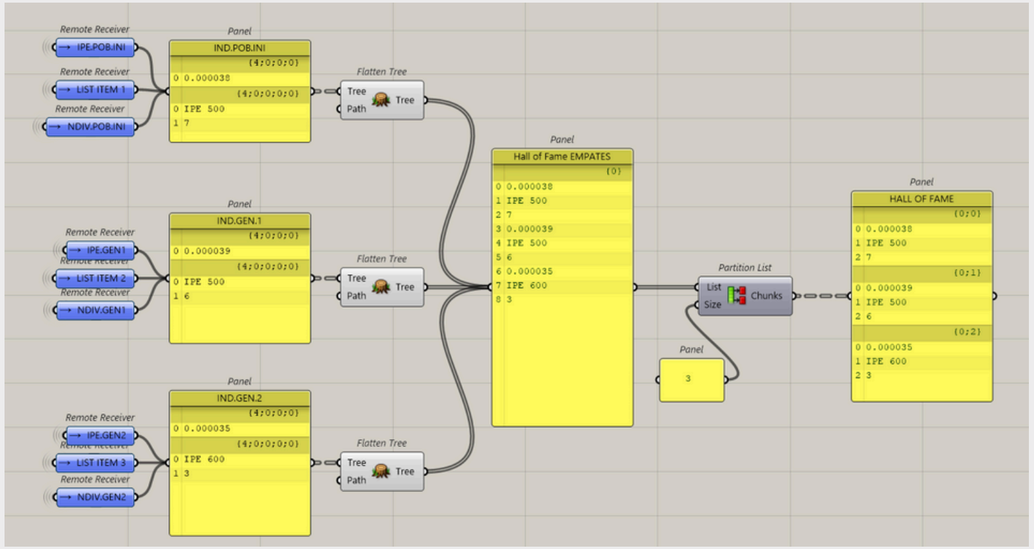
Universidad LOYOLA

SOFTWARES

Grasshopper

Es un lenguaje de programación visual integrado en Rhino para el modelado paramétrico.

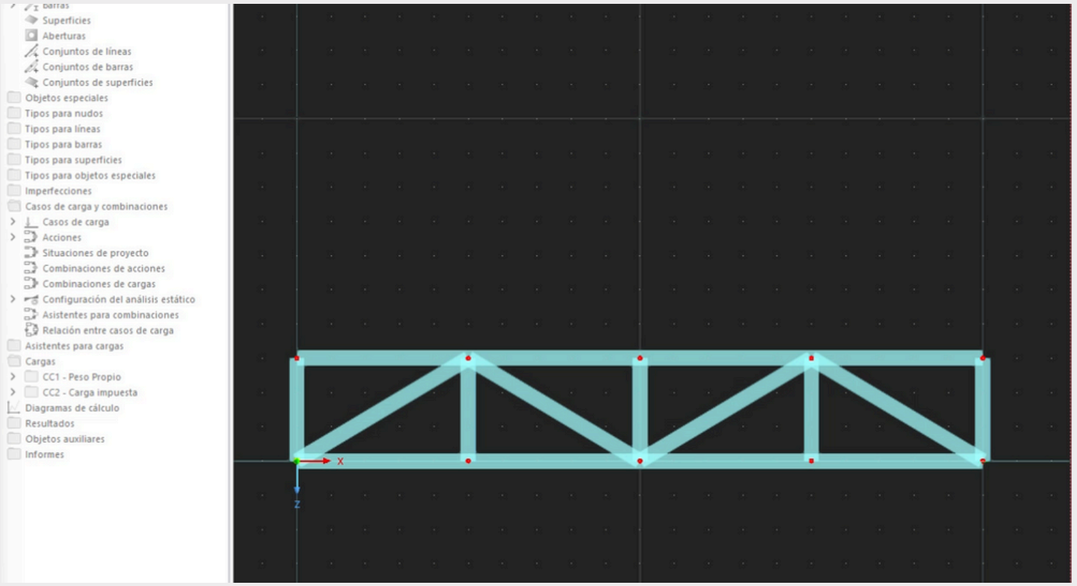
- Entorno gráfico intuitivo.
- Capacidad de modelado paramétrico.
- Integración con otros softwares gracias a numerosos Plug-ins.
- Uso industrial amplio.
- Evolución continua y gran potencial.
- Soporte comunitario activo.



SOFTWARES

Dlubal RFEM

Herramienta de análisis de elementos finitos para modelado y cálculo estructural.



- Análisis preciso de deformaciones.
- Cálculos paralelos mejorados.
- Visualización clara de esfuerzos internos.
- Ofrece diagramas de resultados configurables.
- Ajuste personalizado de unidades.
- Fórmulas de diseño referenciadas.

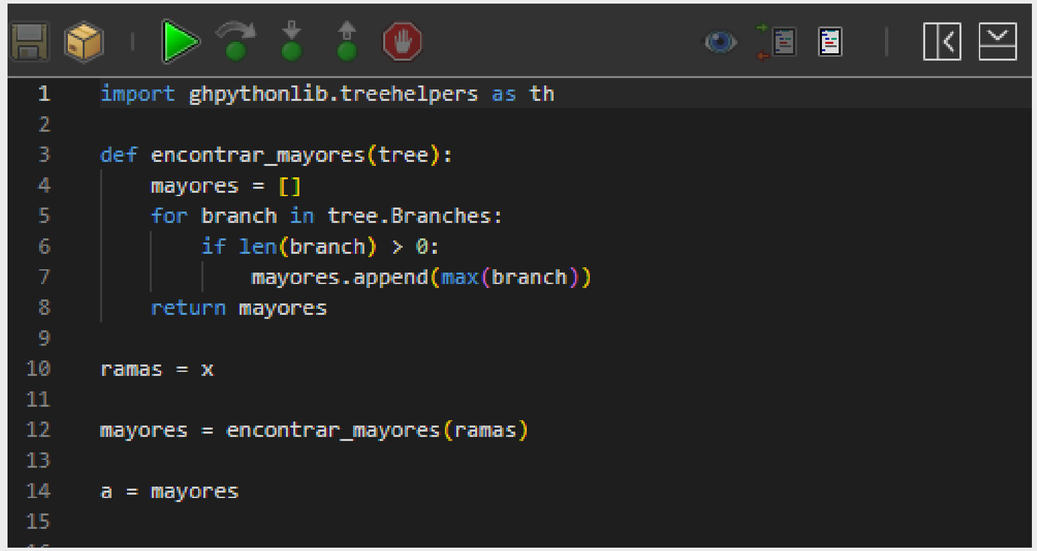
Universidad LOYOLA

SOFTWARES

Python

Es un lenguaje de programación avanzado y versátil, creado en 1990, usado en multitud de campos.

- Sintaxis clara y concisa.
- Código abierto y comunidad activa.
- Integración con otros lenguajes y sistemas.
- Amplia gama de aplicaciones industriales.
- Automatización y control de sistemas.
- Optimización y algoritmos genéticos.



```python
import ghpythonlib.treehelpers as th

def encontrar_mayores(tree):
    mayores = []
    for branch in tree.Branches:
        if len(branch) > 0:
            mayores.append(max(branch))
    return mayores

ramas = x

mayores = encontrar_mayores(ramas)

a = mayores

```

TÉCNICAS

Conexión API

Reglas y procedimientos que permiten la intercomunicación entre diferentes softwares, facilitando el intercambio de información.

Grasshoper y Python:

- Personalización de procesos en diseño paramétrico.
- Aplicación de los operadores genéticos.

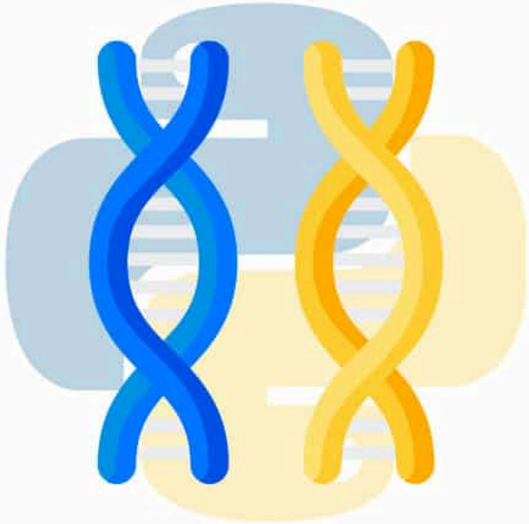
Grasshoper y RFEM:

- Automatización del cálculo de estructuras.
- Importación/exportación de datos esenciales.

TÉCNICAS

Algoritmos genéticos

Modelos de optimización basados en la teoría de la evolución y selección natural.



Términos principales

- Individuo
- Población inicial
- Generación
- Fitness
- Gen

Principales operadores genéticos

Selección

Elegir los individuos para reproducirse.

Cruce

Combinar genes de padres para crear descendencia.

Mutación

Introducir cambios aleatorios en los genes.

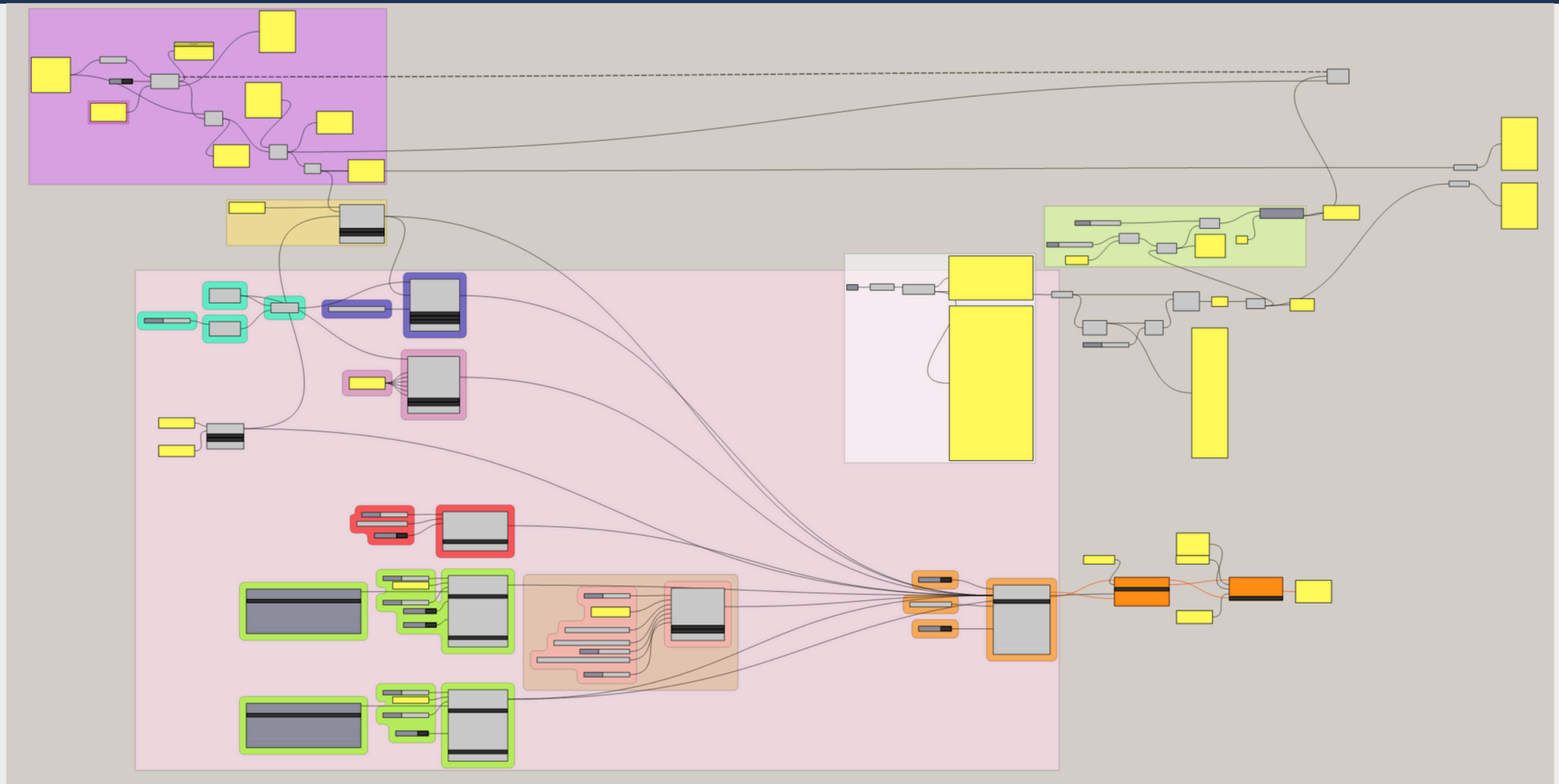
Elitismo

Asegurar la permanencia de los mejores individuos en las próximas generaciones.

CASO 1

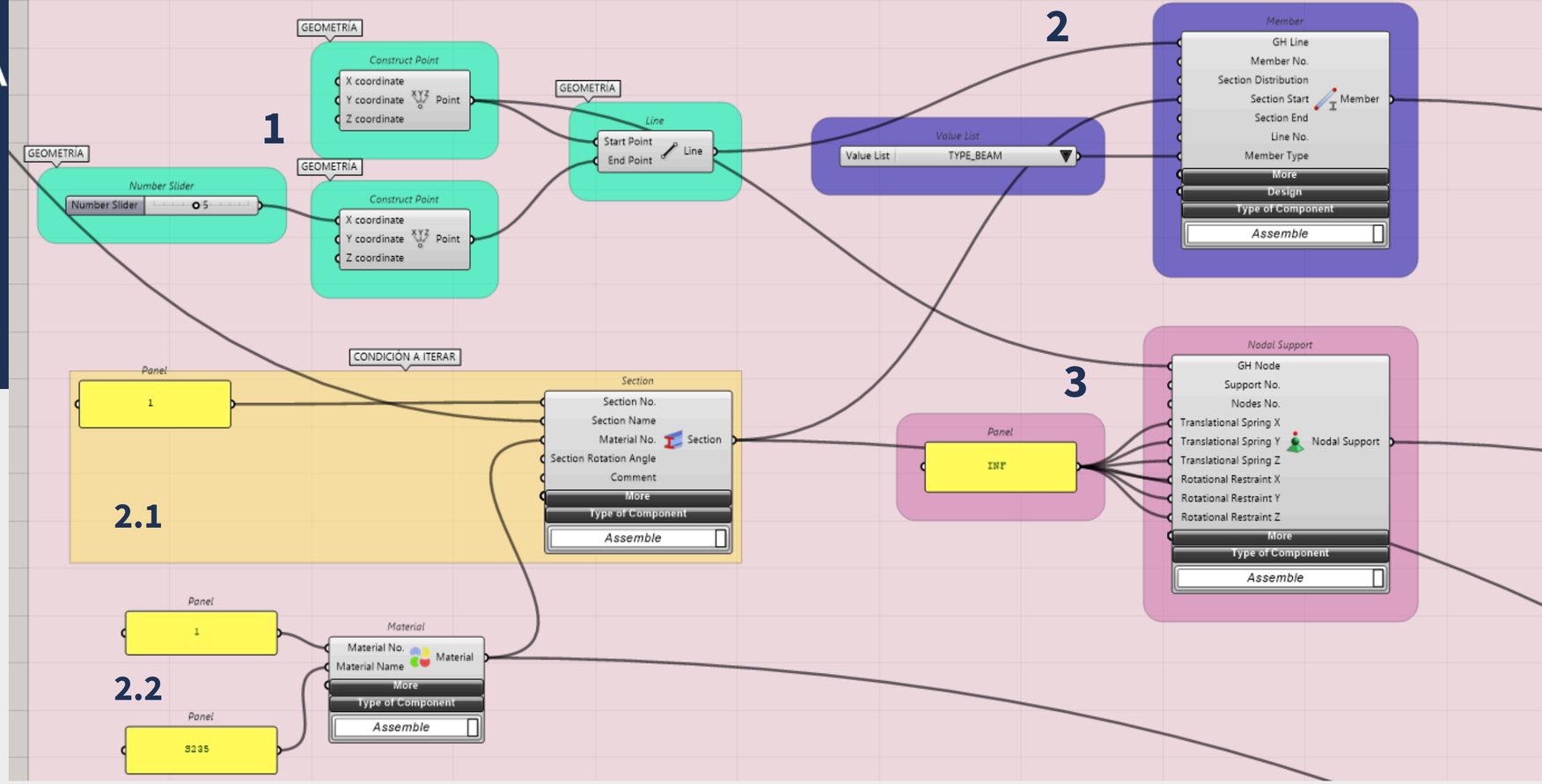
Viga simple

El objetivo es desarrollar un modelo para analizar la deformación de una viga en 2D bajo carga distribuida, iterando entre perfiles IPE hasta cumplir con un límite de deformación nodal.



CASO 1

Viga simple



1 **Geometría básica**

2 **Miembro estructural Viga**

2.1 Sección Inicial

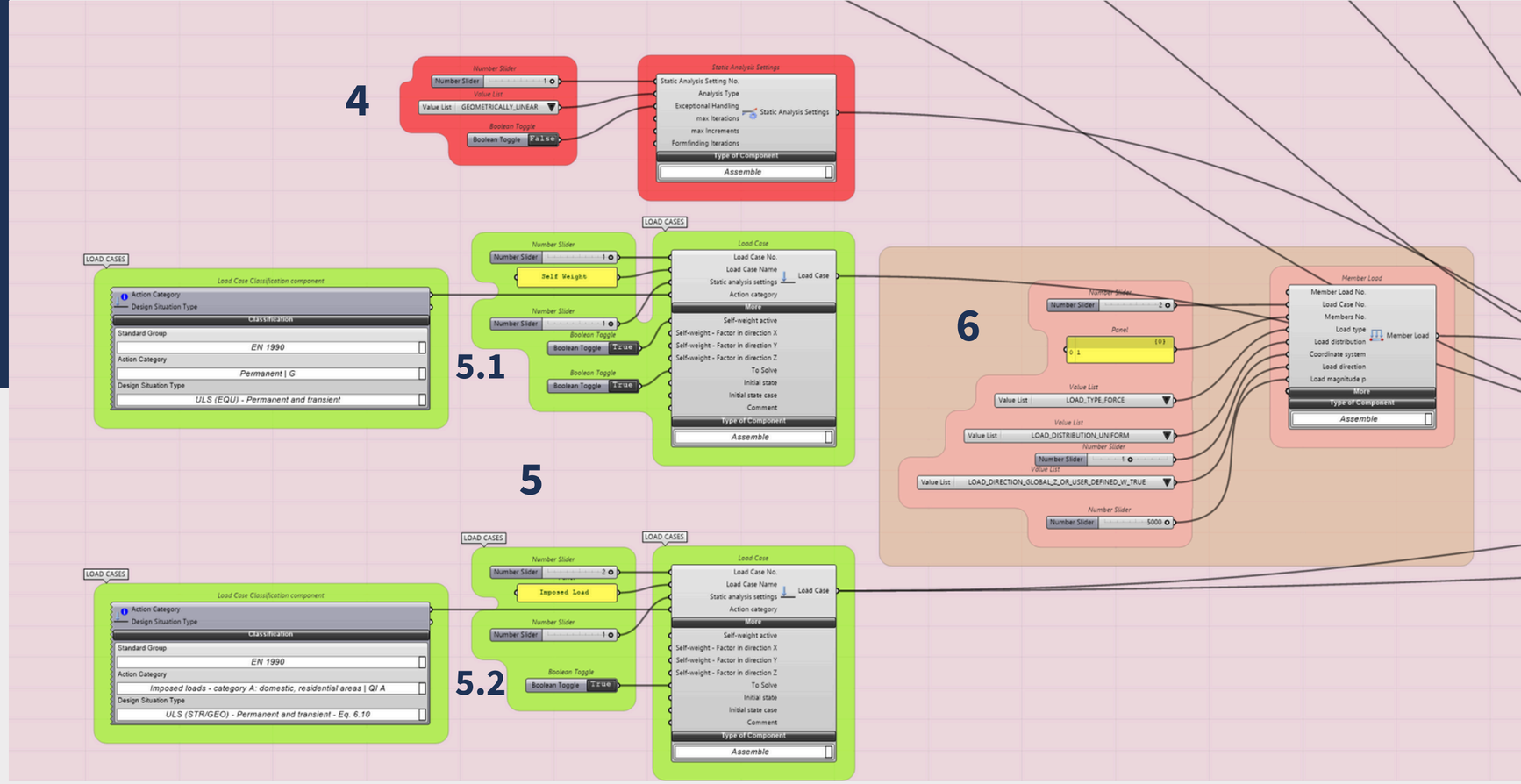
2.2 Material

3 **Apoyos nodales**

Universidad LOYOLA

CASO 1

Viga simple



4 Ajustes de análisis estático

5 Casos de carga

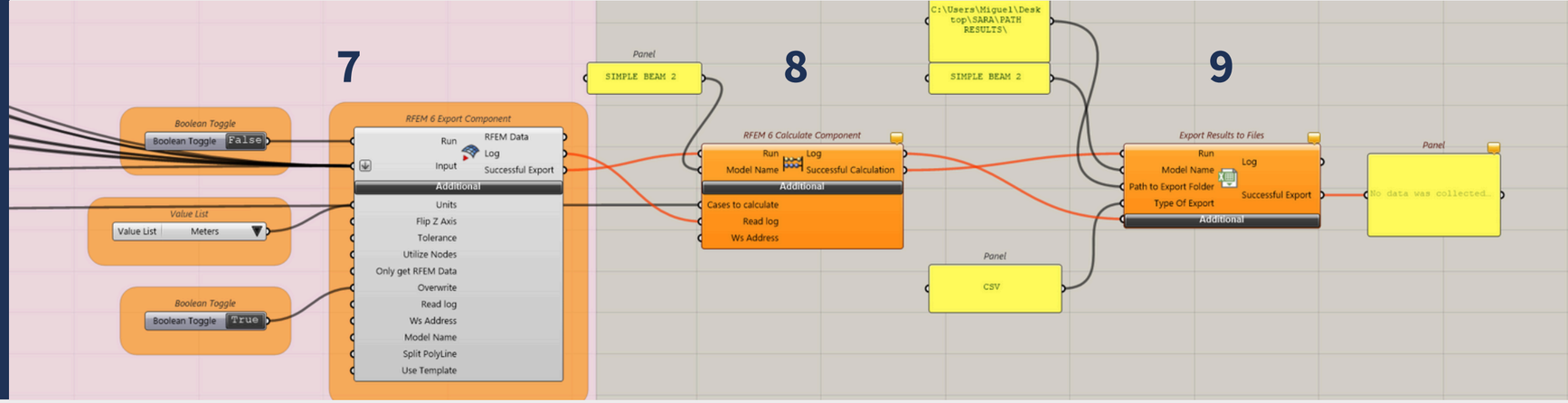
5.1 Peso propio

5.2 Carga impuesta

6 Componente de carga en el miembro

CASO 1

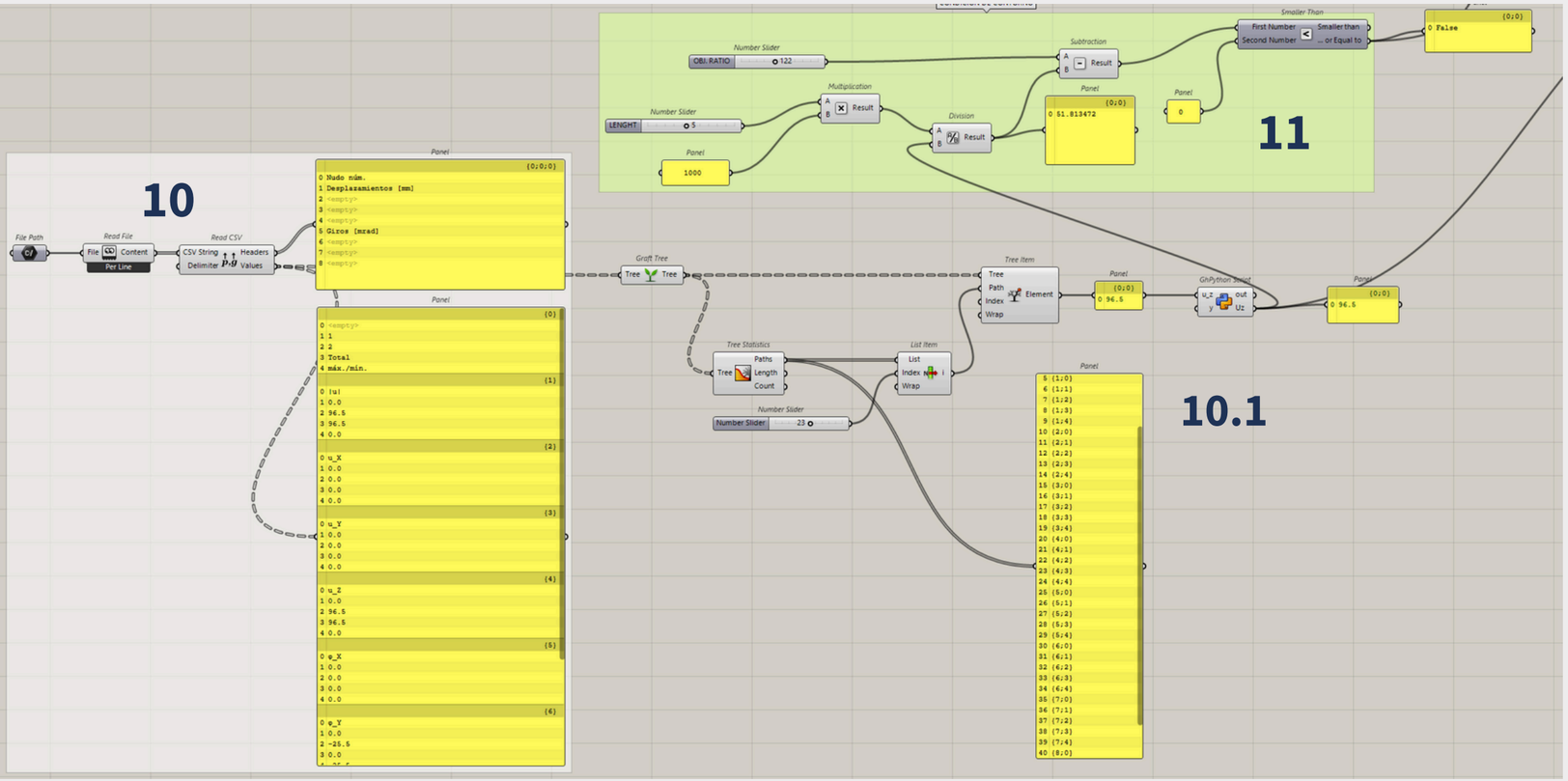
Viga simple



7 Exportación modelo a RFEM

8 Componente de cálculo RFEM

9 Exportación de resultados



10 Lectura de datos del CSV

10.1 Extracción deformación vertical Uz

11 Definición de las condiciones de contorno

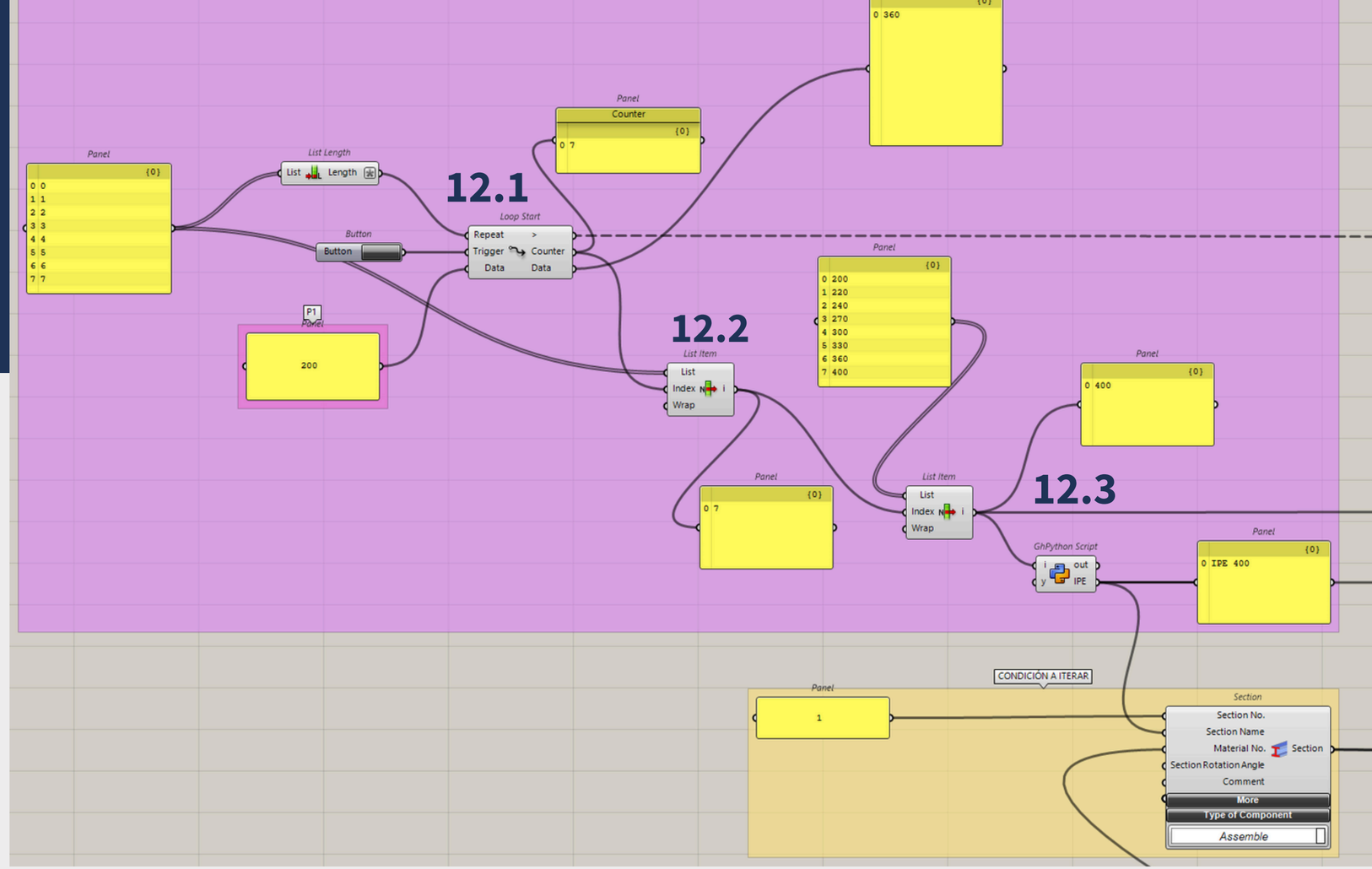
R - (L/Uz) <= 0

CASO 1

Viga simple

12

Bucle iterativo con anemone



12.1

Componentes de inicio y cierre

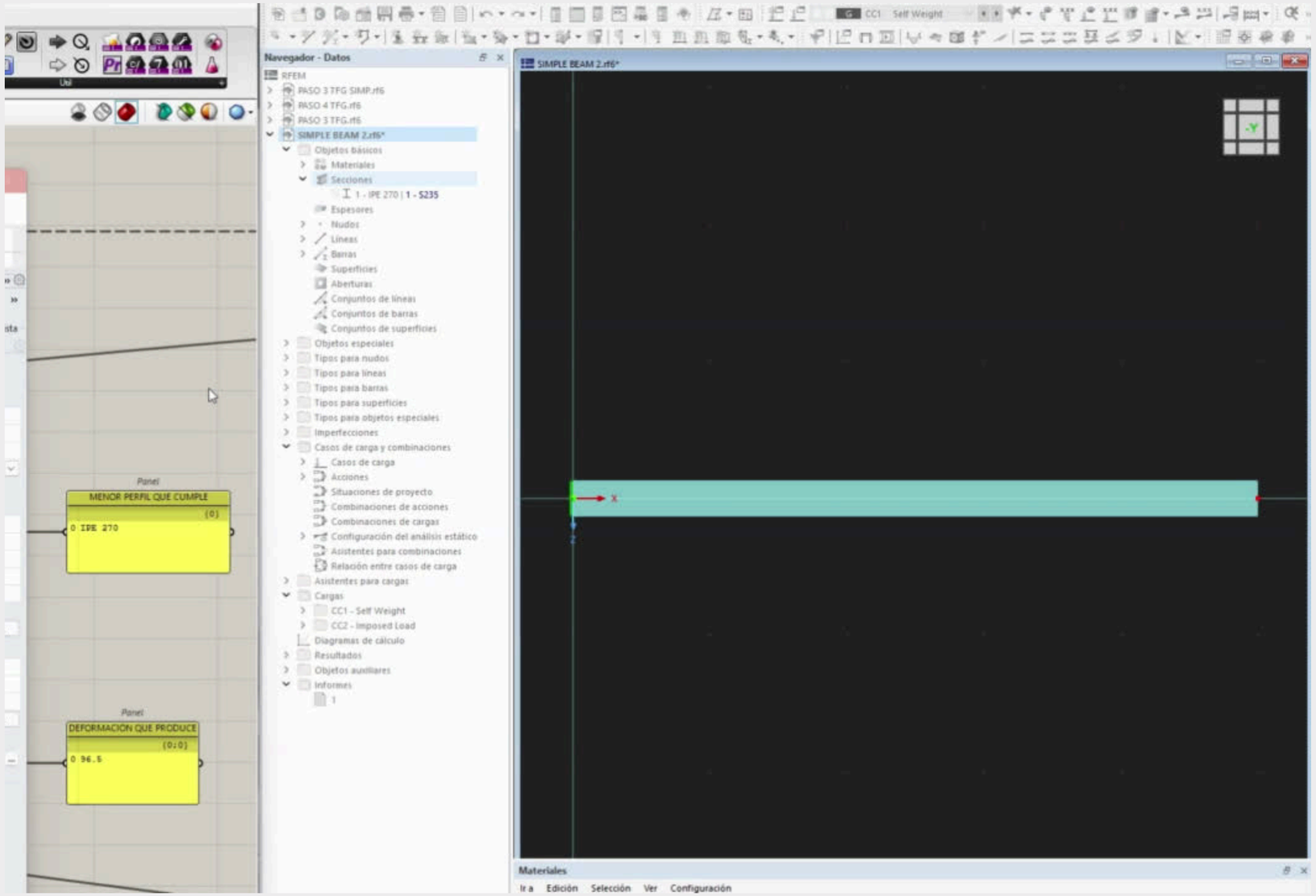
12.2

Uso de la salida “contador”

12.3

Definición de lista de perfiles IPE y vinculación a la sección

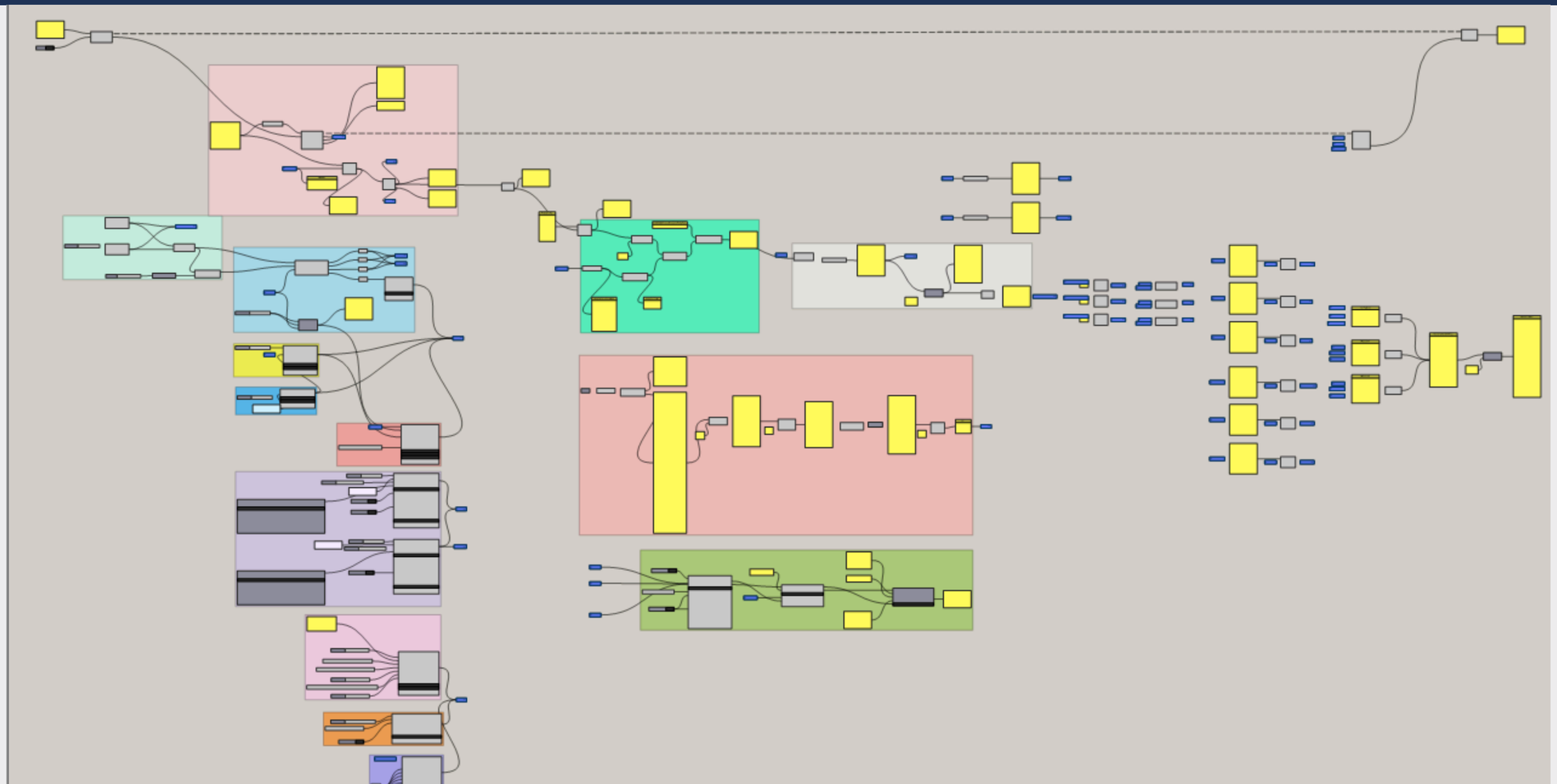
Viga simple


Navegador - Datos
RFEM
PASO 3 TFG SIMP.rf6
PASO 4 TFG.rf6
PASO 3 TFG.rf6
SIMPLE BEAM 2.rf6*
Objetos básicos
Materiales
Secciones
1 - IPE 270 | 1 - S235
Espesores
Nudos
Líneas
Barras
Superficies
Aberturas
Conjuntos de líneas
Conjuntos de barras
Conjuntos de superficies
Objetos especiales
Tipos para nudos
Tipos para líneas
Tipos para barras
Tipos para superficies
Tipos para objetos especiales
Imperfecciones
Casos de carga y combinaciones
Casos de carga
Acciones
Situaciones de proyecto
Combinaciones de acciones
Combinaciones de cargas
Configuración del análisis estático
Asistentes para combinaciones
Relación entre casos de carga
Asistentes para cargas
Cargas
CC1 - Self Weight
CC2 - Imposed Load
Diagramas de cálculo
Resultados
Objetos auxiliares
Informes
1
SIMPLE BEAM 2.rf6*
Panel
MENOR PERFIL QUE CUMPLE
{0}
0 IPE 270
Panel
DEFORMACIÓN QUE PRODUCE
{0;0}
0 96.5
Materiales
Ir a
Edición
Selección
Ver
Configuración

CASO 2

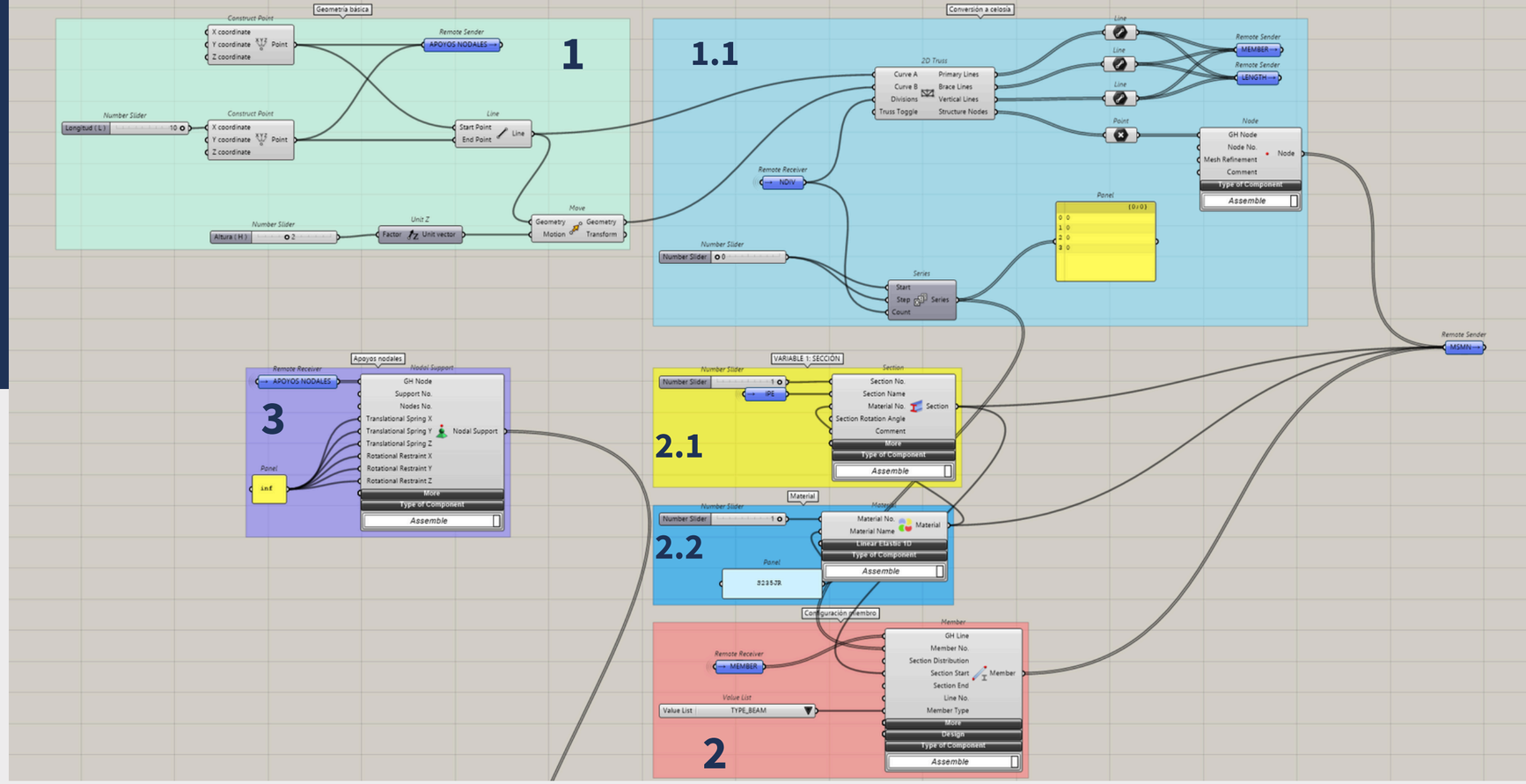
Celosía simple

El objetivo es desarrollar un modelo para encontrar la celosía óptima iterando entre distintas combinaciones de su perfil IPE y número de divisiones, tomando como criterio una función de aptitud, propio de los AG.



CASO 2

Celosía simple



1 **Geometría básica**

1.1 Creación de la celosía

2 **Miembro estructural Viga**

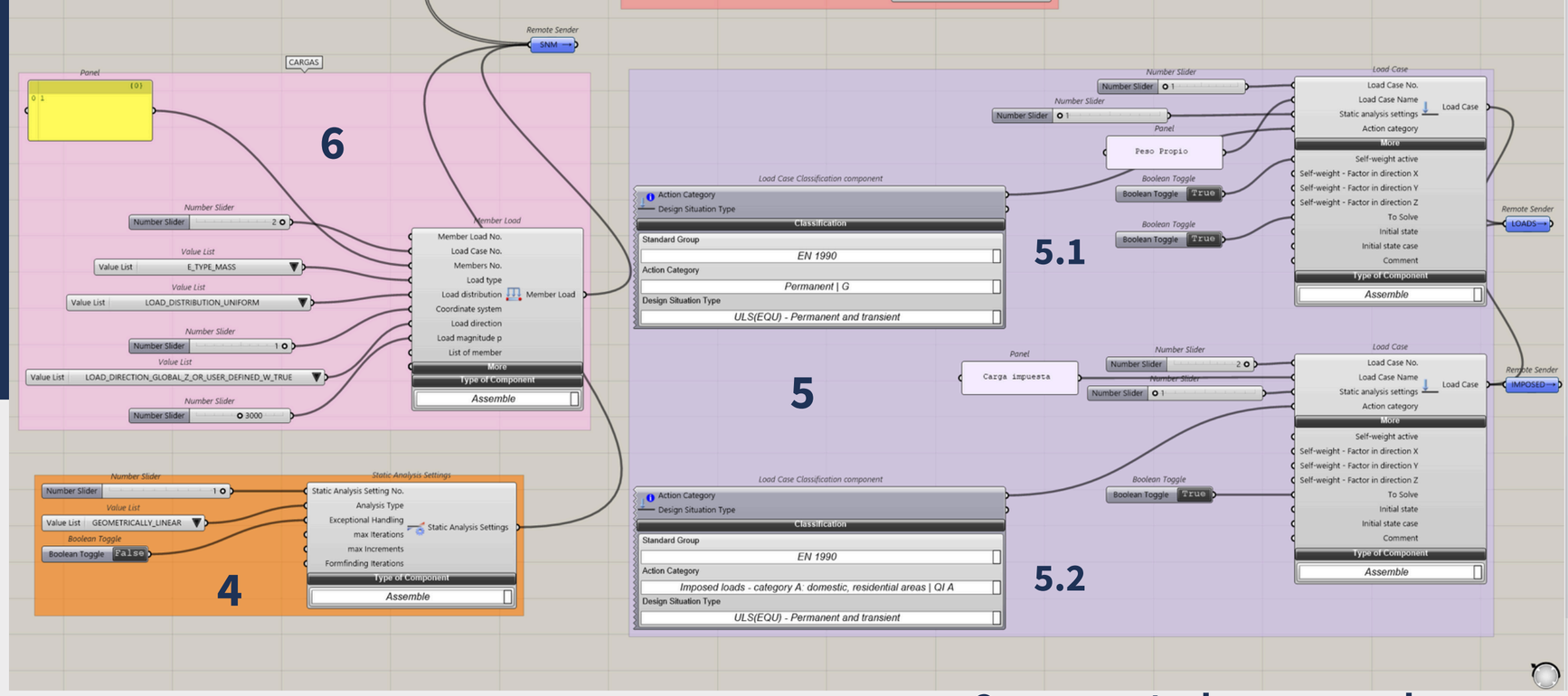
2.1 Sección Inicial

2.2 Material

3 **Apoyos nodales**

CASO 2

Celosía simple



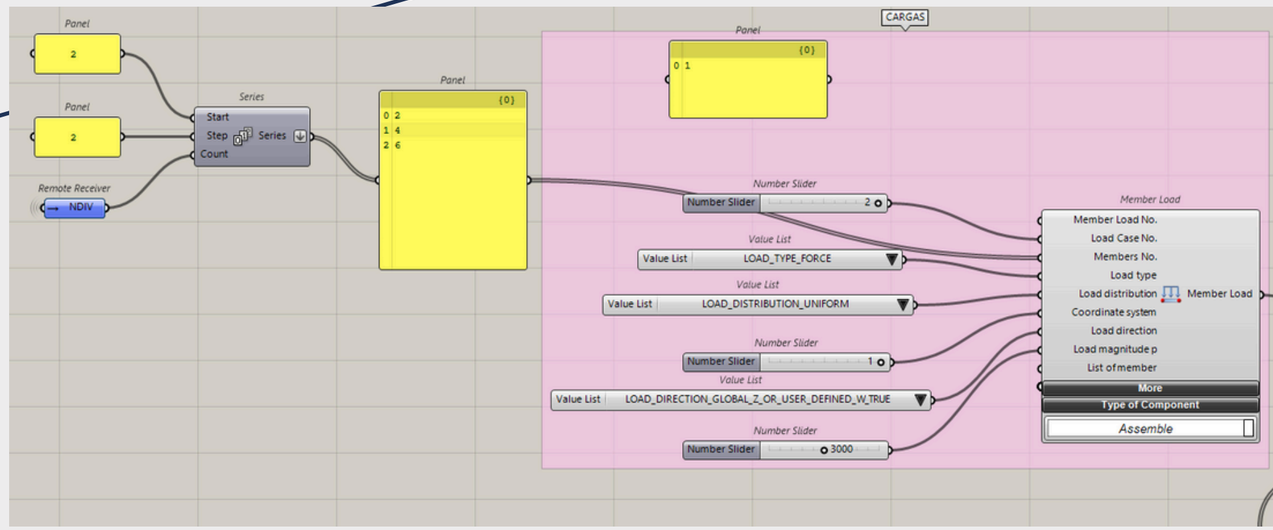
4 **Ajustes de análisis estático**

5 **Casos de carga**

5.1 Peso propio

5.2 Carga impuesta

6 **Componente de carga en el miembro**



CASO 2

Celosía simple

7 Exportación modelo a RFEM

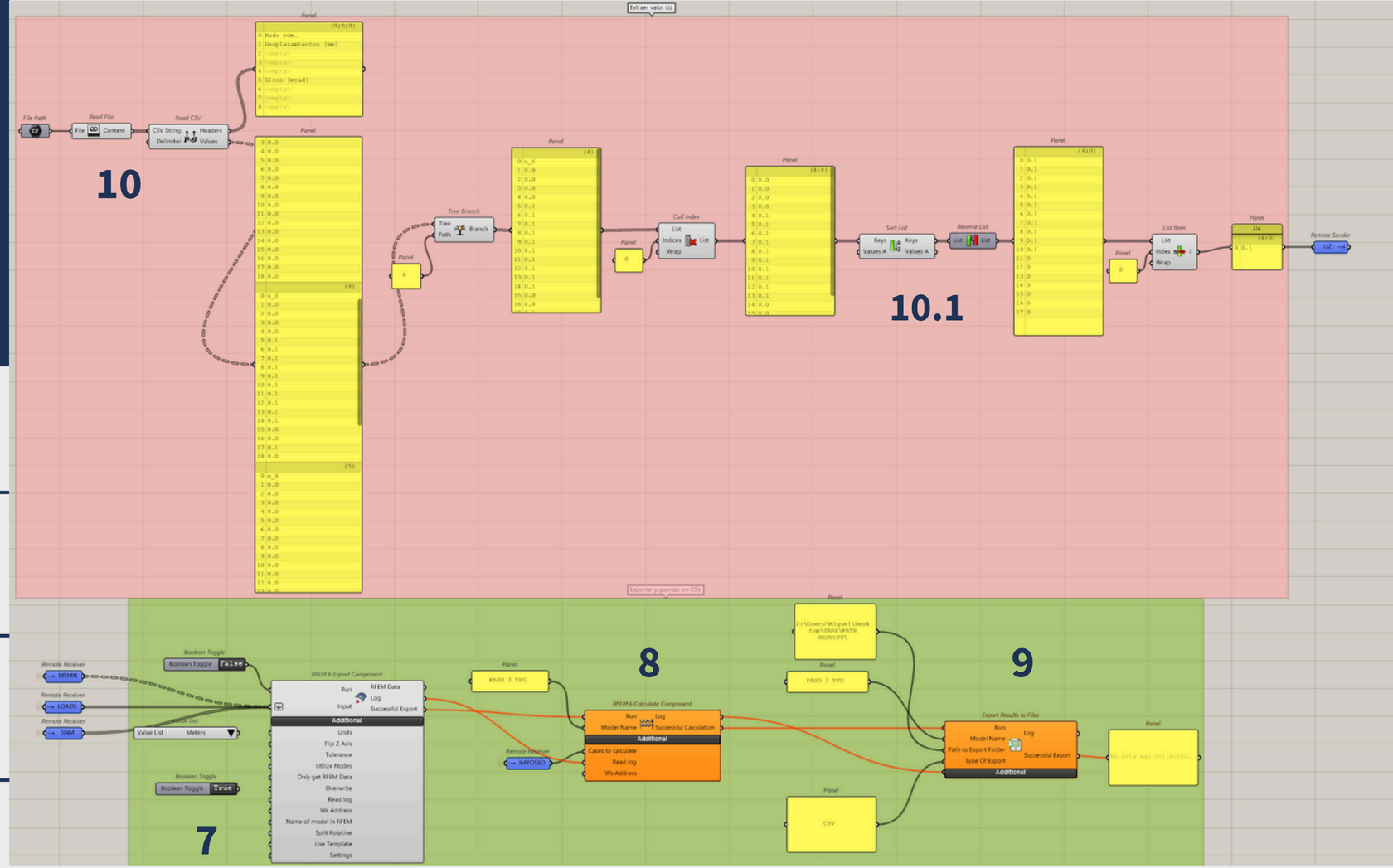
8 Componente de cálculo RFEM

9 Exportación de resultados

10 Lectura de datos del CSV

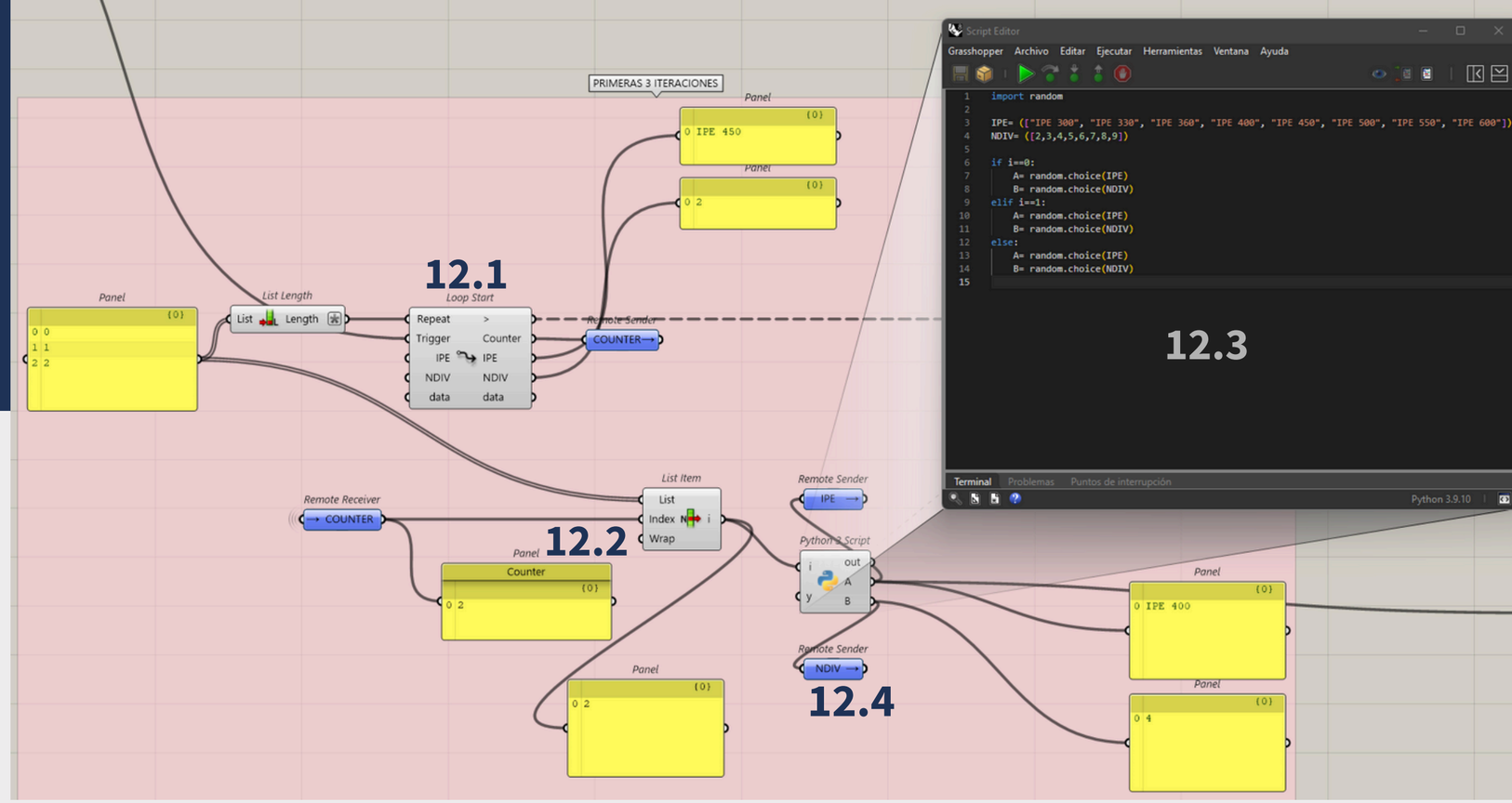
10.1

Extracción deformación vertical Uz



CASO 2

Celosía simple



12

Bucle iterativo con anemone

Genera una población aleatoria y evaluará posteriormente una función

12.1

Componente de inicio

12.2

Uso de la salida “contador”

12.3

Definición de lista de perfiles IPE y rango de Nº de divisiones para generar individuo aleatorio con Python

12.4

Vinculación a la sección y a la geometría para el Nº de divisiones

Universidad LOYOLA

CASO 2

Celosía simple

13

Cálculo del peso propio de la estructura

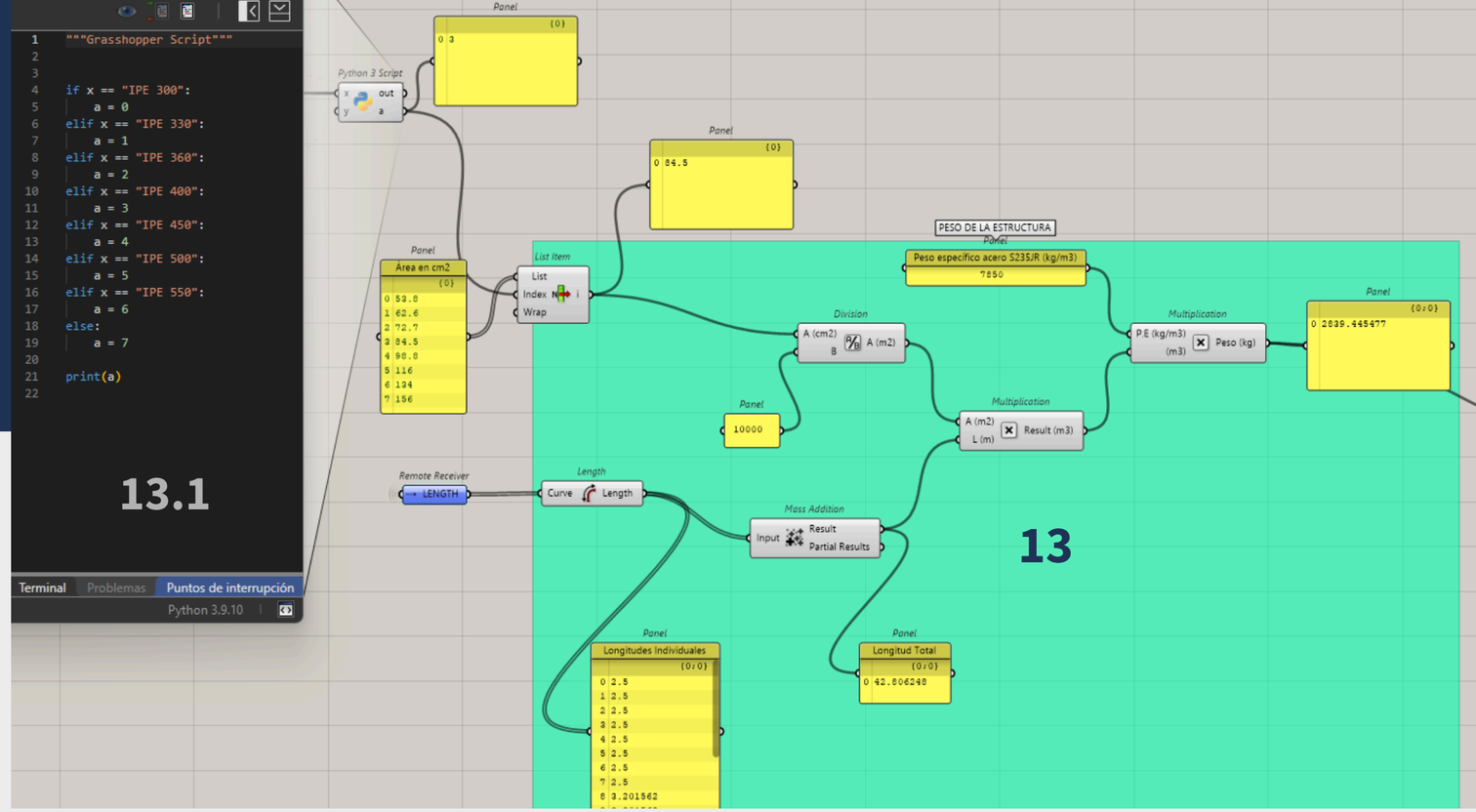
13.1 Código de vinculación del área

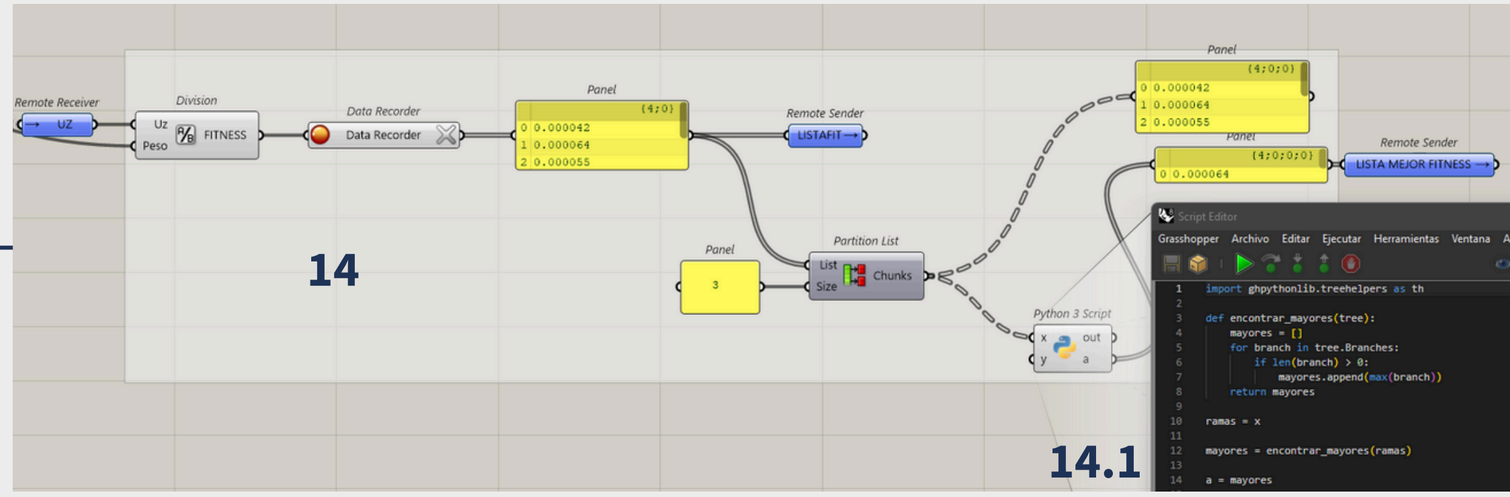
14

Cálculo del fitness

14.1

Código de selección del mayor fitness





CASO 2

Celosía simple

15 Cierre del bucle interior

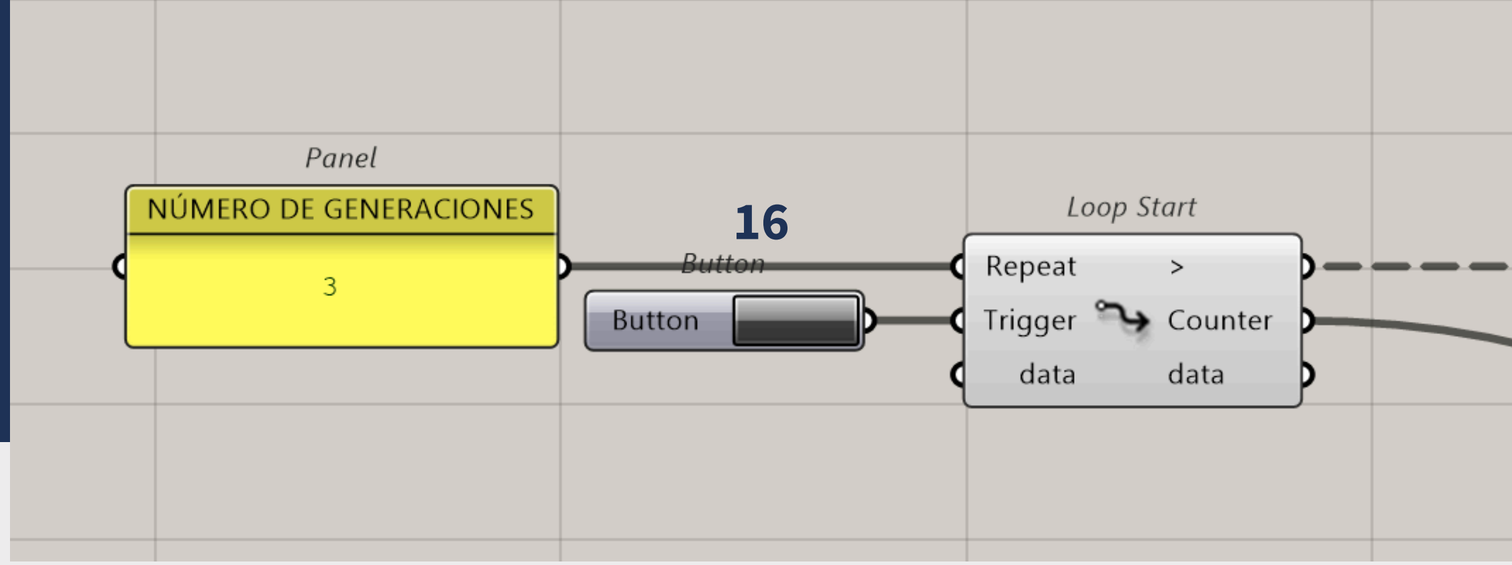
16 Creación bucle exterior

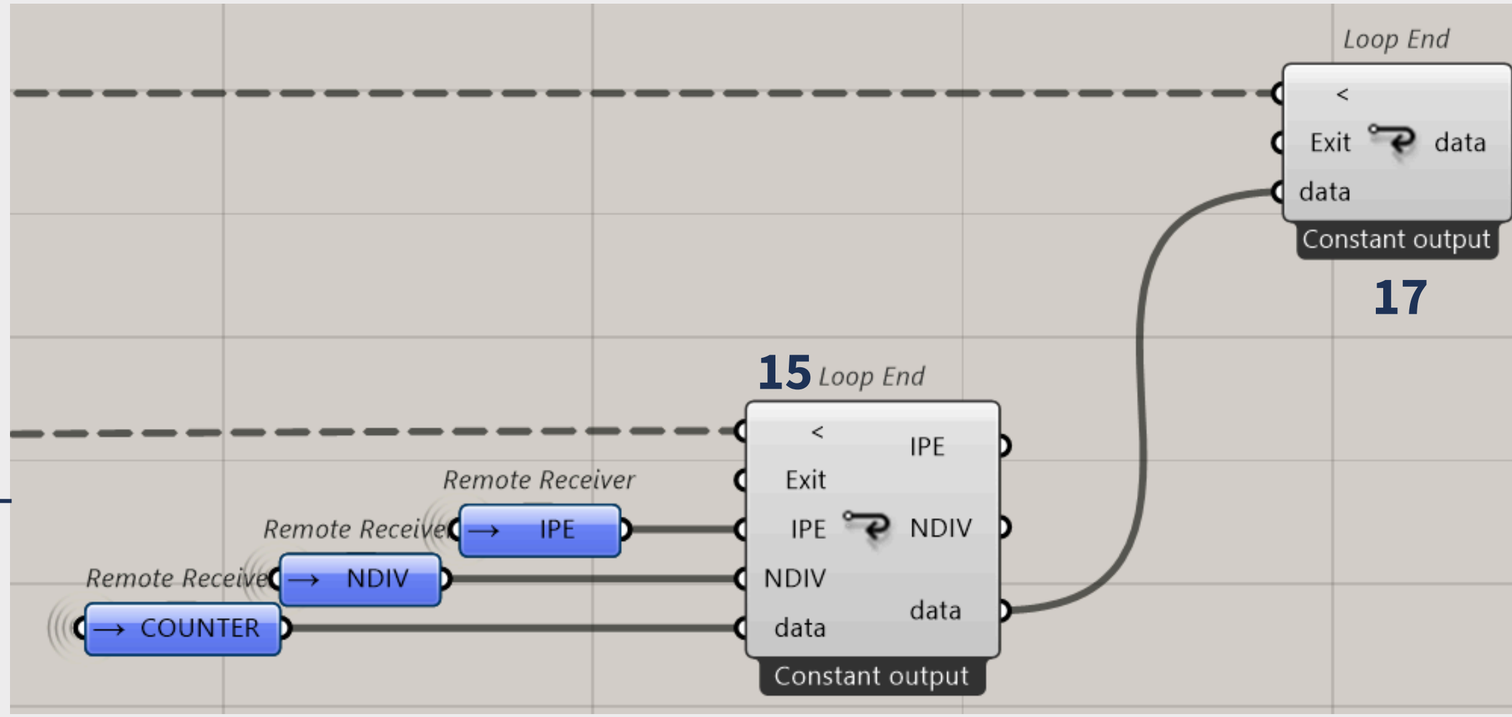
Servirá únicamente para determinar el número de generaciones

17

Cierre del bucle exterior

Anidándolo al interior

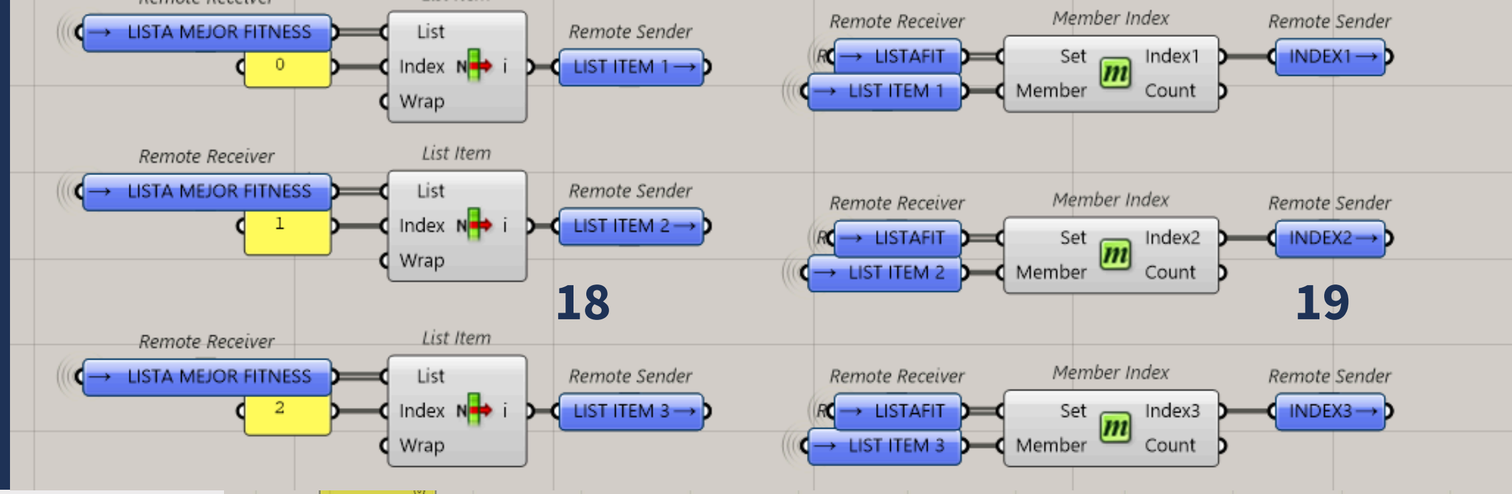




Universidad LOYOLA

CASO 2

Celosía simple



18

Extraemos individualmente el mejor individuo de cada generación

19

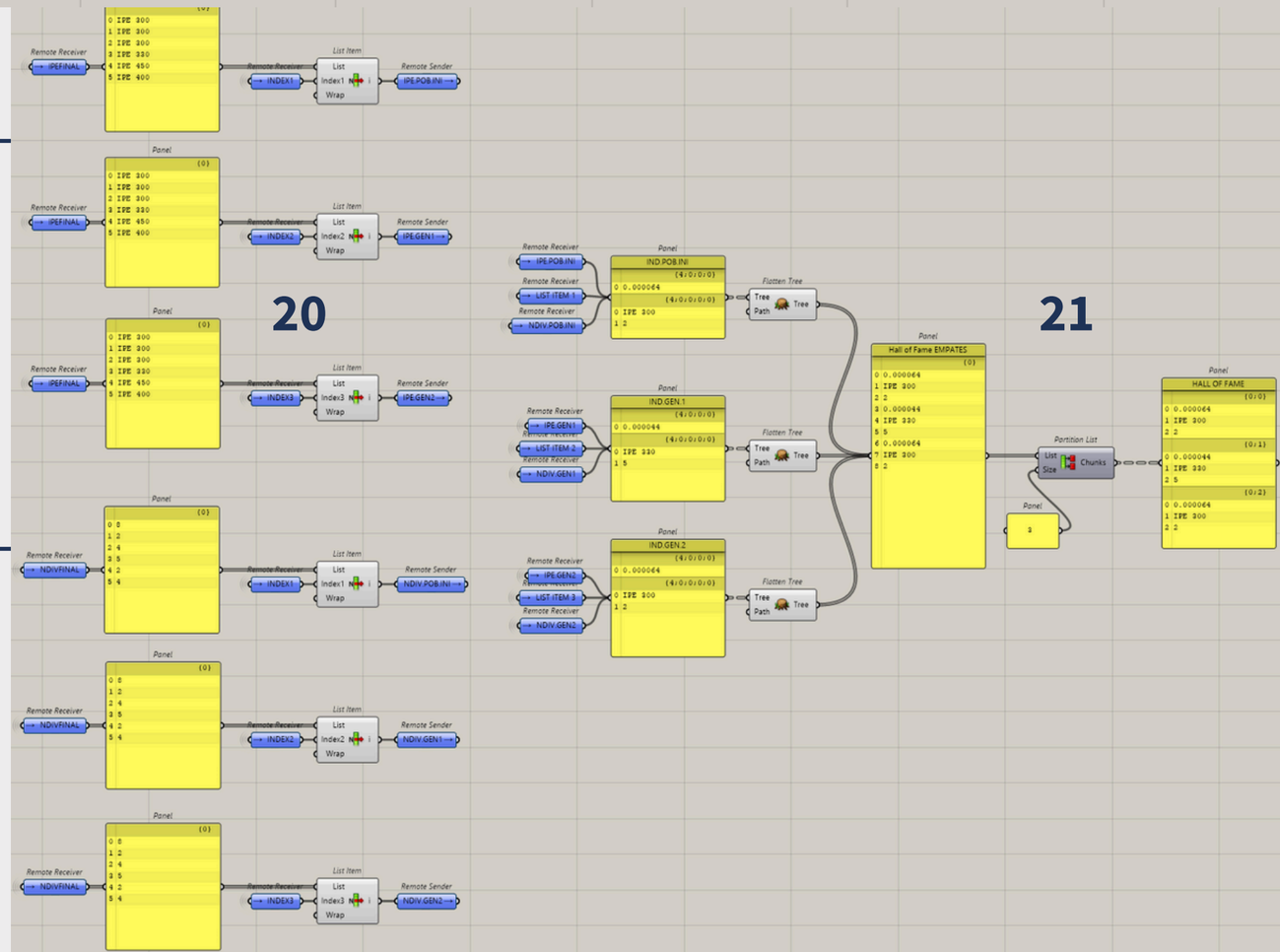
Extraemos sus índices para determinar las posiciones

20

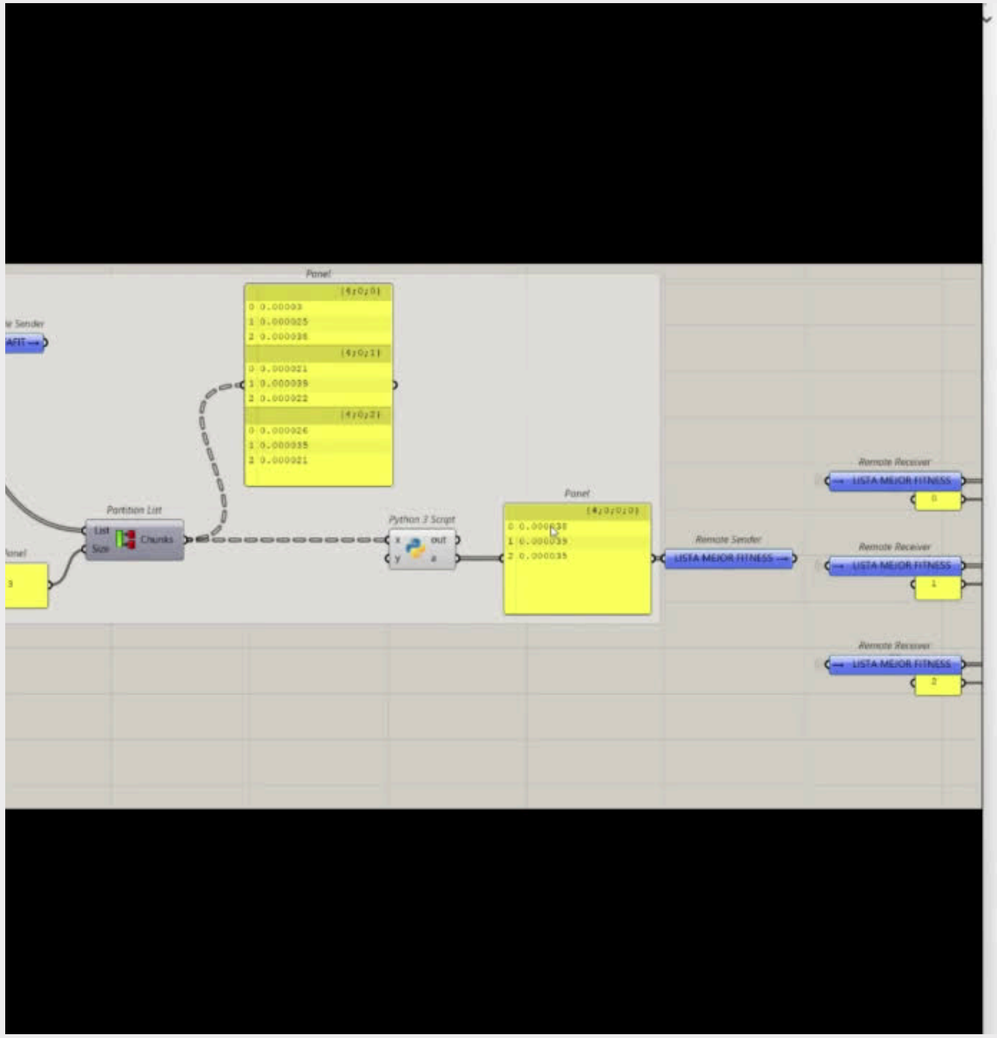
Determinamos el IPE y el NDIV que les corresponden

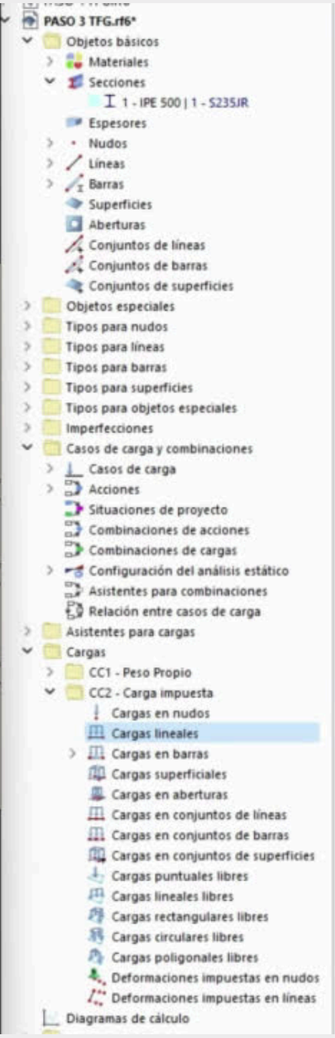
21

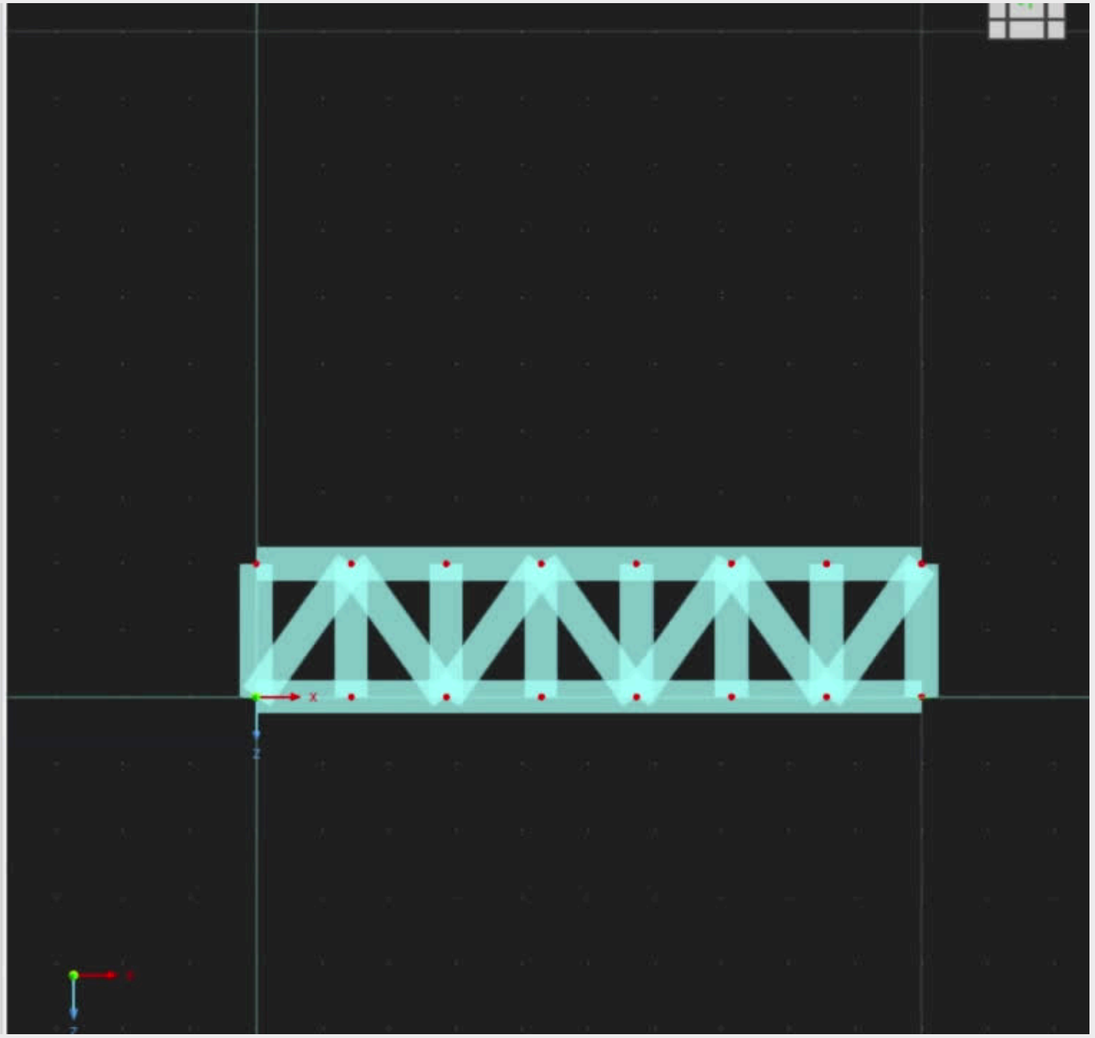
Unificamos fitness, perfil y Nº de divisiones para crear Hall of Fame



Celosía simple



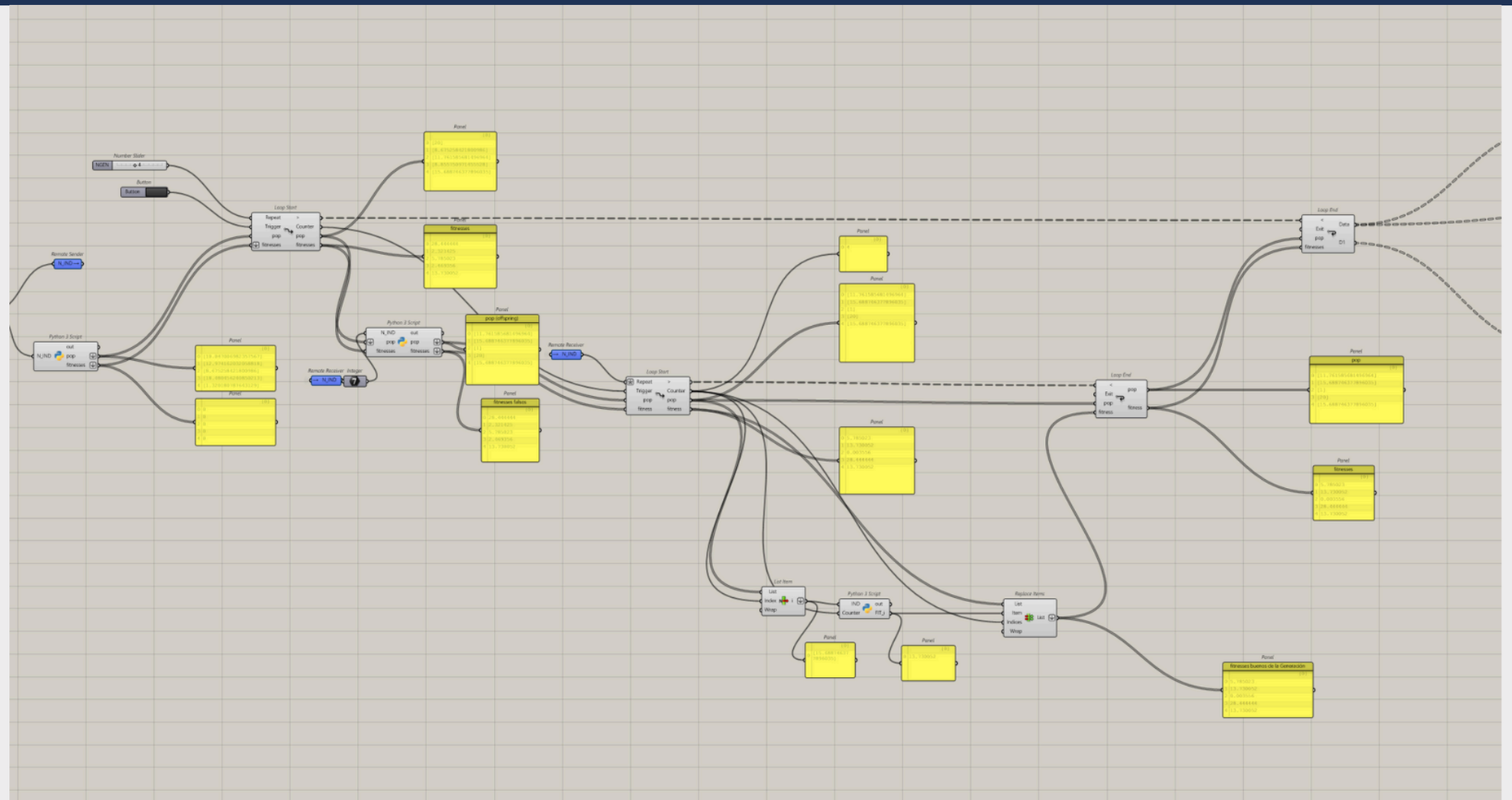




CASO 3

Viga con operadores genéticos

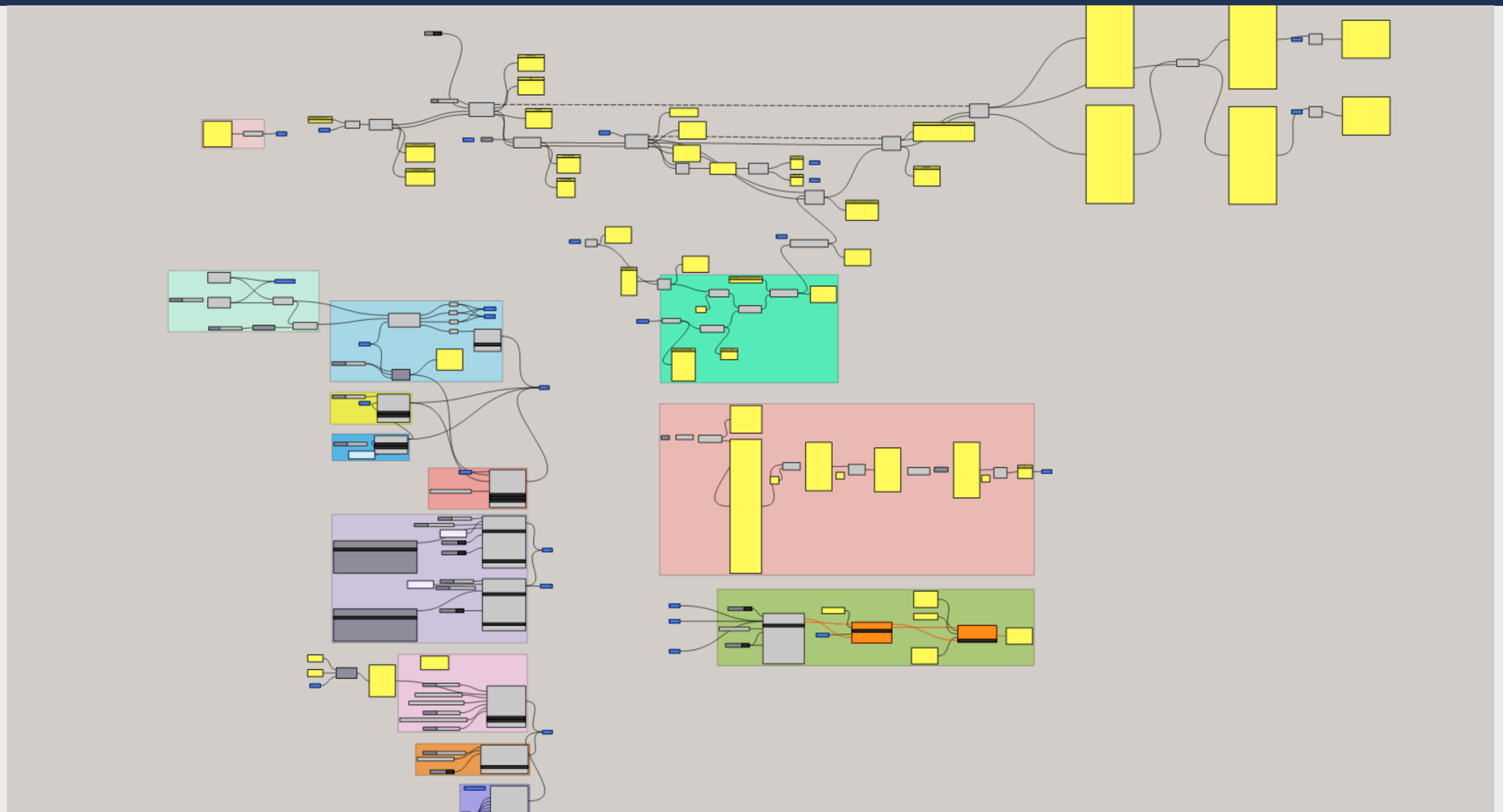
Desarrollo de código de operadores genéticos iterando en un caso simple, nos servirá de base para el caso posterior.



CASO 4

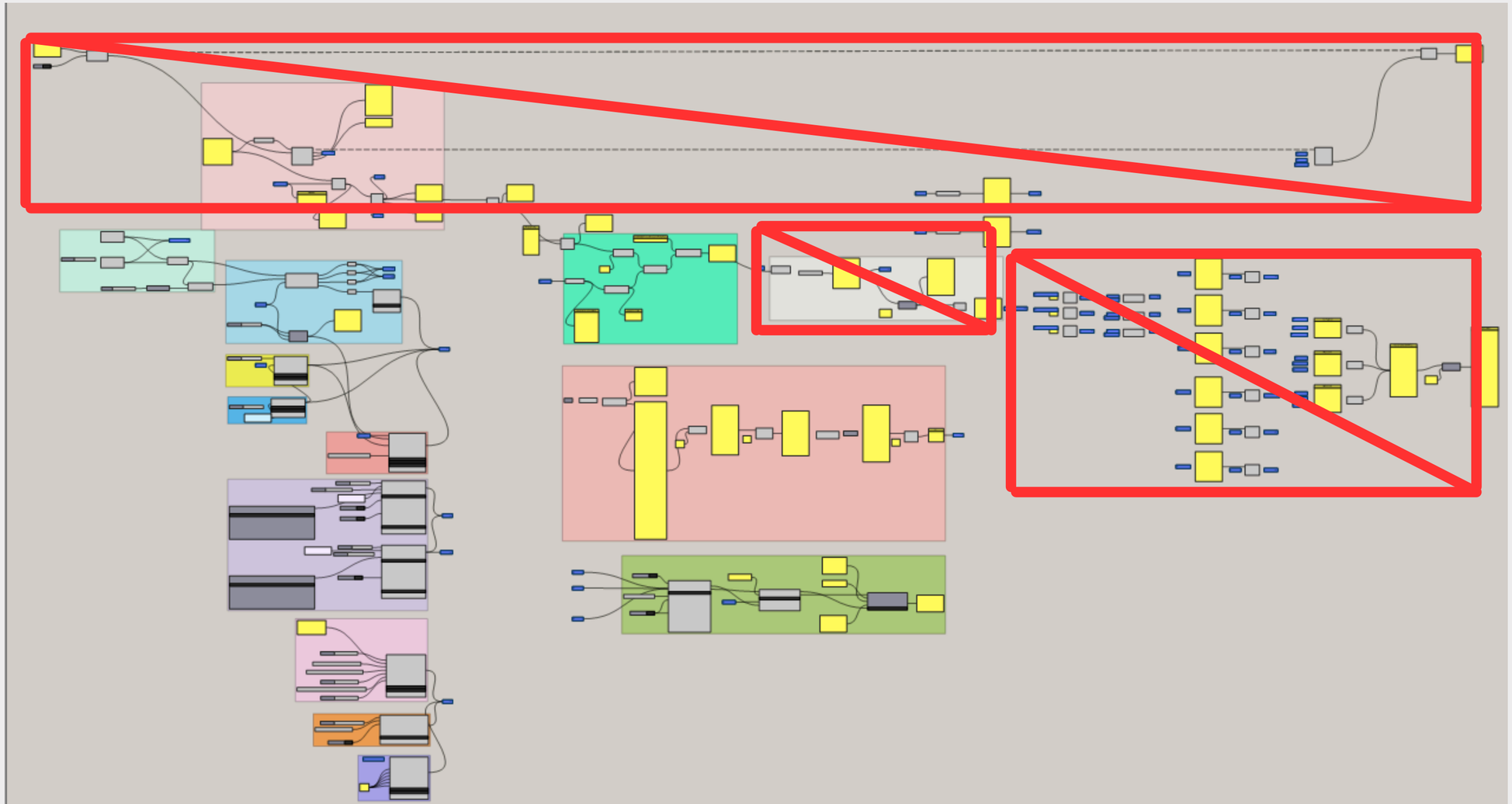
Celosía con operadores genéticos

Combinaremos los dos casos anteriores para crear una aplicación final que optimice el diseño estructural con el uso de algoritmos genéticos en celosías.



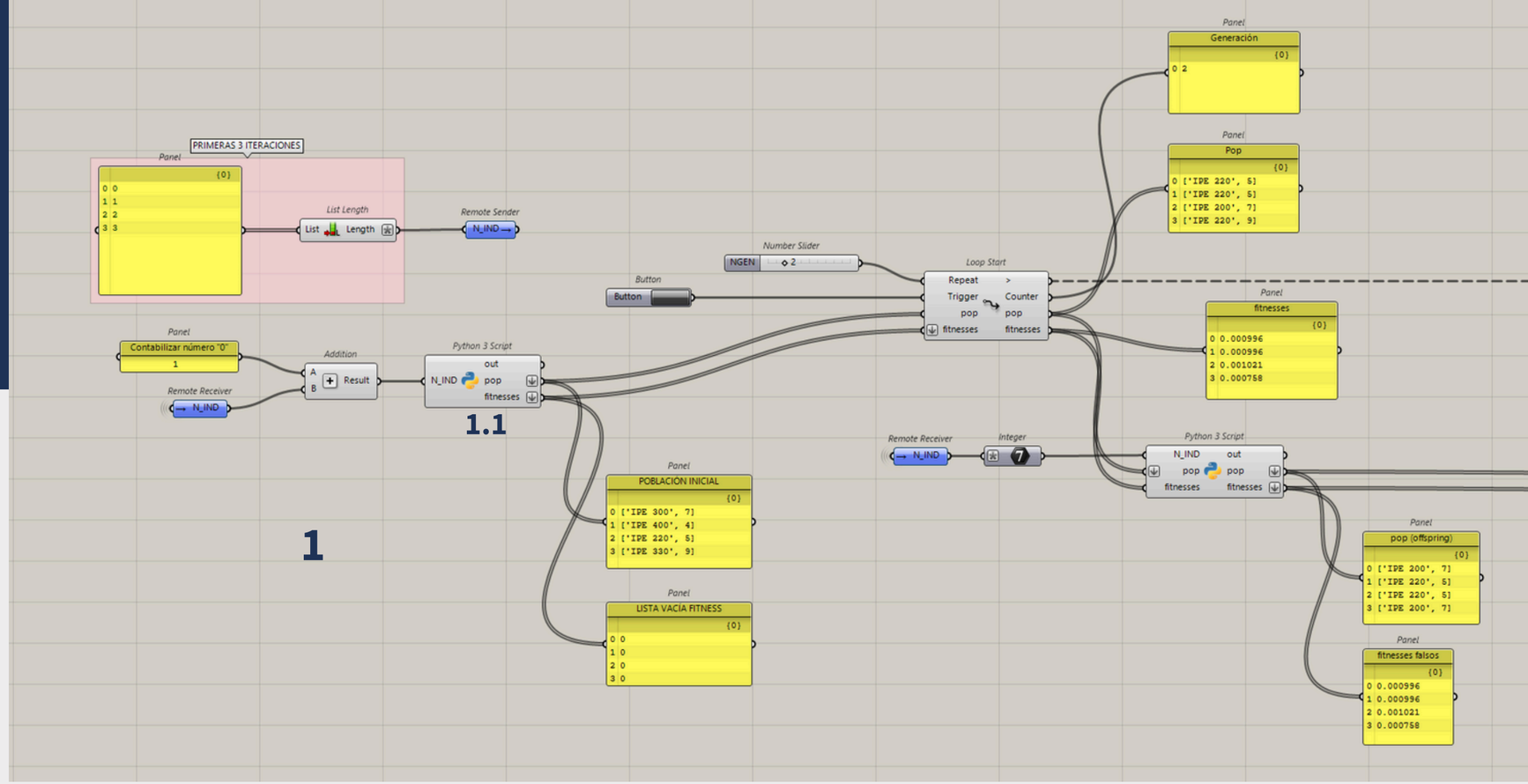
Celosía con operadores genéticos

Restablecimiento del modelo del Caso 2



CASO 4

Celosía con operadores genéticos



1 **Población inicial y lista vacía fitness**

1.1 Definir Nº individuos

1.2 Script *1

2 **Inicio del bucle exterior**

2.1 Definir Nº de generaciones

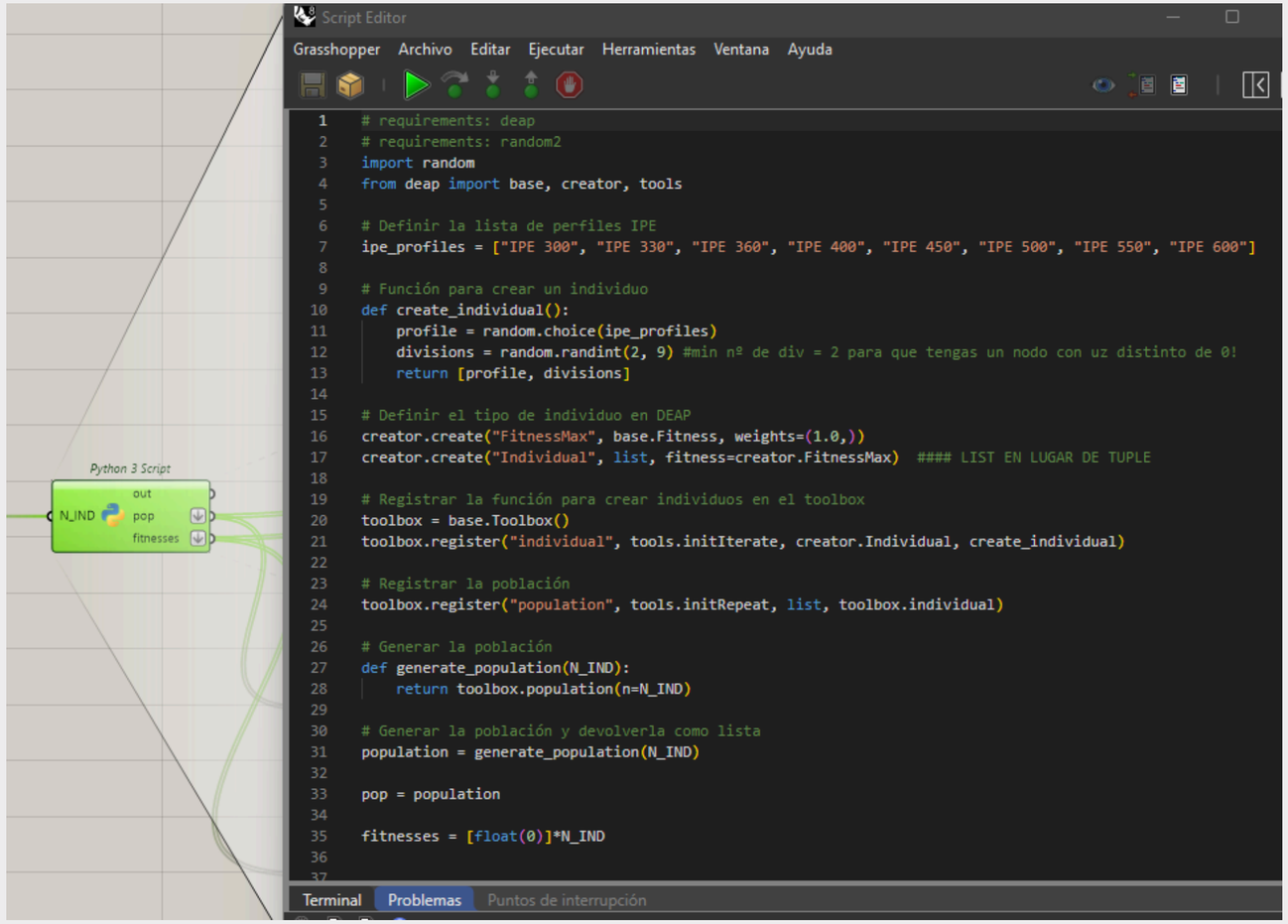
2.2 Vinculación entradas y salidas

2.3 Script *2

Universidad LOYOLA

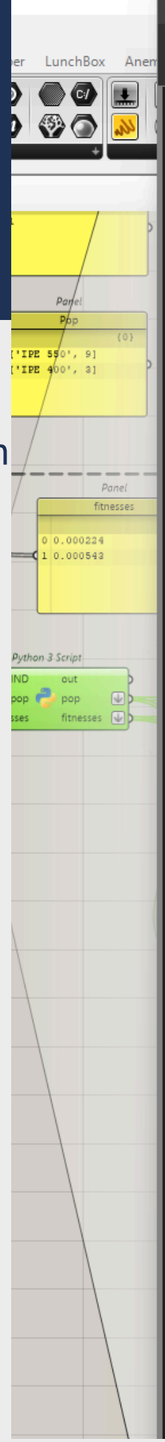
Celosía con operadores genéticos

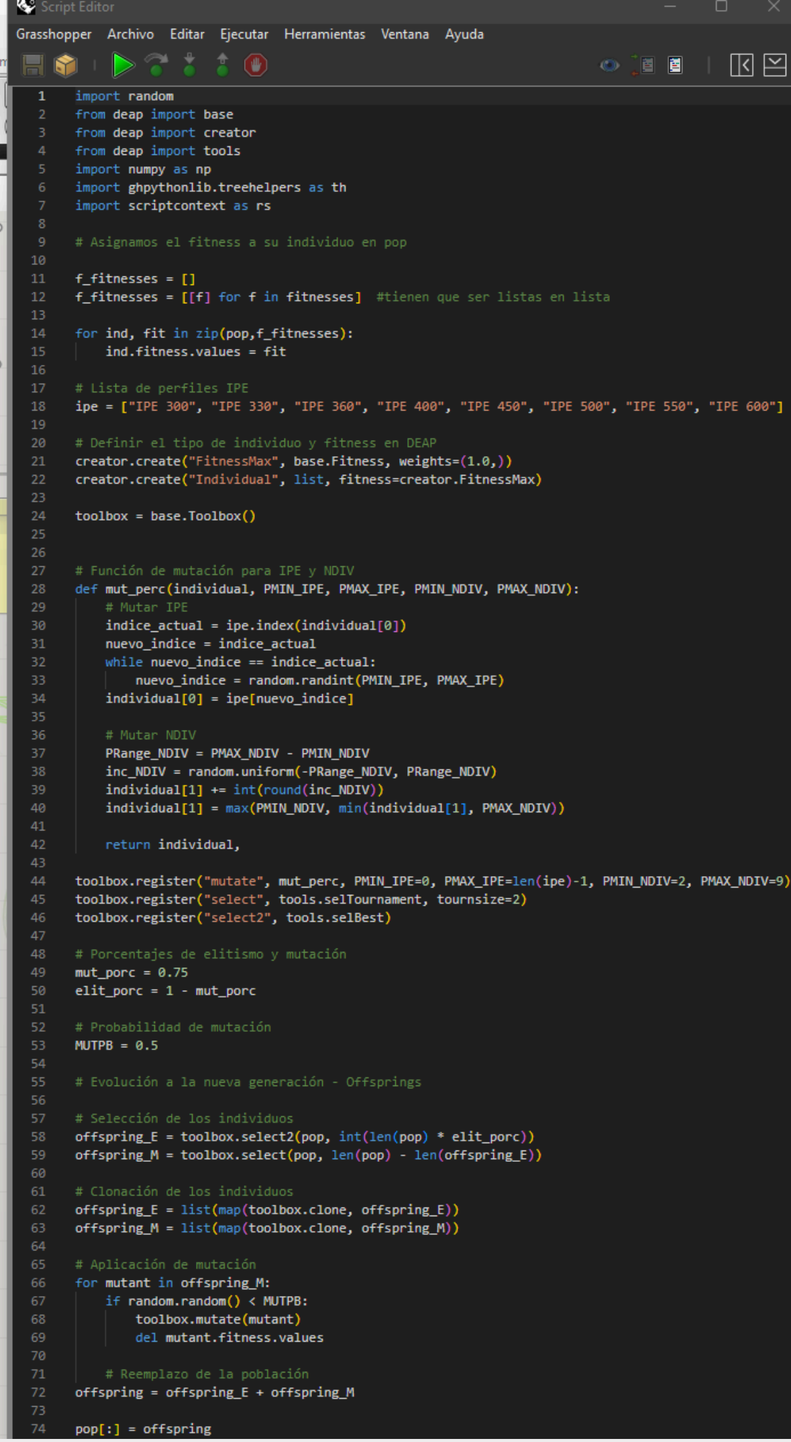
Script 1. Definir pob. inicial y lista de fitness.




```python
# requirements: deap
# requirements: random2
import random
from deap import base, creator, tools

# Definir la lista de perfiles IPE
ipe_profiles = ["IPE 300", "IPE 330", "IPE 360", "IPE 400", "IPE 450", "IPE 500", "IPE 550", "IPE 600"]

# Función para crear un individuo
def create_individual():
    profile = random.choice(ipe_profiles)
    divisions = random.randint(2, 9) #min nº de div = 2 para que tengas un nodo con uz distinto de 0!
    return [profile, divisions]

# Definir el tipo de individuo en DEAP
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)  #### LIST EN LUGAR DE TUPLE

# Registrar la función para crear individuos en el toolbox
toolbox = base.Toolbox()
toolbox.register("individual", tools.initIterate, creator.Individual, create_individual)

# Registrar la población
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

# Generar la población
def generate_population(N_IND):
    return toolbox.population(n=N_IND)

# Generar la población y devolverla como lista
population = generate_population(N_IND)

pop = population

fitnesses = [float(0)]*N_IND
```

Script 2. Implementación de operadores genéticos.






```python
import random
from deap import base
from deap import creator
from deap import tools
import numpy as np
import ghpythonlib.treehelpers as th
import scriptcontext as rs

# Asignamos el fitness a su individuo en pop

f_fitnesses = []
f_fitnesses = [[f] for f in fitnesses]  #tienen que ser listas en lista

for ind, fit in zip(pop,f_fitnesses):
    ind.fitness.values = fit

# Lista de perfiles IPE
ipe = ["IPE 300", "IPE 330", "IPE 360", "IPE 400", "IPE 450", "IPE 500", "IPE 550", "IPE 600"]

# Definir el tipo de individuo y fitness en DEAP
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)

toolbox = base.Toolbox()


# Función de mutación para IPE y NDIV
def mut_perc(individual, PMIN_IPE, PMAX_IPE, PMIN_NDIV, PMAX_NDIV):
    # Mutar IPE
    indice_actual = ipe.index(individual[0])
    nuevo_indice = indice_actual
    while nuevo_indice == indice_actual:
        nuevo_indice = random.randint(PMIN_IPE, PMAX_IPE)
    individual[0] = ipe[nuevo_indice]

    # Mutar NDIV
    PRange_NDIV = PMAX_NDIV - PMIN_NDIV
    inc_NDIV = random.uniform(-PRange_NDIV, PRange_NDIV)
    individual[1] += int(round(inc_NDIV))
    individual[1] = max(PMIN_NDIV, min(individual[1], PMAX_NDIV))

    return individual,

toolbox.register("mutate", mut_perc, PMIN_IPE=0, PMAX_IPE=len(ipe)-1, PMIN_NDIV=2, PMAX_NDIV=9)
toolbox.register("select", tools.selTournament, tournsize=2)
toolbox.register("select2", tools.selBest)

# Porcentajes de elitismo y mutación
mut_porc = 0.75
elit_porc = 1 - mut_porc

# Probabilidad de mutación
MUTPB = 0.5

# Evolución a la nueva generación - Offsprings

# Selección de los individuos
offspring_E = toolbox.select2(pop, int(len(pop) * elit_porc))
offspring_M = toolbox.select(pop, len(pop) - len(offspring_E))

# Clonación de los individuos
offspring_E = list(map(toolbox.clone, offspring_E))
offspring_M = list(map(toolbox.clone, offspring_M))

# Aplicación de mutación
for mutant in offspring_M:
    if random.random() < MUTPB:
        toolbox.mutate(mutant)
        del mutant.fitness.values

    # Reemplazo de la población
offspring = offspring_E + offspring_M

pop[:] = offspring
```

CASO 4

Celosía con operadores genéticos

3

Inicio del bucle interior

Evalúa el fitness en cada individuo y lo vincula con el modelo ya creado.

4

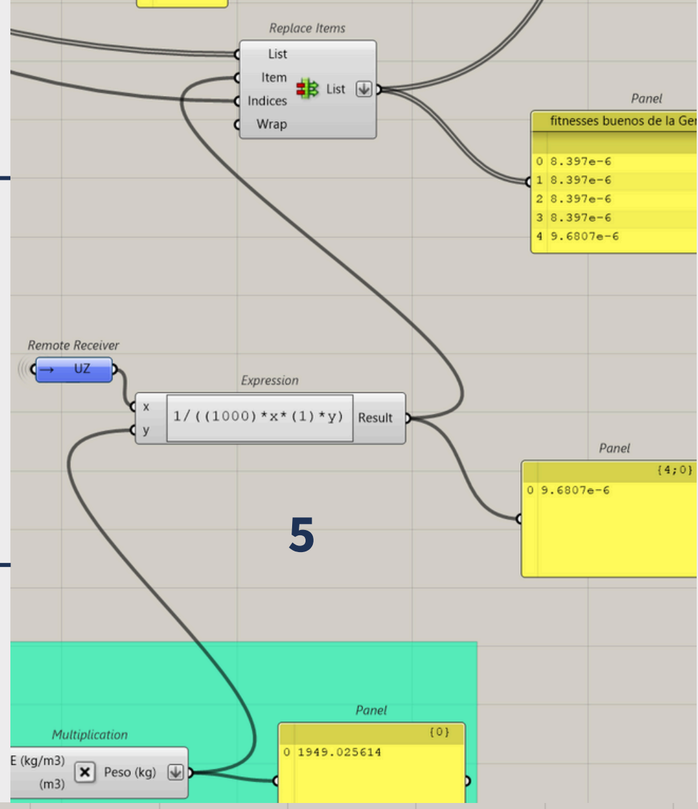
Separar genes de los individuos

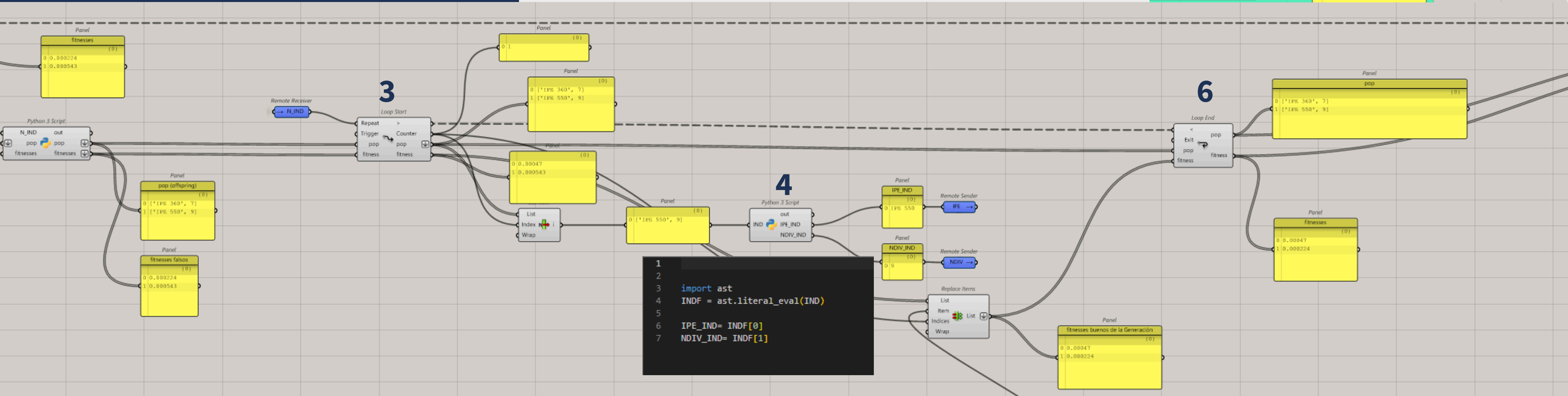
5

Definir nueva función de fitness

6

Cierre del bucle interior





CASO 4

Celosía con operadores genéticos

7

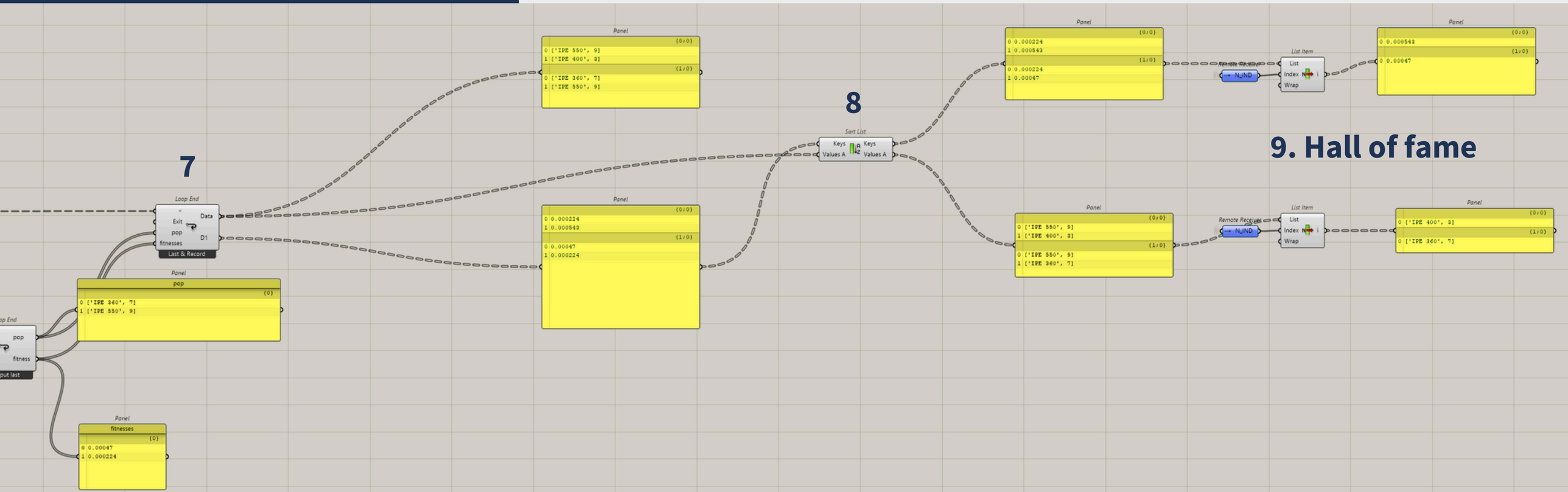
Cierre del bucle exterior y extracción de datos iniciales

8

Separar generaciones en secciones y ordenar sus fitness

9

Extraer el último individuo de cada lista, será el mejor.



Resultados y análisis

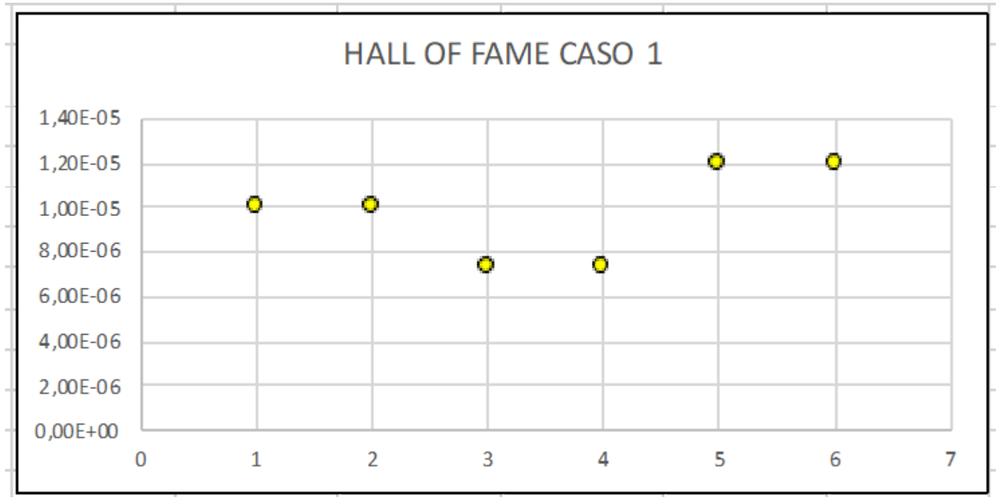
Se ejecutarán numerosas pruebas realizando modificaciones en los casos y se les mostrará alguna de ejemplo además de los resultados finales de todas ellas.

- Ejecutar el modelo variando la geometría del modelo.
- Modificar la carga aplicada y observar su impacto en los resultados.
- Alterar la función de fitness para evaluar diferentes criterios.
- Ajustar los valores de la lista de perfiles IPE para identificar las configuraciones óptimas.

Resultados y análisis

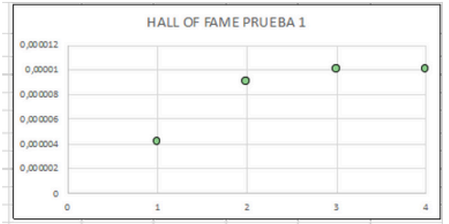
CASO 1. MANTENIENDO LAS CARACTERÍSTICAS ESTABLECIDAS SIN MODIFICACIÓN

MEJORES INDIVIDUOS DE CADA PRUEBA DEL CASO			
PRUEBA	IPE	NDIV	FITNESS
1	330	2	1,00E-05
2	330	2	1,00E-05
3	300	9	7,3733E-06
4	300	9	7,3733E-06
5	**300**	**2**	**0,000012**
6	**300**	**2**	**0,000012**

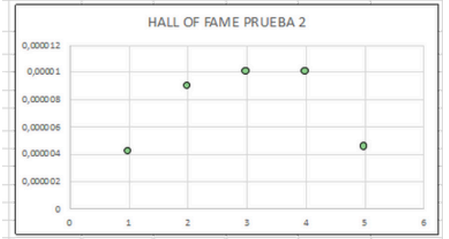


CASO 1: INICIAL, MANTENIENDO LAS CARACTERÍSTICAS ESTABLECIDAS SIN MODIFICACIONES.

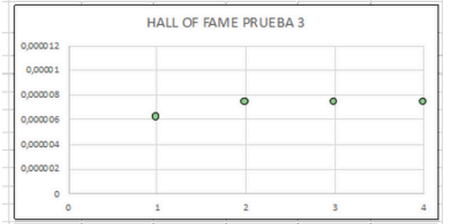
PRUEBA 1			
CARACTERÍSTICAS			
Nº INDIVIDUOS	3		
Nº GENERACIONES TOTAL	3		
MEJOR INDIVIDUO DE LA PRUEBA	330	2	1,00E-05
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	500	6	4,1758E-06
1	360	2	8,99E-06
2	330	2	1,00E-05
3	330	2	1,00E-05



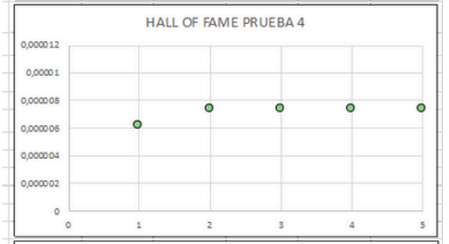
PRUEBA 2			
CARACTERÍSTICAS			
Nº INDIVIDUOS	3		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	330	2	1,00E-05
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	500	6	4,1758E-06
1	360	2	8,9913E-06
2	330	2	1,00E-05
3	330	2	1,00E-05
4	550	3	4,5225E-06



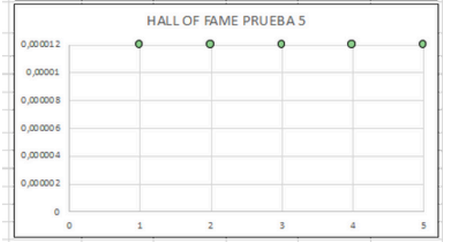
PRUEBA 3			
CARACTERÍSTICAS			
Nº INDIVIDUOS	4		
Nº GENERACIONES TOTAL	3		
MEJOR INDIVIDUO DE LA PRUEBA	300	9	7,3733E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	400	5	6,1645E-06
1	300	9	7,3733E-06
2	300	9	7,3733E-06
3	300	9	7,3733E-06



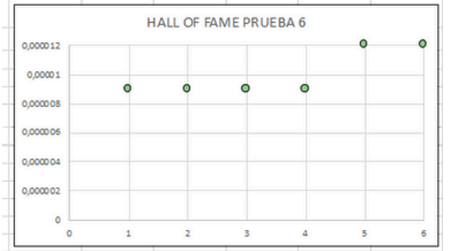
PRUEBA 4			
CARACTERÍSTICAS			
Nº INDIVIDUOS	4		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	300	9	7,3733E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	400	5	6,1645E-06
1	300	9	7,3733E-06
2	300	9	7,3733E-06
3	300	9	7,3733E-06
4	300	9	7,3733E-06



PRUEBA 5			
CARACTERÍSTICAS			
Nº INDIVIDUOS	5		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	300	2	0,000012
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	300	2	1,20E-05
1	300	2	1,20E-05
2	300	2	1,20E-05
3	300	2	1,20E-05
4	300	2	1,20E-05



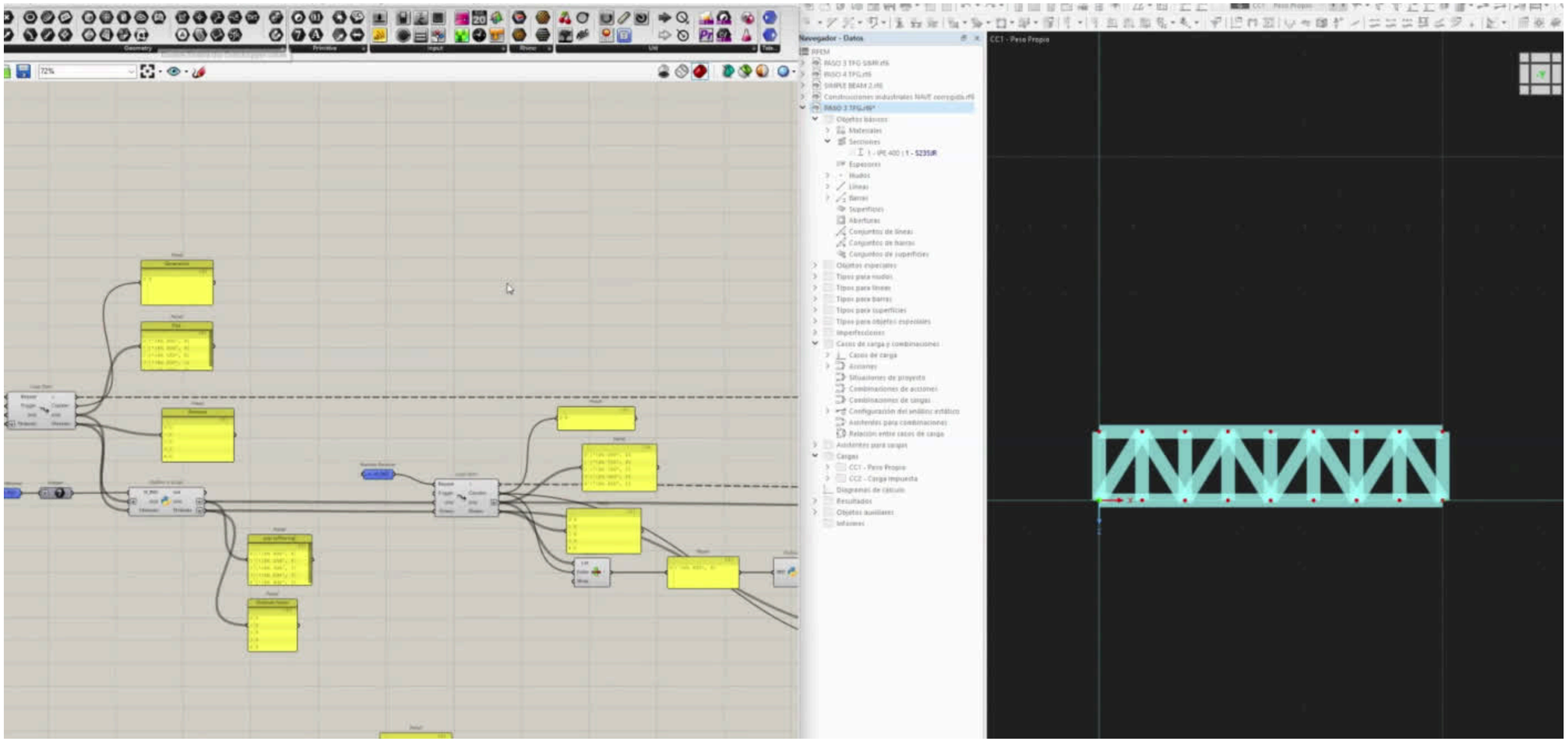
PRUEBA 6			
CARACTERÍSTICAS			
Nº INDIVIDUOS	5		
Nº GENERACIONES TOTAL	5		
MEJOR INDIVIDUO DE LA PRUEBA	300	2	0,000012
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	360	2	8,9913E-06
1	360	2	8,9913E-06
2	360	2	8,9913E-06
3	360	2	8,9913E-06
4	300	2	1,20E-05
5	300	2	1,20E-05



Resultados y análisis

CASO 1. MANTENIENDO LAS CARACTERÍSTICAS ESTABLECIDAS SIN MODIFICACIÓN

PRUEBA 5

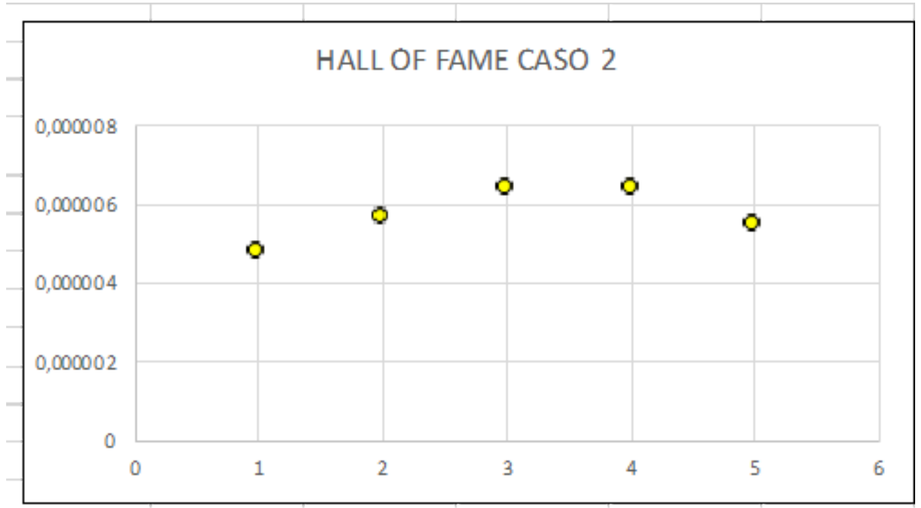


Resultados y análisis

CASO 2. CAMBIO EN LA GEOMETRÍA

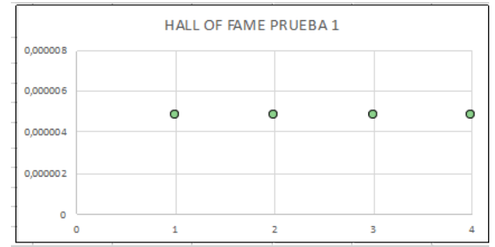
L = 20m h = 3m

MEJORES INDIVIDUOS DE CADA PRUEBA DEL CASO			
PRUEBA	IPE	NDIV	FITNESS
1	300	7	4,80E-06
2	300	4	5,70E-06
3	**300**	**2**	**6,3932E-06**
4	**300**	**2**	**6,3932E-06**
5	330	2	5,49E-06

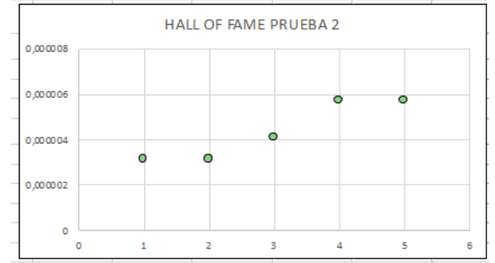


CASO 2: CAMBIO EN LA GEOMETRÍA LONGITUD=20m ALTURA=3m

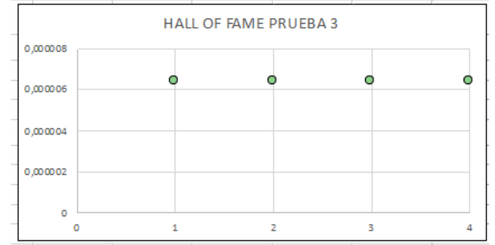
PRUEBA 1			
CARACTERÍSTICAS			
Nº INDIVIDUOS	3		
Nº GENERACIONES TOTAL	3		
MEJOR INDIVIDUO DE LA PRUEBA	300	7	4,80E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	300	7	4,8038E-06
1	300	7	4,8038E-06
2	300	7	4,8038E-06
3	300	7	4,8038E-06



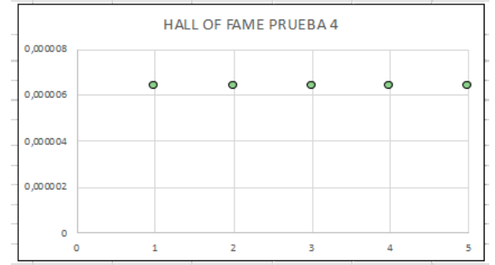
PRUEBA 2			
CARACTERÍSTICAS			
Nº INDIVIDUOS	3		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	300	4	5,70E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	450	4	3,11E-06
1	450	4	3,11E-06
2	400	2	4,07E-06
3	300	4	5,70E-06
4	300	4	5,70E-06



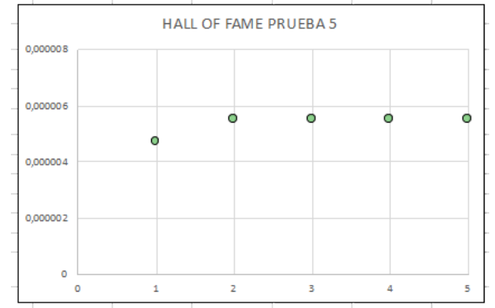
PRUEBA 3			
CARACTERÍSTICAS			
Nº INDIVIDUOS	5		
Nº GENERACIONES TOTAL	3		
MEJOR INDIVIDUO DE LA PRUEBA	300	2	6,3932E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	300	2	6,3932E-06
1	300	2	6,3932E-06
2	300	2	6,3932E-06
3	300	2	6,3932E-06



PRUEBA 4			
CARACTERÍSTICAS			
Nº INDIVIDUOS	4		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	300	2	6,3932E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	300	2	6,3932E-06
1	300	2	6,3932E-06
2	300	2	6,3932E-06
3	300	2	6,3932E-06
4	300	2	6,3932E-06



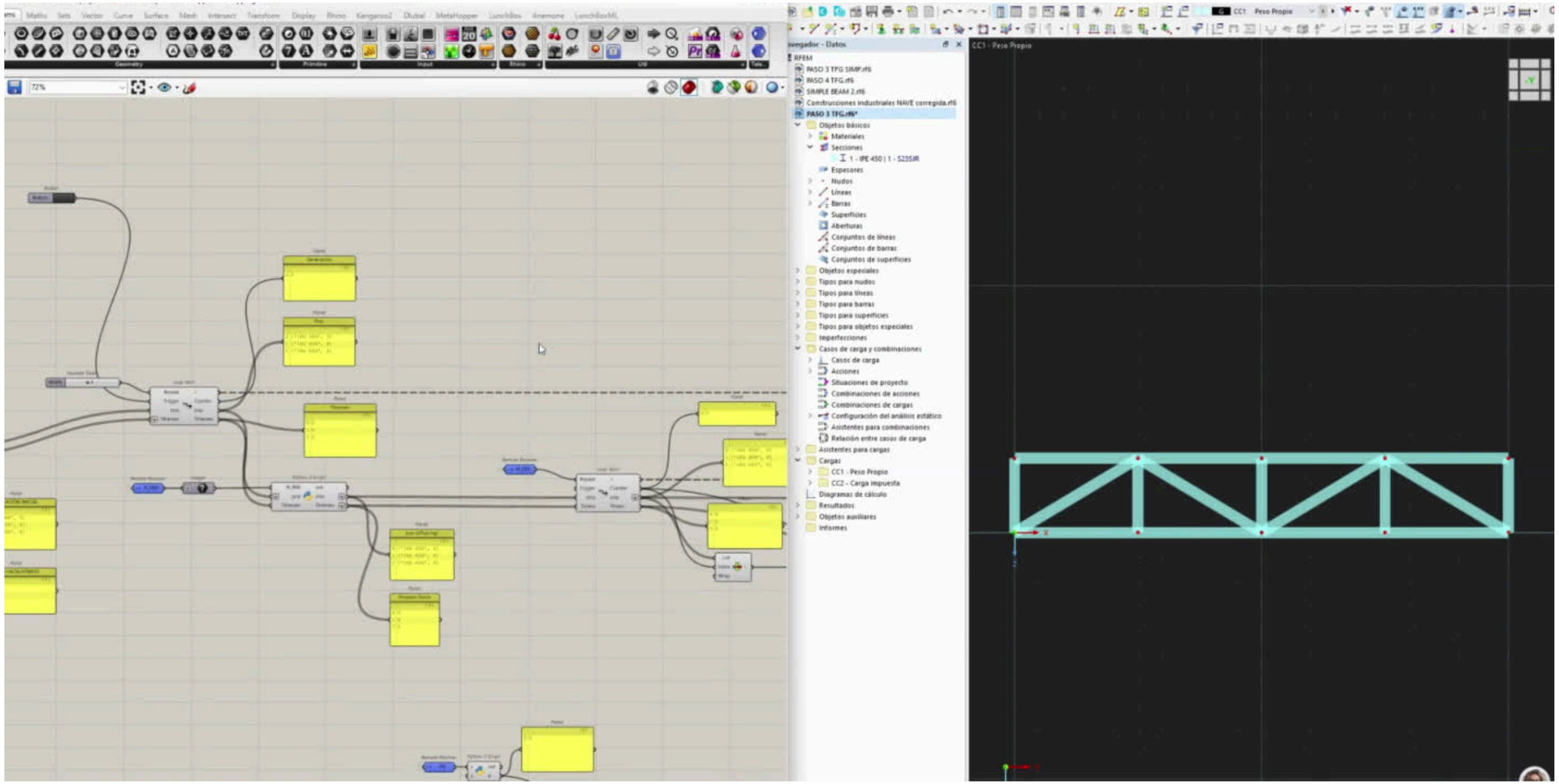
PRUEBA 5			
CARACTERÍSTICAS			
Nº INDIVIDUOS	5		
Nº GENERACIONES TOTAL	6		
MEJOR INDIVIDUO DE LA PRUEBA	330	2	5,49E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	360	2	4,73E-06
1	330	2	5,49E-06
2	330	2	5,49E-06
3	330	2	5,49E-06
4	330	2	5,49E-06
5	330	2	5,49E-06
6	330	2	5,49E-06



Resultados y análisis

CASO 2. CAMBIO EN LA GEOMETRÍA L = 20m h = 3m

PRUEBA 2

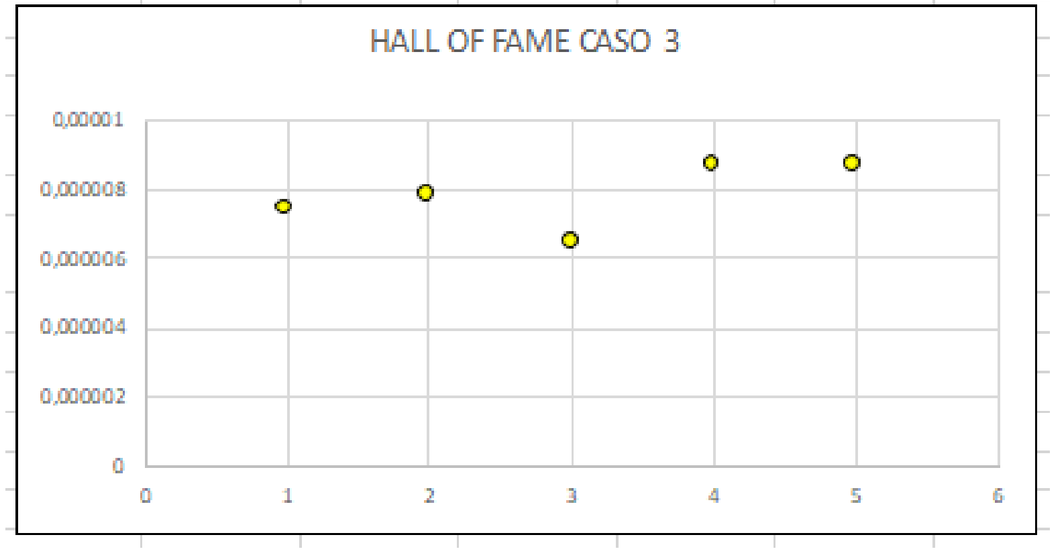


Resultados y análisis

CASO 3. CAMBIO EN LA GEOMETRÍA Y EN LA CARGA IMPUESTA.

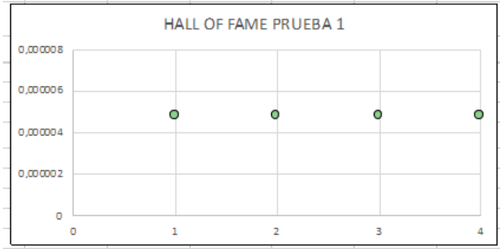
L = 15m h = 2m P= 10kN

MEJORES INDIVIDUOS DE CADA PRUEBA DEL CASO			
PRUEBA	IPE	NDIV	FITNESS
1	330	2	7,45E-06
2	300	4	7,84E-06
3	360	2	6,4167E-06
4	**300**	**2**	**8,6708E-06**
5	**300**	**2**	**8,6708E-06**

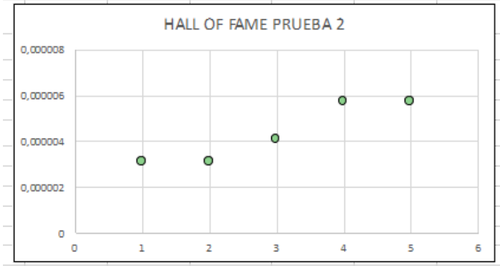


CASO 3: CAMBIO EN LA GEOMETRÍA Y CARGA. L=15m H=2m CARGA= 10KN

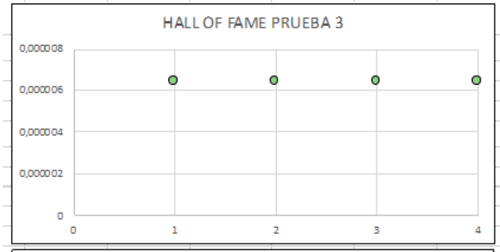
PRUEBA 1			
CARACTERÍSTICAS			
Nº INDIVIDUOS	3		
Nº GENERACIONES TOTAL	3		
MEJOR INDIVIDUO DE LA PRUEBA	330	2	7,45E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	400	2	5,21E-06
1	400	2	5,21E-06
2	400	2	5,21E-06
3	330	2	7,45E-06



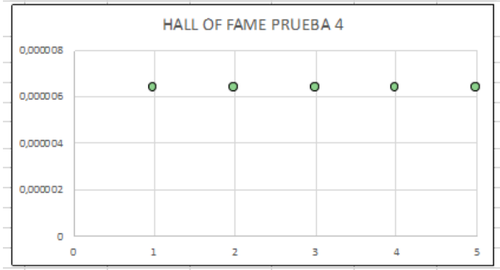
PRUEBA 2			
CARACTERÍSTICAS			
Nº INDIVIDUOS	3		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	300	4	7,84E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	300	8	6,3885E-06
1	330	2	7,4519E-06
2	330	2	7,4519E-06
3	300	4	7,84E-06
4	330	2	7,4519E-06



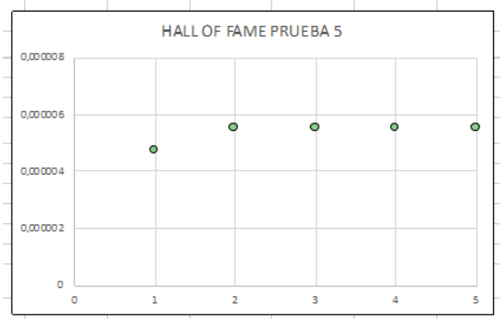
PRUEBA 3			
CARACTERÍSTICAS			
Nº INDIVIDUOS	6		
Nº GENERACIONES TOTAL	3		
MEJOR INDIVIDUO DE LA PRUEBA	360	2	6,4167E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	330	6	6,0743E-06
1	330	6	6,0743E-06
2	330	6	6,0743E-06
3	360	2	6,4167E-06



PRUEBA 4			
CARACTERÍSTICAS			
Nº INDIVIDUOS	4		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	300	2	8,6708E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	300	6	7,0679E-06
1	300	6	7,0679E-06
2	300	6	7,0679E-06
3	300	2	8,6708E-06
4	300	2	8,6708E-06



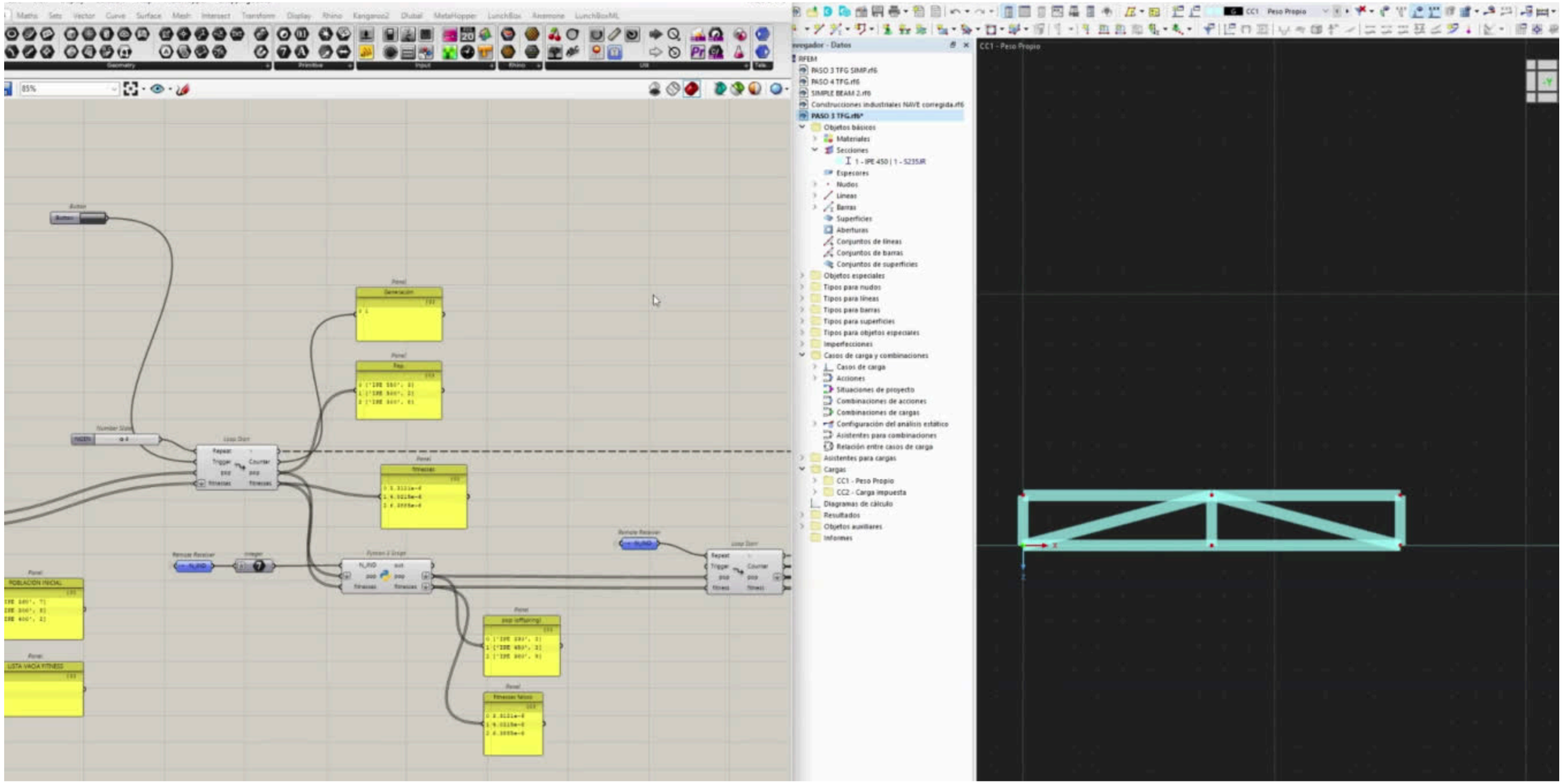
PRUEBA 5			
CARACTERÍSTICAS			
Nº INDIVIDUOS	5		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	300	2	8,6708E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	330	7	5,77E-06
1	300	2	8,6708E-06
2	300	2	8,6708E-06
3	300	2	8,6708E-06
4	300	2	8,6708E-06



Resultados y análisis

CASO 3. CAMBIO EN LA GEOMETRÍA Y EN LA CARGA IMPUESTA. L = 15m h = 2m P= 10kN

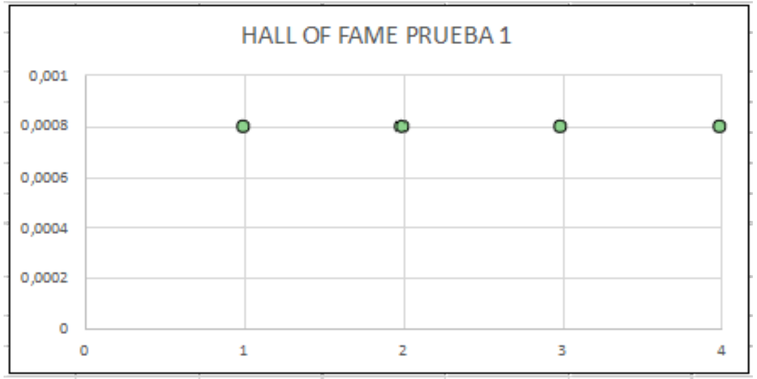
PRUEBA 2

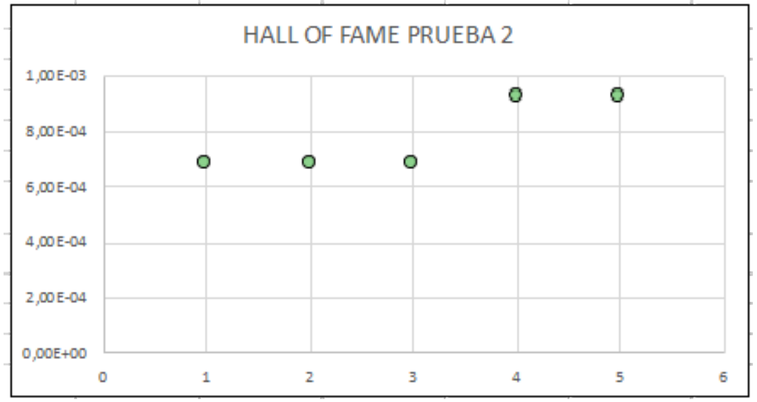


Resultados y análisis

CASO 4. CAMBIO EN LA FUNCIÓN DE FITNESS CON MÉTODO DE PESOS PONDERADOS.
alpha (peso) = 0,7 beta (deformación) = 0,3

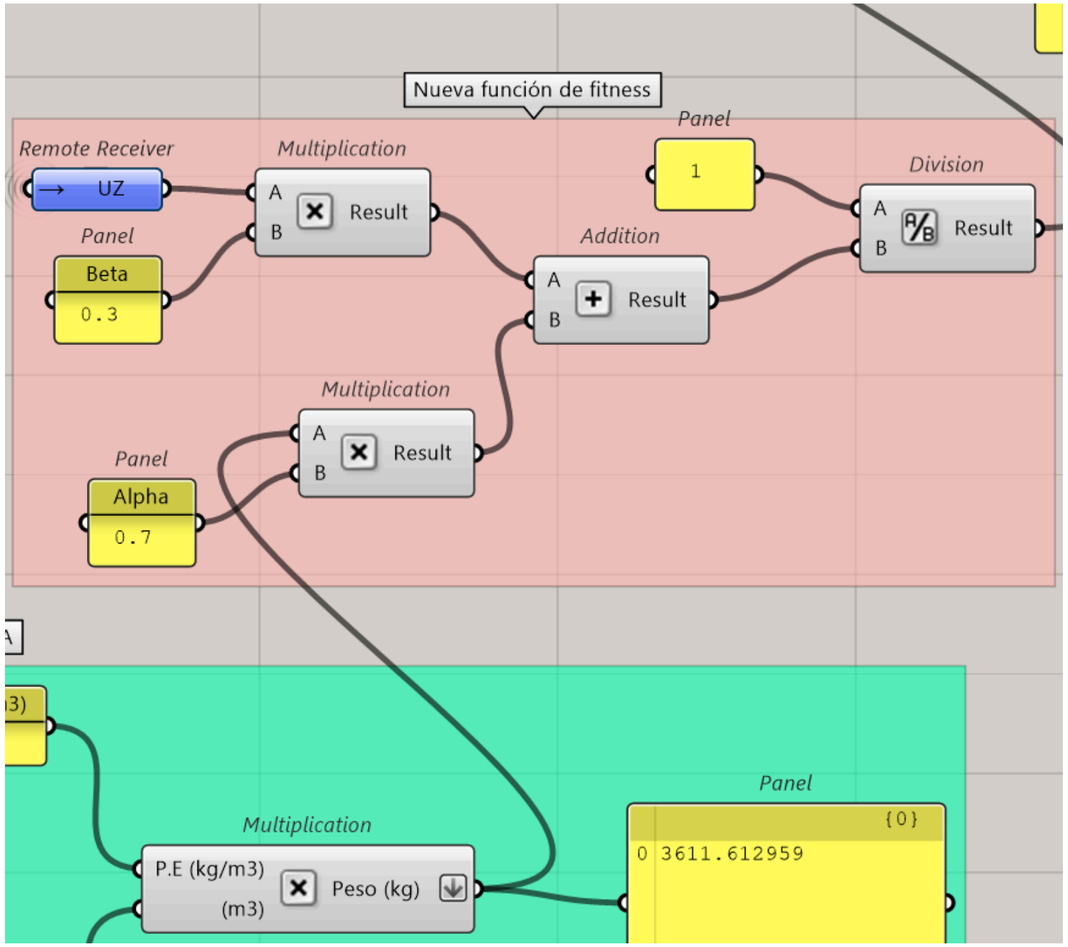
CASO 4: CAMBIO DE LA FUNCIÓN DE FITNESS CON MÉTODO DE PESOS PONDERADOS.			
PRUEBA 1			
CARACTERÍSTICAS			
Nº INDIVIDUOS	3		
Nº GENERACIONES TOTAL	3		
MEJOR INDIVIDUO DE LA PRUEBA	330	2	7,91E-04
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	330	2	7,91E-04
1	330	2	7,91E-04
2	330	2	7,91E-04
3	330	2	7,91E-04
PRUEBA 2			
CARACTERÍSTICAS			
Nº INDIVIDUOS	5		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	300	2	9,20E-04
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	360	2	6,81E-04
1	360	2	6,81E-04
2	360	2	6,81E-04
3	300	2	9,20E-04
4	300	2	9,20E-04



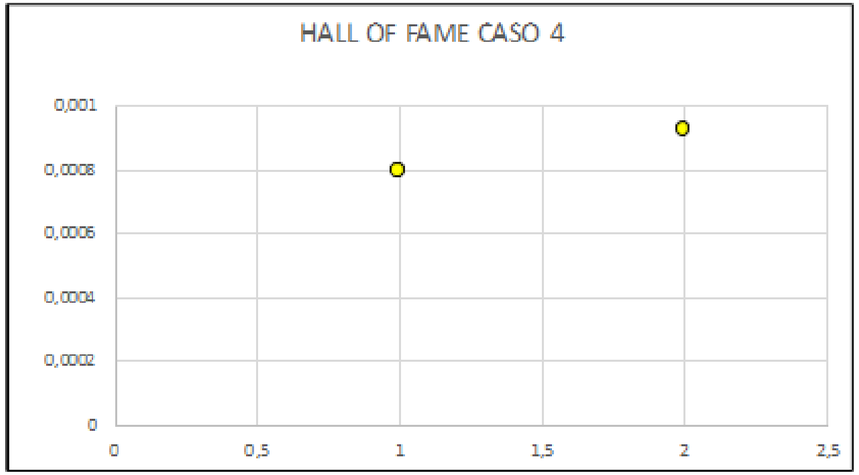


Resultados y análisis

CASO 4. CAMBIO EN LA FUNCIÓN DE FITNESS CON MÉTODO DE PESOS PONDERADOS.
alpha (peso) = 0,7 beta (deformación) = 0,3



MEJORES INDIVIDUOS DE CADA PRUEBA DEL CASO			
PRUEBA	IPE	NDIV	FITNESS
1	330	2	7,91E-04
2	**300**	**2**	**9,20E-04**



Resultados y análisis

CASO 5. CAMBIO EN LA FUNCIÓN DE FITNESS CON MÉTODO DE PESOS PONDERADOS Y MODIFICACIÓN DE LA LISTA DE PERFILES IPE.

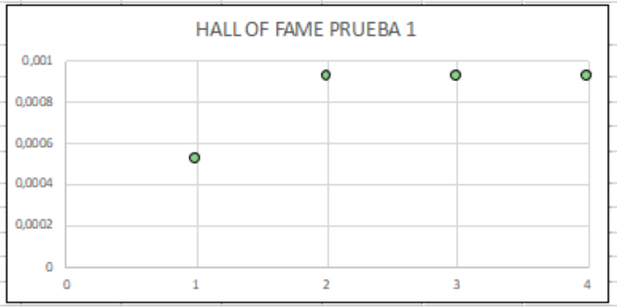
alpha (peso) = 0,7 beta (deformación) = 0,3

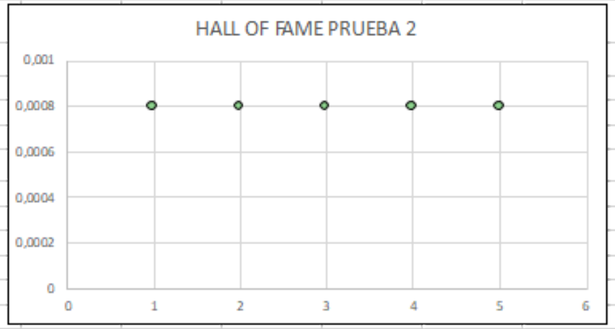
CASO 5: CAMBIO DE LA LISTA DE PERFILES IPE + CAMBIO FITNESS PASO 4

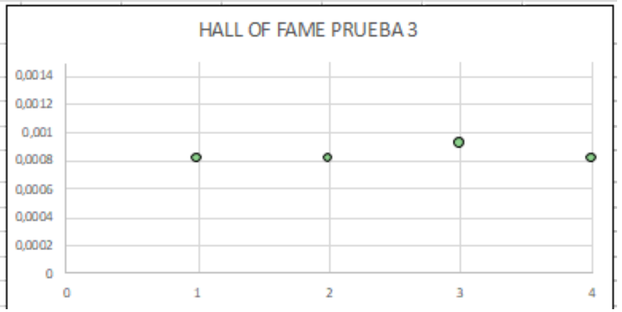
PRUEBA 1			
CARACTERÍSTICAS			
Nº INDIVIDUOS	3		
Nº GENERACIONES TOTAL	3		
MEJOR INDIVIDUO DE LA PRUEBA	240	4	9,22E-04
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	270	9	5,21E-04
1	240	4	9,22E-04
2	240	4	9,22E-04
3	240	4	9,22E-04

PRUEBA 2			
CARACTERÍSTICAS			
Nº INDIVIDUOS	5		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	240	6	7,96E-04
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	240	6	7,96E-04
1	240	6	7,96E-04
2	240	6	7,96E-04
3	240	6	7,96E-04
4	240	6	7,96E-04

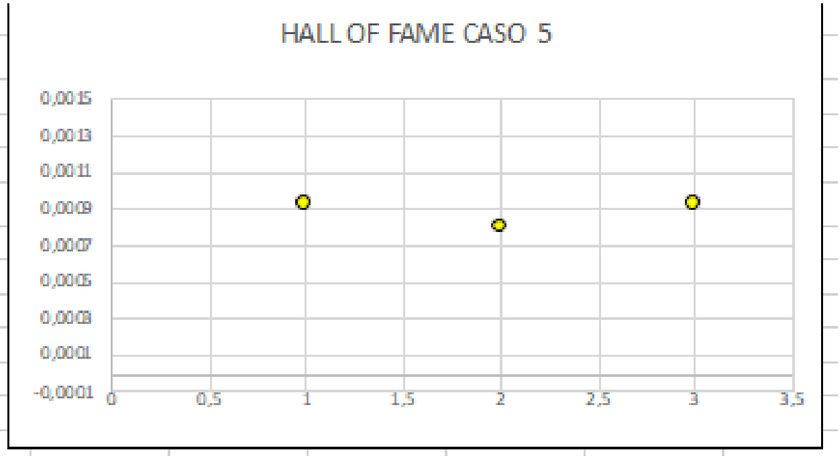
PRUEBA 3			
CARACTERÍSTICAS			
Nº INDIVIDUOS	4		
Nº GENERACIONES TOTAL	3		
MEJOR INDIVIDUO DE LA PRUEBA	240	4	9,22E-04
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	220	8	8,08E-04
1	220	8	8,08E-04
2	240	4	9,22E-04
3	220	8	8,08E-04







MEJORES INDIVIDUOS DE CADA PRUEBA DEL CASO			
PRUEBA	IPE	NDIV	FITNESS
1	240	4	9,22E-04
2	240	6	7,96E-04
3	240	4	9,22E-04



Resultados y análisis

CASO 5. CAMBIO EN LA FUNCIÓN DE FITNESS CON MÉTODO DE PESOS PONDERADOS Y MODIFICACIÓN DE LA LISTA DE PERFILES IPE.
alpha (peso) = 0,7 beta (deformación) = 0,3

Cambio en script de población inicial.

```python
# requirements: deap
# requirements: random2
import random
from deap import base, creator, tools

# Definir la lista de perfiles IPE
ipe_profiles = ["IPE 200", "IPE 220", "IPE 240", "IPE 270", "IPE 300", "IPE 330", "IPE 360", "IPE 400"]

# Función para crear un individuo
def create_individual():
    profile = random.choice(ipe_profiles)
    divisions = random.randint(2, 9) #min nº de div = 2 para que tengas un nodo con uz distinto de 0!
    return [profile, divisions]

# Definir el tipo de individuo en DEAP
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)  #### LIST EN LUGAR DE TUPLE

# Registrar la función para crear individuos en el toolbox
toolbox = base.Toolbox()
toolbox.register("individual", tools.initIterate, creator.Individual, create_individual)

# Registrar la población
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

# Generar la población
def generate_population(N_IND):
    return toolbox.population(n=N_IND)

# Generar la población y devolverla como lista
population = generate_population(N_IND)
```

Cambio script cálculo del peso.

Script Editor

Grasshopper Archivo Editar Ejecutar Herramientas Ventana Ayuda

```python
"""Grasshopper Script"""


if x == "IPE 200":
    a = 0
elif x == "IPE 220":
    a = 1
elif x == "IPE 240":
    a = 2
elif x == "IPE 270":
    a = 3
elif x == "IPE 300":
    a = 4
elif x == "IPE 330":
    a = 5
elif x == "IPE 360":
    a = 6
else:
    a = 7

print(a)
```

CONCLUSIÓN

Resumen de hallazgos

- Evolución desde vigas simples hasta modelos complejos.
- Herramientas para visualizar procesos de optimización.
- Desarrollo de habilidades de programación y de las API.
  - Integración de los algoritmos genéticos, calculo estructural avanzado y programación visual en un mismo entorno de trabajo.
- Formación continua y desarrollo profesional.

CONCLUSIÓN

Cumplimiento de objetivos

Comprender los fundamentos de los AG y programación visual.	Integrar 3 softwares distintos: RFEM, Grasshopper y Python.	Adecuada implementación de Algoritmos genéticos.
Automatizar el diseño estructural de vigas simples y celosías.	Validar la precisión y eficiencia de la aplicación.	Fomentar la innovación en este campo.

CONCLUSIÓN

Relevancia de las aportaciones

- Innovación en el diseño estructural.
  - Visualización intuitiva y ajustable.
  - Facilita el entendimiento y la colaboración.
- Simplifica procesos de trabajo y personalización de modelos.
- Ahorro de tiempo y recursos en cálculo estructural, obteniendo las estructuras más económicas.
- Impacto en la formación académica.
- Promoción de las nuevas tecnologías y herramientas avanzadas.
- Base para futuras investigaciones.

CONCLUSIÓN

Limitaciones para el estudio

- Capacidad computacional y compatibilidad de sistemas.
- Integración de herramientas y falta de desarrollo en el ámbito.
- Tiempo de pruebas y cargas de las iteraciones.
- Limitaciones temporales y circunstancias externas

CONCLUSIÓN

Sugerencias para futuras investigaciones

- Ampliar el proyecto a modelos en 3D.
- Mejorar la extracción del peso propio de la estructura.
- Añadir optmización topológica más avanzada.
- Mejorar aún más la calidad de la función de fitness.

Gracias por su atención

Sara Trinidad Quiñones

Universidad Loyola Andalucía (Sevilla)

strinidadquinones@al.uloyola.es
saratrinidadquinones@gmail.com