Problema 1

TP2: Algoritmos Genéticos

Problema

Optimizar el siguiente sistema:

$$F(W, w, w_0, \xi) = g(\sum_{j=1}^{2} W_j g(\sum_{k=1}^{3} w_{jk}\xi_k - w_{j0}) - W_0)$$

- 11 números reales

Problema

```
dataset = [
    [(4.4793, -4.075, -4.075), 0],
    [(-4.1793, -4.9218, 1.7664), 1],
    [(-3.9439, -0.7689, 4.8830), 1],
]


def g(x):
    try:
        return math.e**x/(1+math.e**x)
    except OverflowError:
        return 1


def F(W, w, w0, E):
    return g(sum(
        W[j+1]*g(sum(
            w[j][k]*E[k]
            for k in range(0, 3)
        ) - w0[j])
        for j in range(0, 2)
    ) - W[0])


def E(W, w, w0):
    return sum((OUT - F(W, w, w0, IN))**2 for (IN, OUT) in dataset)
```

```
def error(genotype):
    W = (genotype[0:3])
    w = ((genotype[3:6]), (genotype[6:9]))
    w0 = (genotype[9:11])
    return E(W, w, w0)
```

Resultado Óptimo


```
{
    "generations": 21,
    "individual": {
        "W": [
            64.0528788936326,
            -298.04532992902136,
            101.25578699118827
        ],
        "w": [
            [
                34.592597995020924,
                10.456639892066423,
                -23.06847772526052
            ],
            [
                4.151287835307428,
                -1.0166116956699431,
                22.954148293892544
            ]
        ],
        "w0": [
            -12.38979828464492,
            -28.673389751460977
        ]
    },
    "F1": 5.530265749821902e-158,
    "F2": 1.0,
    "F3": 1.0,
    "E": 3.058383925e-315
}
```



```
{
    "parent_selection":"random",
    "selection":"direct",
    "stop_condition": "error",
    "error": 300,
    "population": 100,
    "mutation":0.5,
    "deviation":10,
    "seed": 1,
    "range":[-10,10]
}
```

Resultado Óptimo


```
{
    "generations": 30,
    "individual": {
        "W": [
            351.63532609328144,
            143.64297371865058,
            248.56271379039254
        ],
        "w": [
            [
                -74.65124478102081,
                18.71971154737944,
                18.15533598815387
            ],
            [
                -42.66036673067842,
                -33.07768739909587,
                32.316182524683676
            ]
        ],
        "w0": [
            -49.89514960253319,
            -0.4981423960474256
        ]
    },
    "F1": 1.9351660462344503e-153,
    "F2": 1.0,
    "F3": 1.0,
    "E": 3.7448676264986747e-306
}
```



```
{
    "parent_selection":"random",
    "selection":"roulette",
    "stop_condition": "error",
    "error": 300,
    "population": 100,
    "mutation":0.5,
    "deviation":10,
    "seed": 3,
    "range":[-10,10]
}
```

Resultado Óptimo


```
{
    "generations": 18,
    "individual": {
        "W": [
            65.4224613159826,
            -295.68306792454655,
            103.32524220041317
        ],
        "w": [
            [
                33.40153426304293,
                -7.269036269212831,
                -24.639946633602957
            ],
            [
                2.091921472827025,
                21.658802940023794,
                78.55124403886197
            ]
        ],
        "w0": [
            37.28696541770327,
            -29.069516211243585
        ]
    },
    "F1": 1.4923176175721325e-157,
    "F2": 1.0,
    "F3": 1.0,
    "E": 2.227011872e-314
}
```



```
{
    "parent_selection":"random",
    "selection":"boltzmann",
    "stop_condition": "error",
    "To" : 10,
    "Tc": 0.1,
    "k": 0.0077,
    "error": 300,
    "population": 100,
    "mutation":0.5,
    "deviation":10,
    "seed": 2,
    "range":[-10,10]
}
```

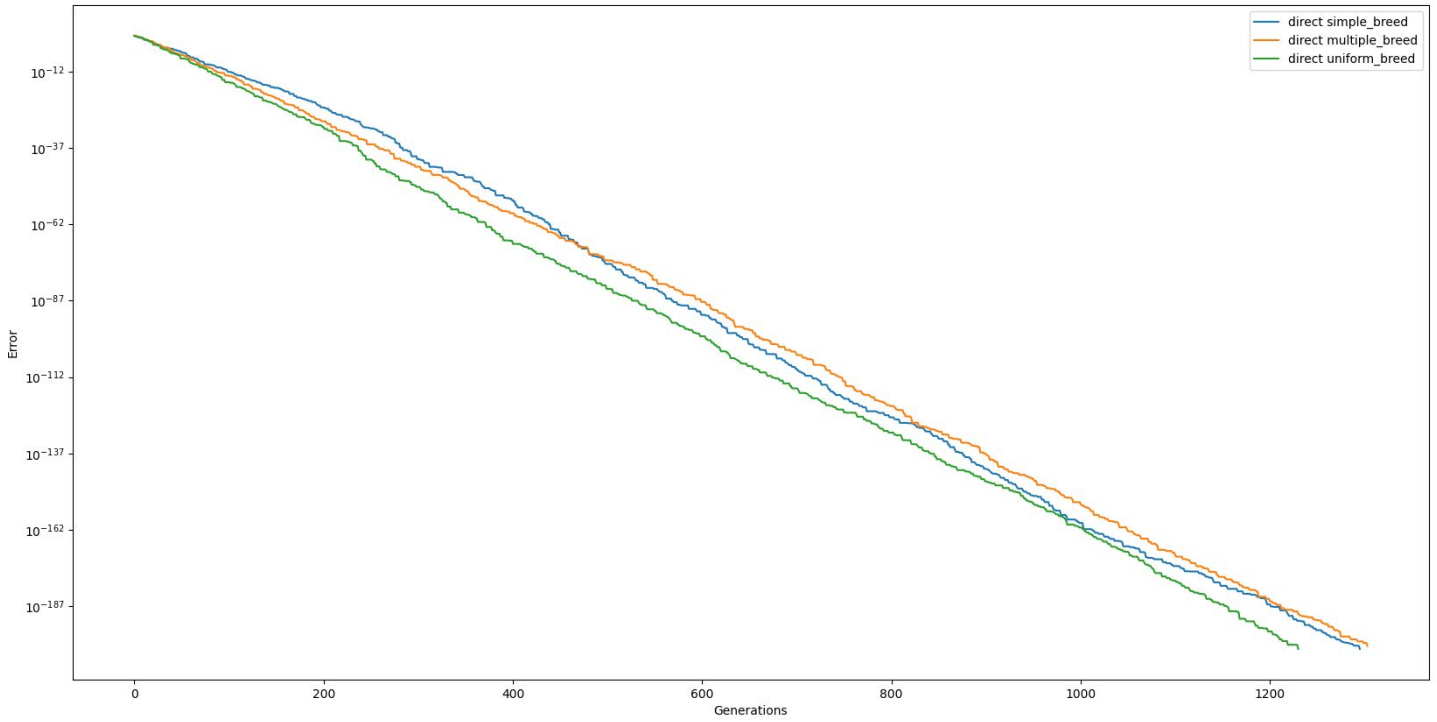
Fitness

- Queremos minimizar el error
- El error va de 0 a 3
- ¿Qué función de fitness nos conviene?
  - No elegimos 3 - E ya que mientras más achicamos el error, menos significancia tiene este
  - 1/E nos sirve ya que mientras más chico es el E, más impacta en la función (cuidado con error 0)

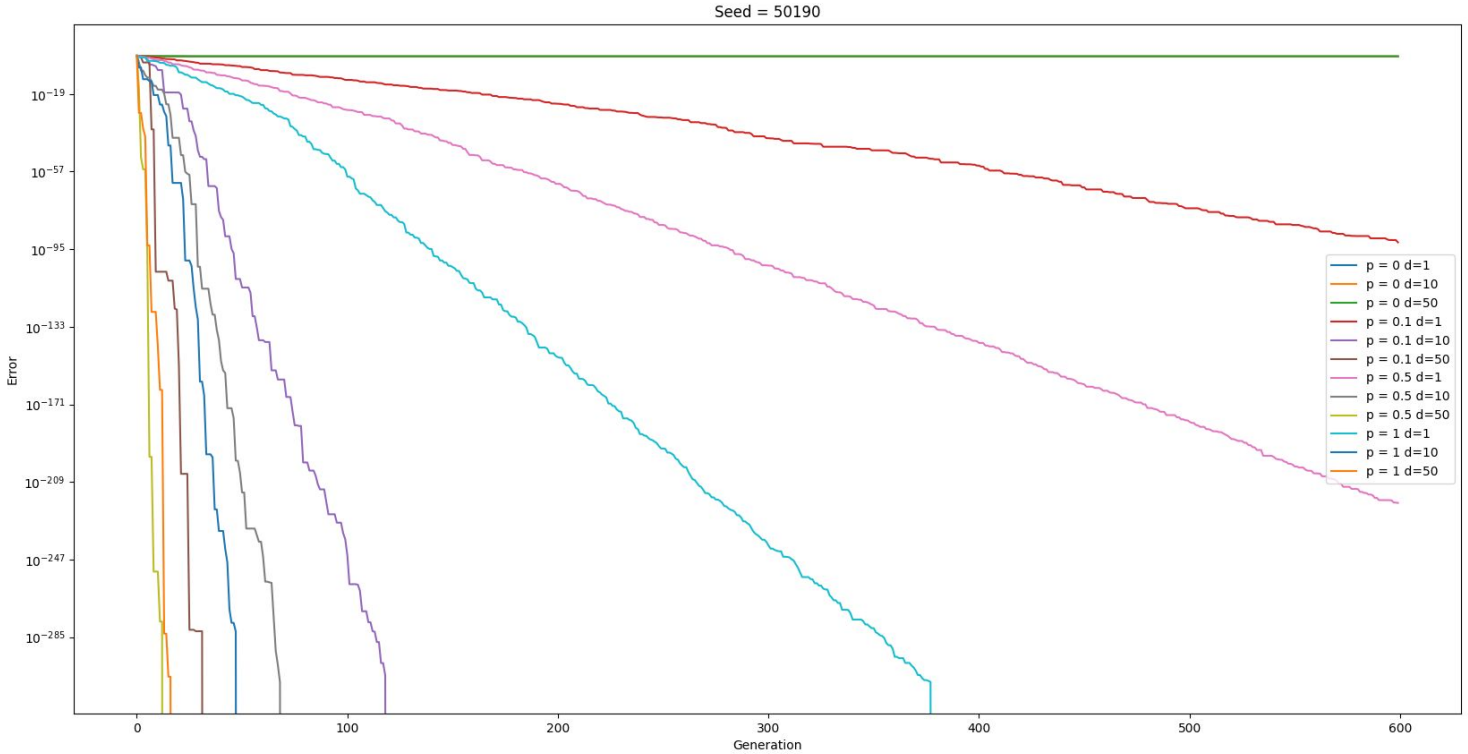
Parámetros constantes para los gráficos

- Población: 100
- Rango de población inicial: [0, 1]
- Mutación: 0.1
- desviación: 1
- Selección de padres random con reemplazo

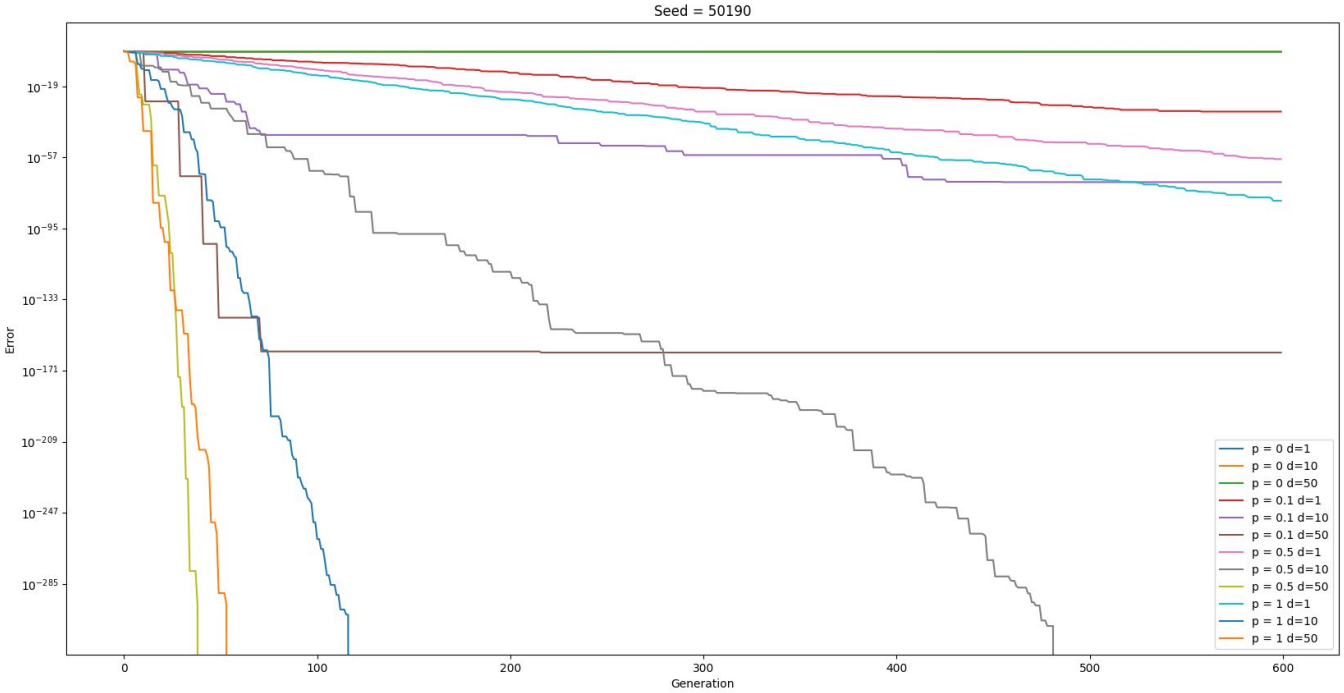
ELITE generation = 600



ELITE - mutación



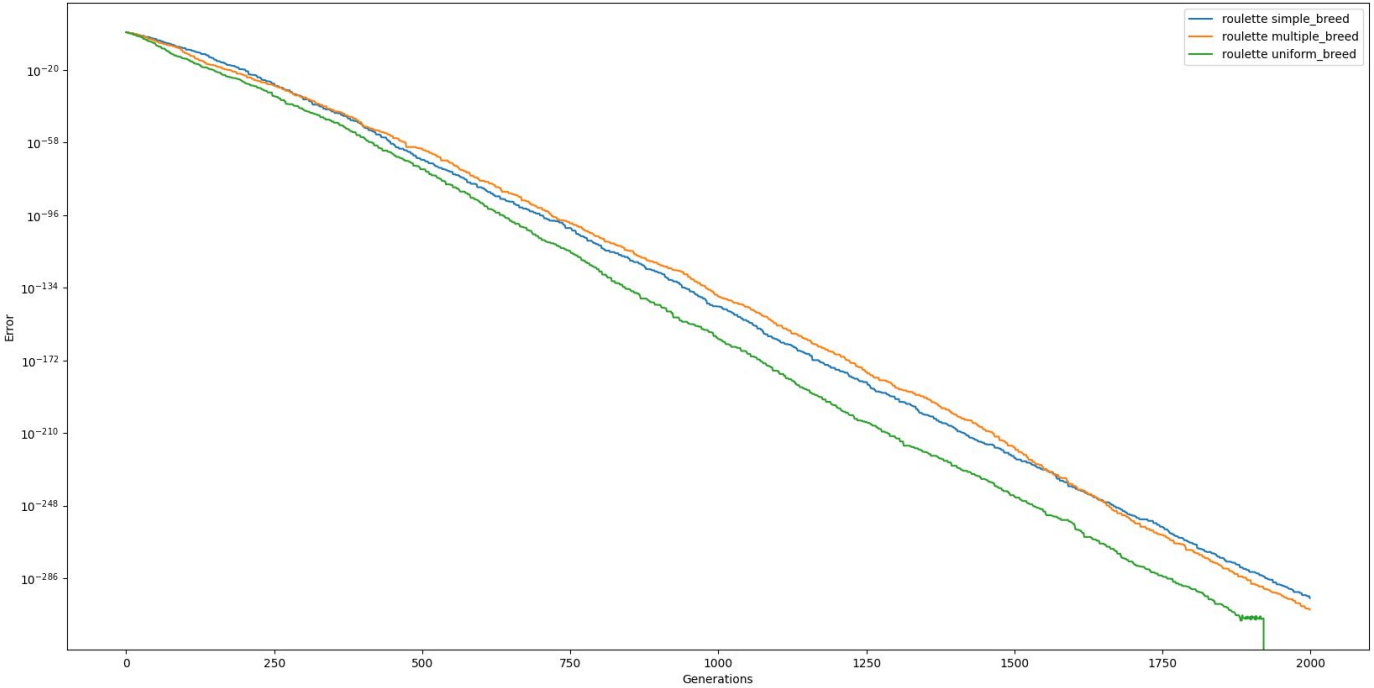
ELITE - Mutations one



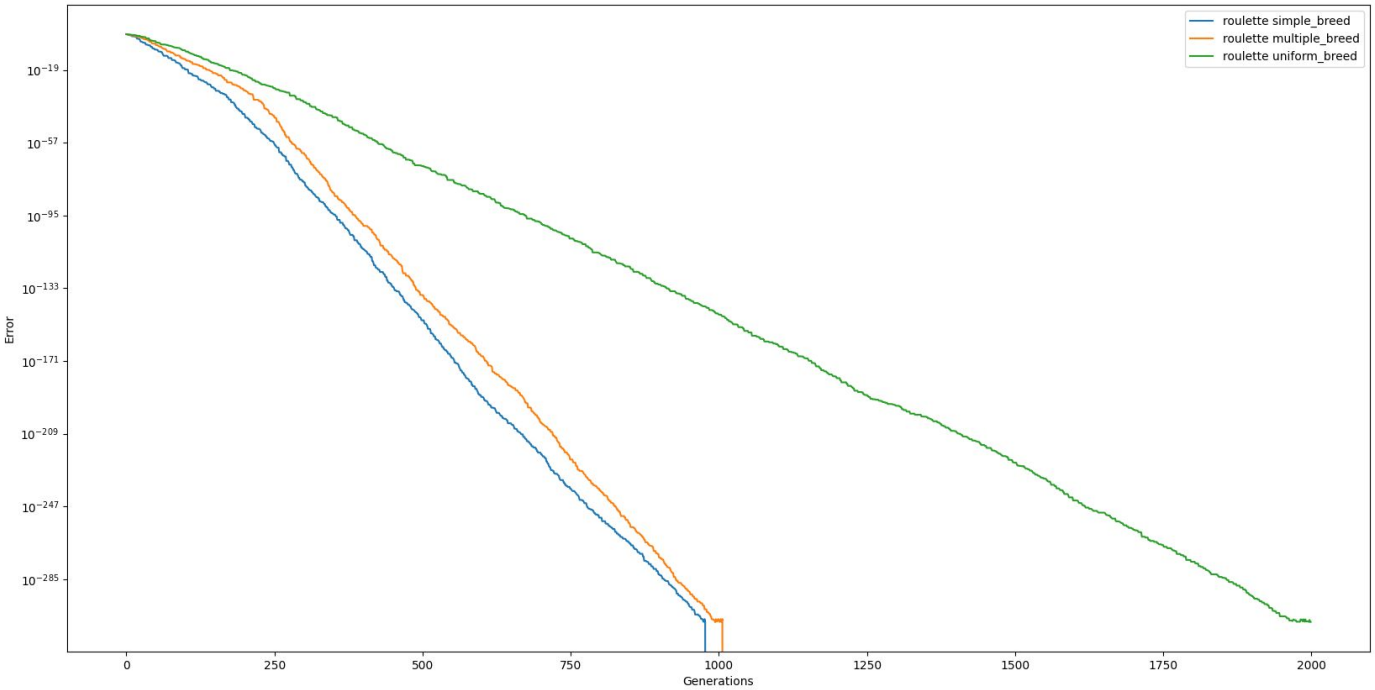
Conclusiones Elite

- Llega rápido a un error muy bajo
- Si el problema tuviese más máximos locales podría fallar más
- Cuanta más mutación, se puede observar que llega más rápido a converger. También tiene menos chances de estancarse en máximos locales

Ruleta


roulette simple_breed
roulette multiple_breed
roulette uniform_breed
Error
10^{-20}
10^{-58}
10^{-96}
10^{-134}
10^{-172}
10^{-210}
10^{-248}
10^{-286}
0
250
500
750
1000
1250
1500
1750
2000
Generations

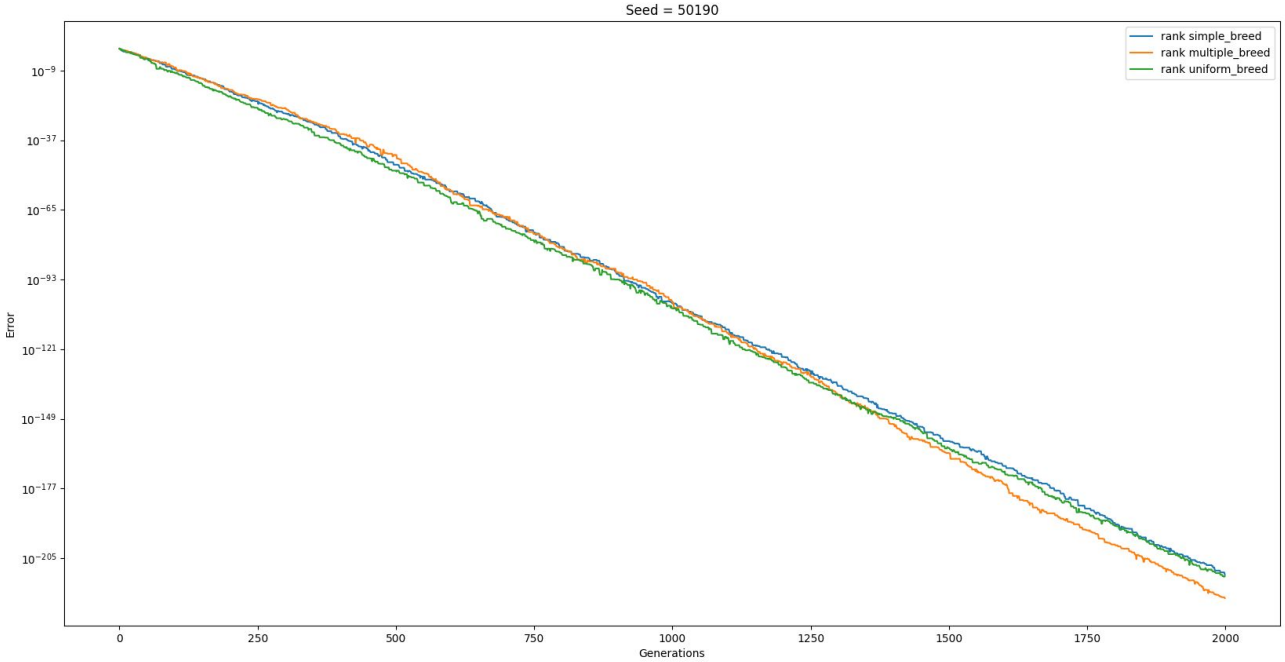
Ruleta


roulette simple_breed
roulette multiple_breed
roulette uniform_breed
Error
10^{-19}
10^{-57}
10^{-95}
10^{-133}
10^{-171}
10^{-209}
10^{-247}
10^{-285}
0
250
500
750
1000
1250
1500
1750
2000
Generations

Conclusiones Ruleta

- No hay un mejor método de cruza para todos los cazos, como se ve en ambos ejemplos, el resultado de cada uno cambia mucho
- Sin embargo, la cruza uniforme es más consistente, ya que mantiene en todos casos un alto nivel de mezcla
- Podemos ver como es más lento que el elite ya que no siempre elige los mejores, pero ayuda a una búsqueda más diversa.

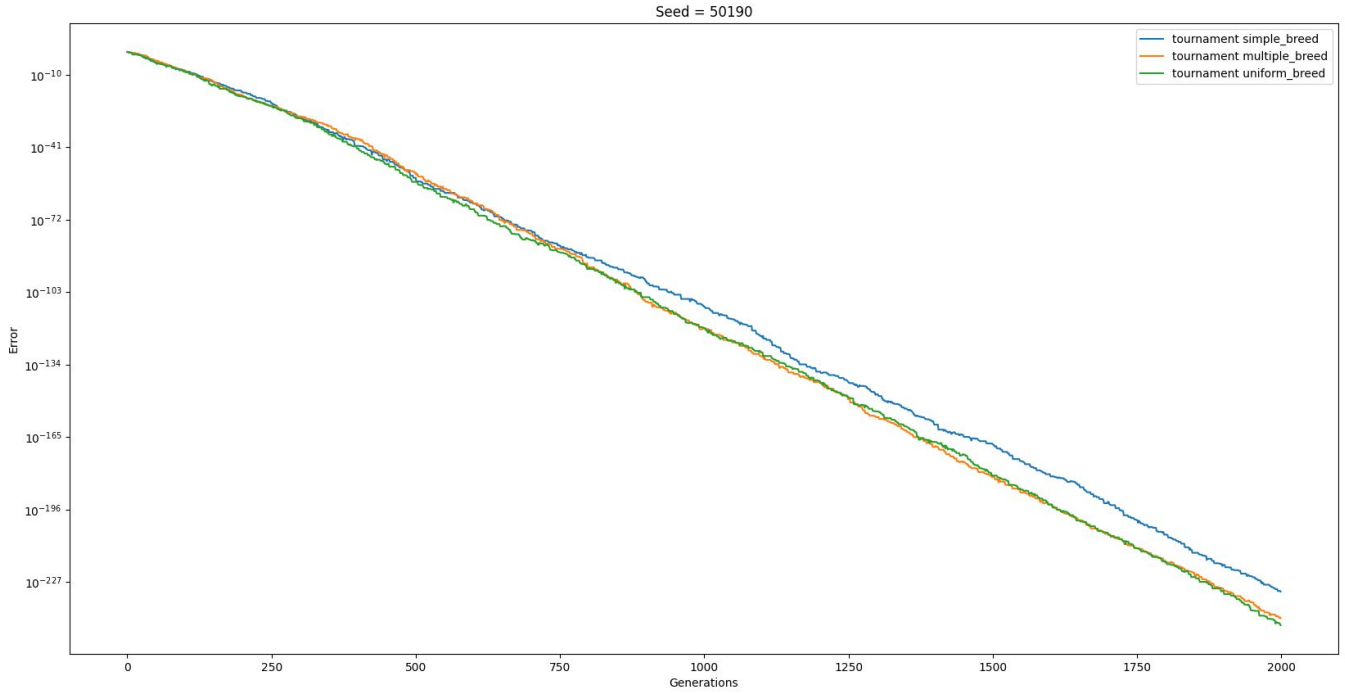
Rank


Seed = 50190
rank simple_breed
rank multiple_breed
rank uniform_breed
Error
10^{-9}
10^{-37}
10^{-65}
10^{-93}
10^{-121}
10^{-149}
10^{-177}
10^{-205}
0
250
500
750
1000
1250
1500
1750
2000
Generations

Conclusiones Rank

- Volvemos a ver como los métodos de cruza tienen poca importancia
- Tenemos una pendiente aún menos pronunciada que en el método ruleta
- Menor pendiente debido a probabilidad basada en el indice una vez calculado el fitness

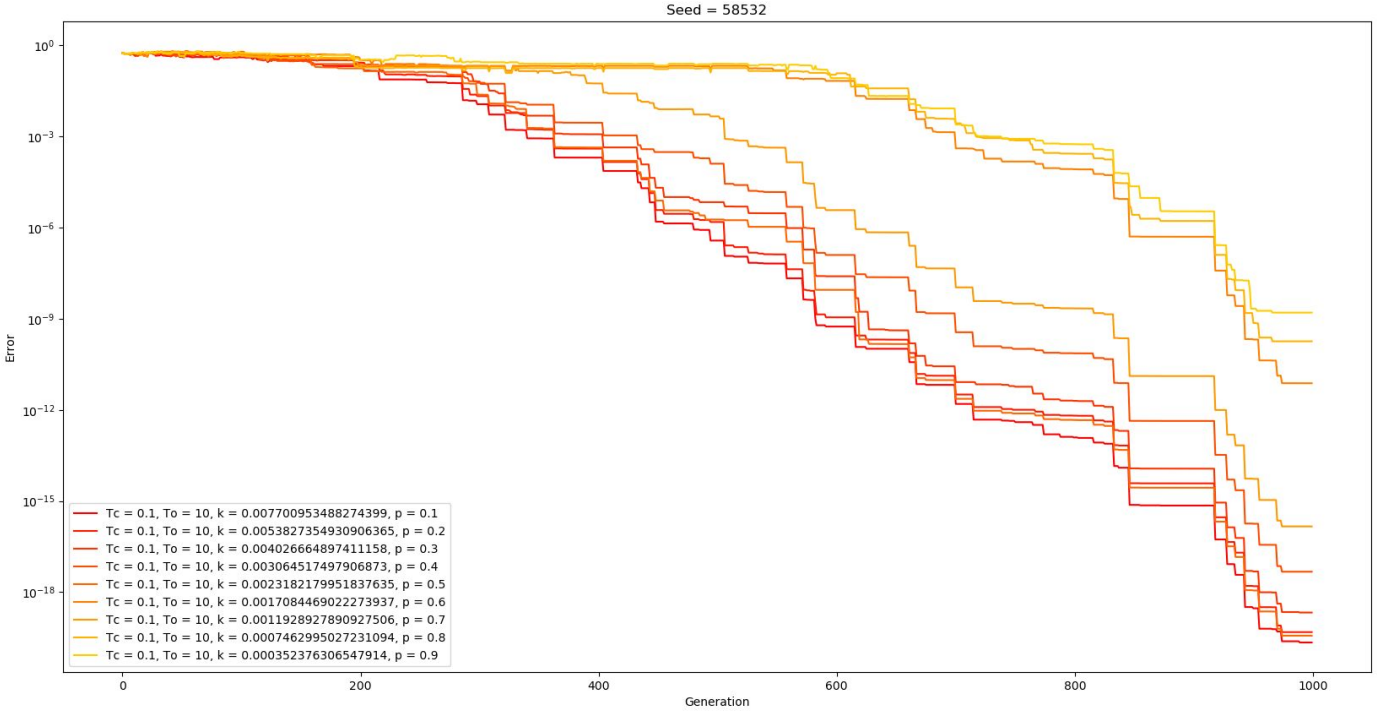
Torneos


Seed = 50190
tournament simple_breed
tournament multiple_breed
tournament uniform_breed
Error
10^{-10}
10^{-41}
10^{-72}
10^{-103}
10^{-134}
10^{-165}
10^{-196}
10^{-227}
0
250
500
750
1000
1250
1500
1750
2000
Generations

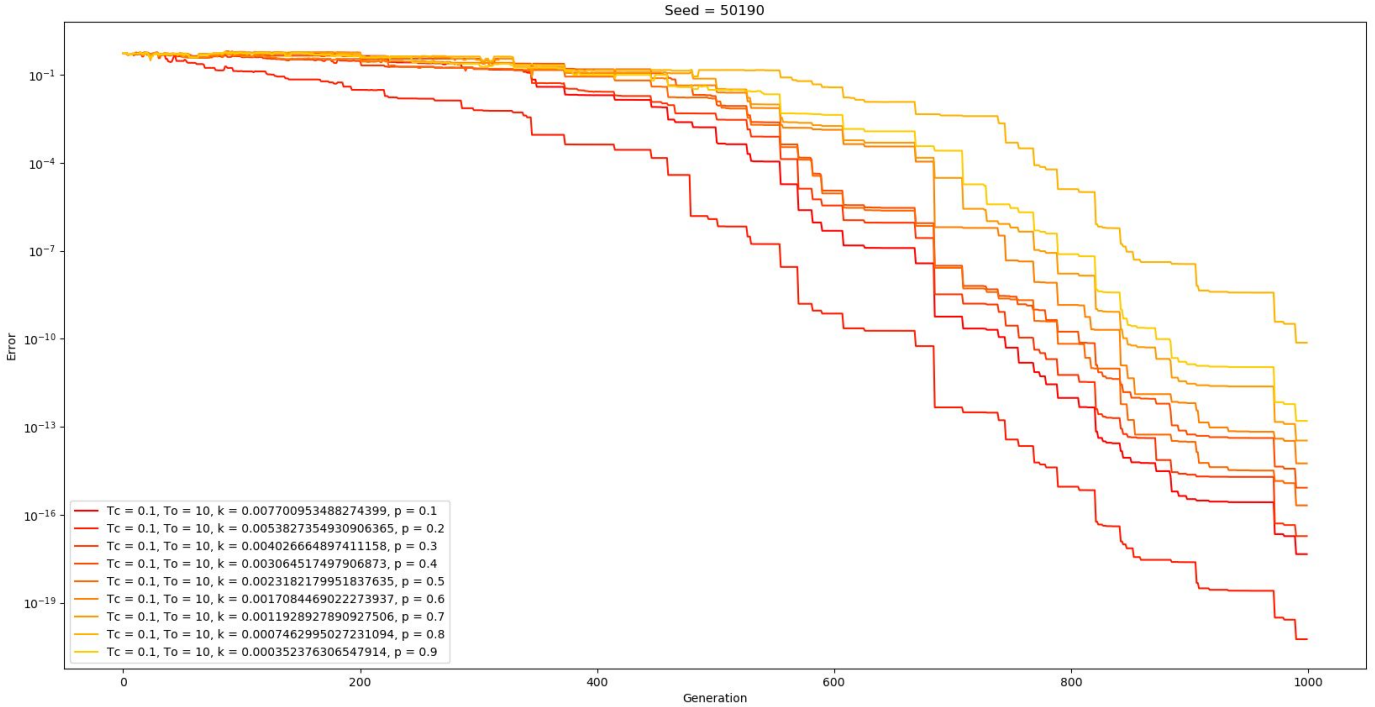
Conclusiones Torneos

- Volvemos a ver baja importancia en los algoritmos de cruza
- Mejor pendiente que en el rank, ya que entre los 4 seleccionados, para que salga triunfante el peor la probabilidad es muy baja ya que tiene que ganar dos batallas

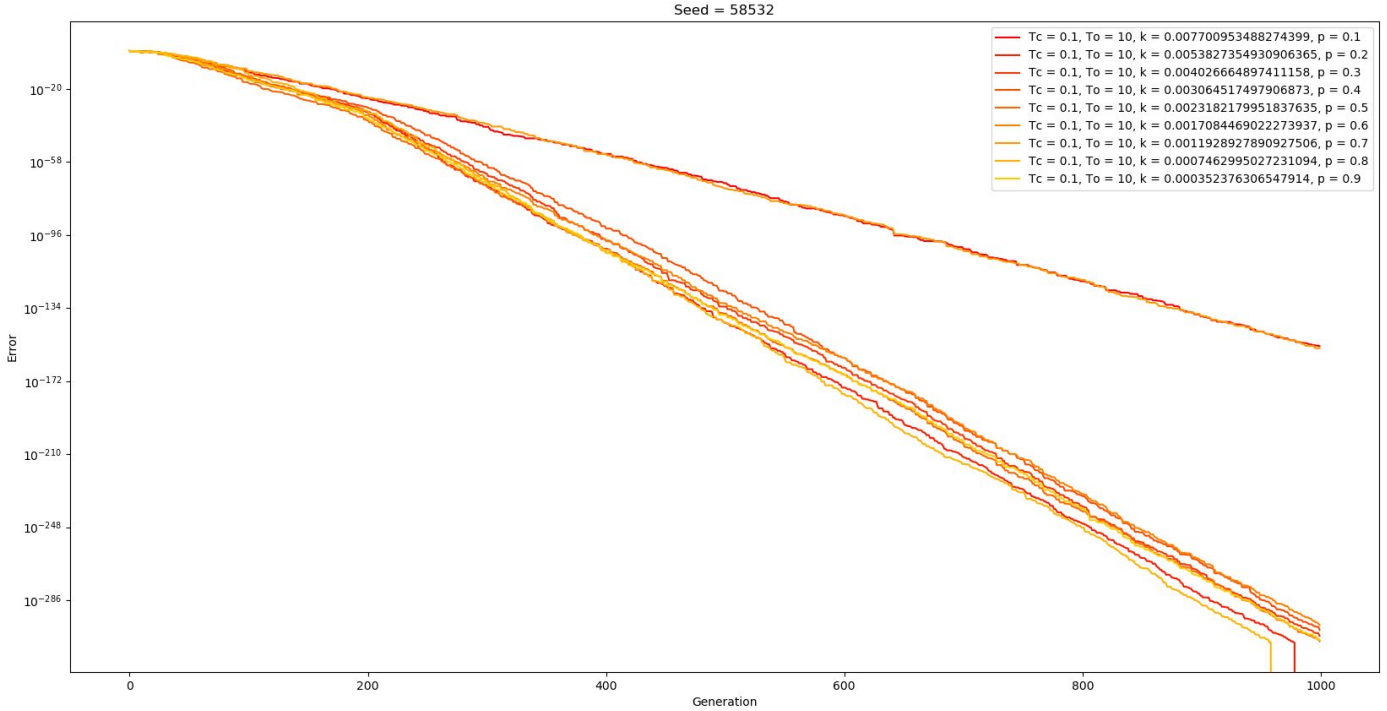
Boltzman - M = 0.001



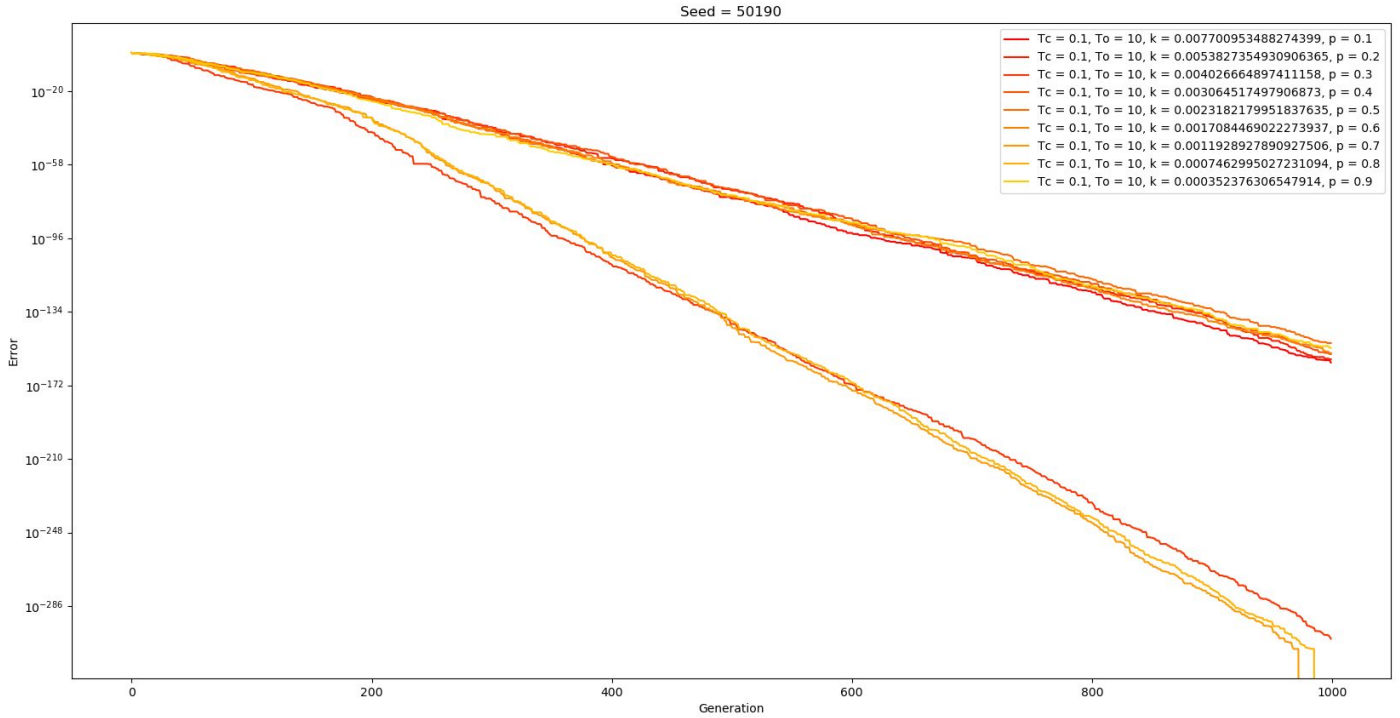
Boltzman - M = 0.001



Boltzman

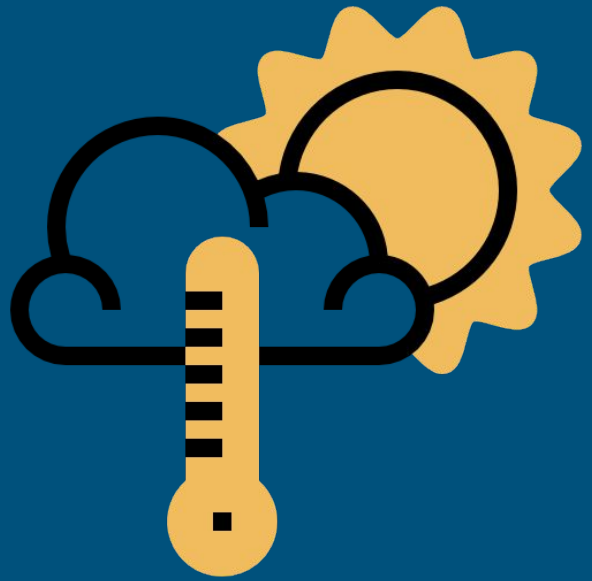

Seed = 58532
Tc = 0.1, To = 10, k = 0.007700953488274399, p = 0.1
Tc = 0.1, To = 10, k = 0.005382735493090636, p = 0.2
Tc = 0.1, To = 10, k = 0.004026664897411158, p = 0.3
Tc = 0.1, To = 10, k = 0.003064517497906873, p = 0.4
Tc = 0.1, To = 10, k = 0.002318217995183763, p = 0.5
Tc = 0.1, To = 10, k = 0.0017084469022273937, p = 0.6
Tc = 0.1, To = 10, k = 0.0011928927890927506, p = 0.7
Tc = 0.1, To = 10, k = 0.0007462995027231094, p = 0.8
Tc = 0.1, To = 10, k = 0.000352376306547914, p = 0.9
Error
10^{-20}
10^{-58}
10^{-96}
10^{-134}
10^{-172}
10^{-210}
10^{-248}
10^{-286}
0
200
400
600
800
1000
Generation

Boltzman

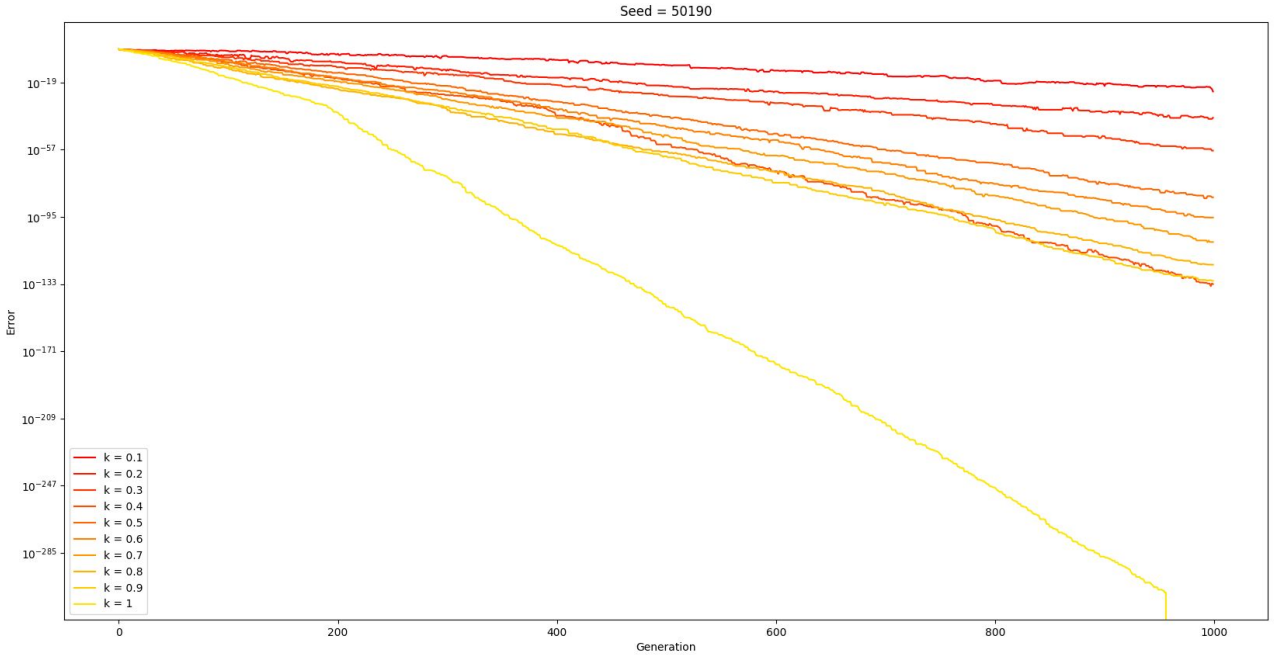

Seed = 50190
Tc = 0.1, To = 10, k = 0.007700953488274399, p = 0.1
Tc = 0.1, To = 10, k = 0.0053827354930906365, p = 0.2
Tc = 0.1, To = 10, k = 0.004026664897411158, p = 0.3
Tc = 0.1, To = 10, k = 0.003064517497906873, p = 0.4
Tc = 0.1, To = 10, k = 0.0023182179951837635, p = 0.5
Tc = 0.1, To = 10, k = 0.0017084469022273937, p = 0.6
Tc = 0.1, To = 10, k = 0.0011928927890927506, p = 0.7
Tc = 0.1, To = 10, k = 0.0007462995027231094, p = 0.8
Tc = 0.1, To = 10, k = 0.0003523763065479 14, p = 0.9
Error
10^{-20}
10^{-58}
10^{-96}
10^{-134}
10^{-172}
10^{-210}
10^{-248}
10^{-286}
0
200
400
600
800
1000
Generation

Boltzman conclusiones

- A mayor mutación, nos conviene elegir un k menor para que sea más elitista y elegir la mutación que nos lleve más rápido a la solución
- Esto conviene si el problema es simple (poco riesgo de caer en máximos locales)
- Nos quedamos con To =10, Tc =0.1 y k=0.0077



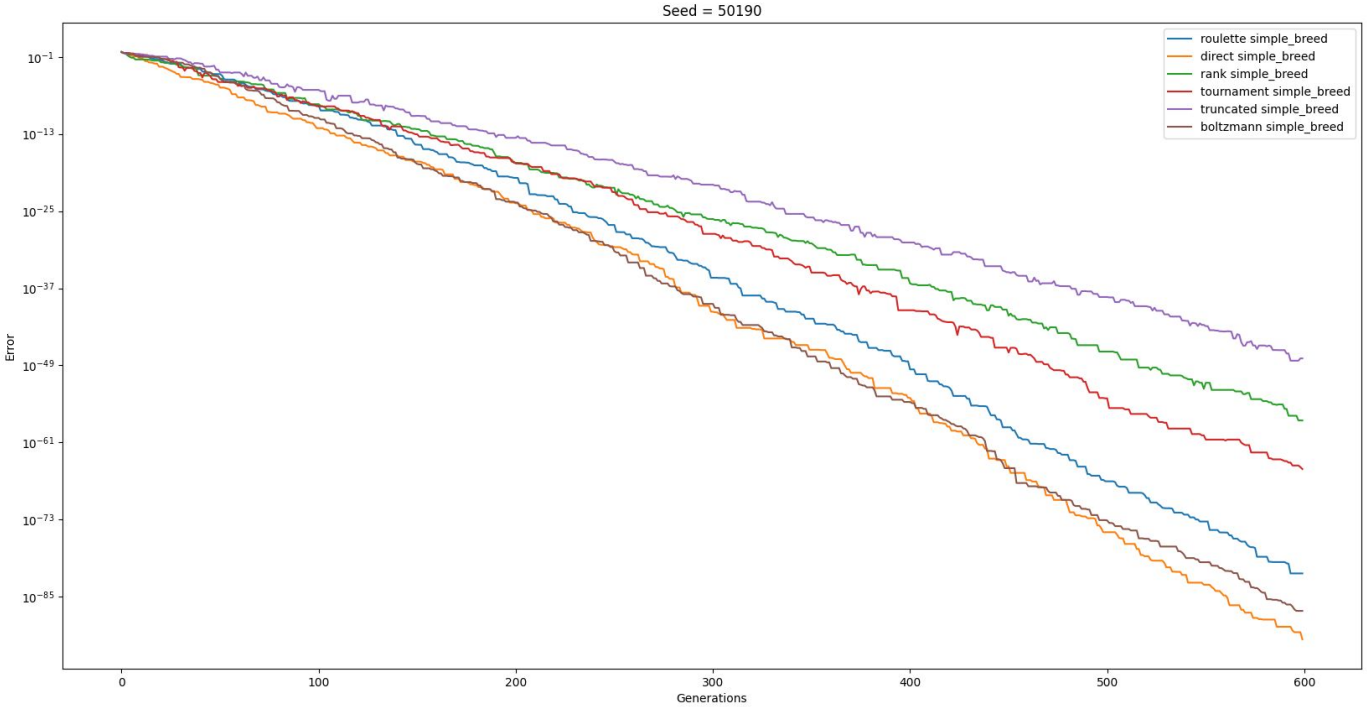
Selección truncada



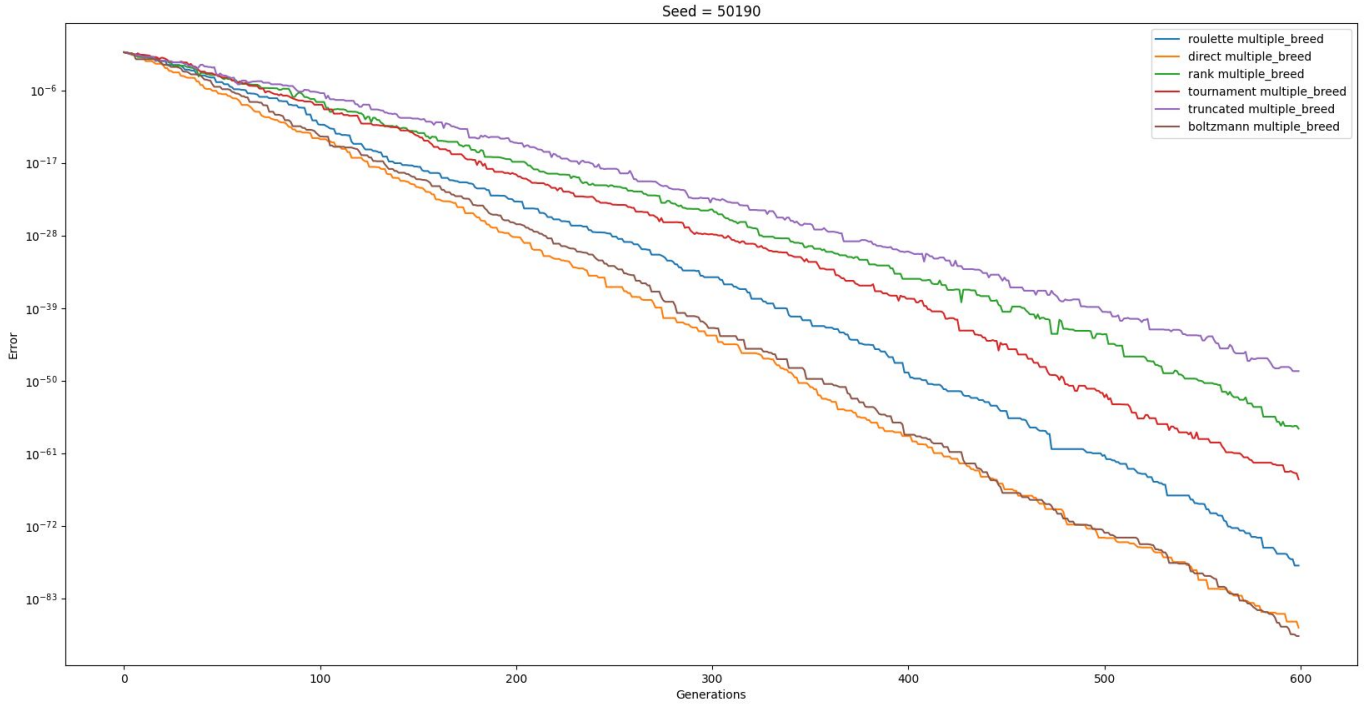
Selección truncada conclusiones

- Como se puede ver cuando trunca la mitad de la población este método se vuelve igual al elite y se puede observar su claro incremento en pendiente
- Al truncar menos, caemos menos en los máximos locales
- Elegimos una truncación de 50, por ende tenemos un random entre los 150 mejores

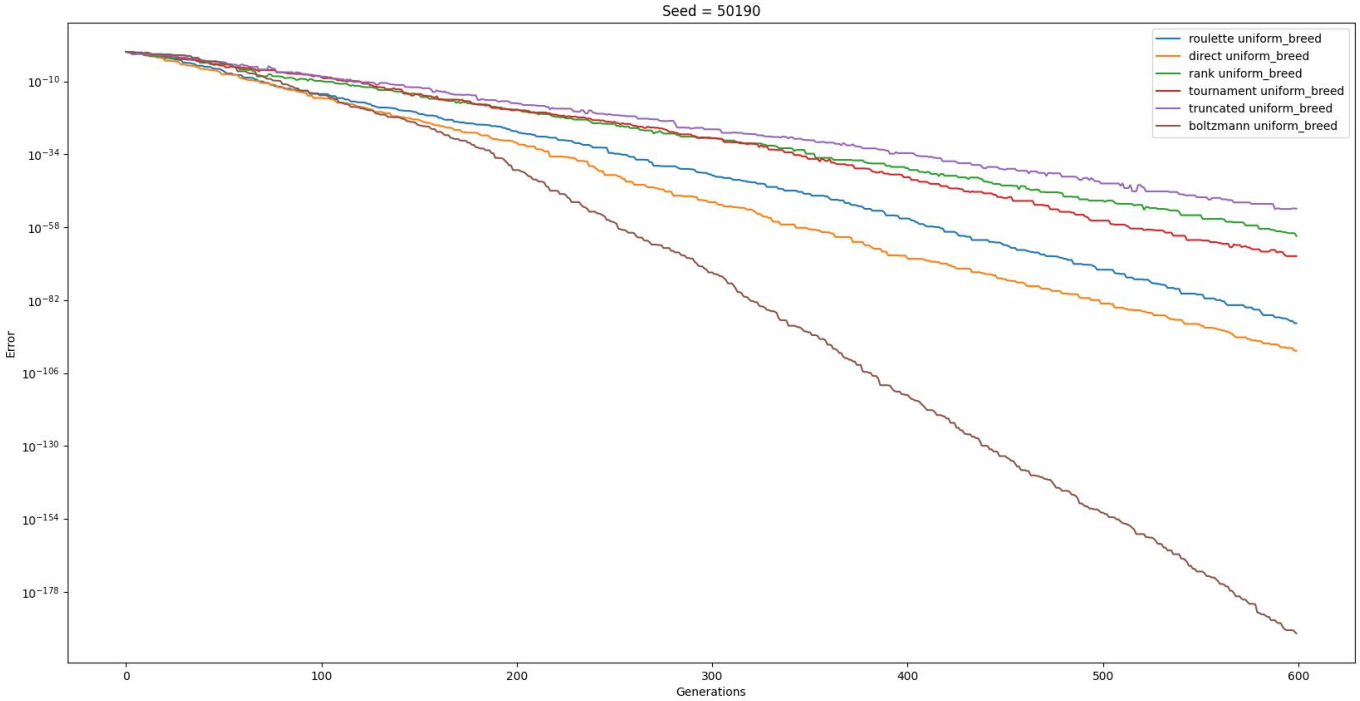
Generations = 600- Simple Breed



Generations = 600 - Multiple Breed



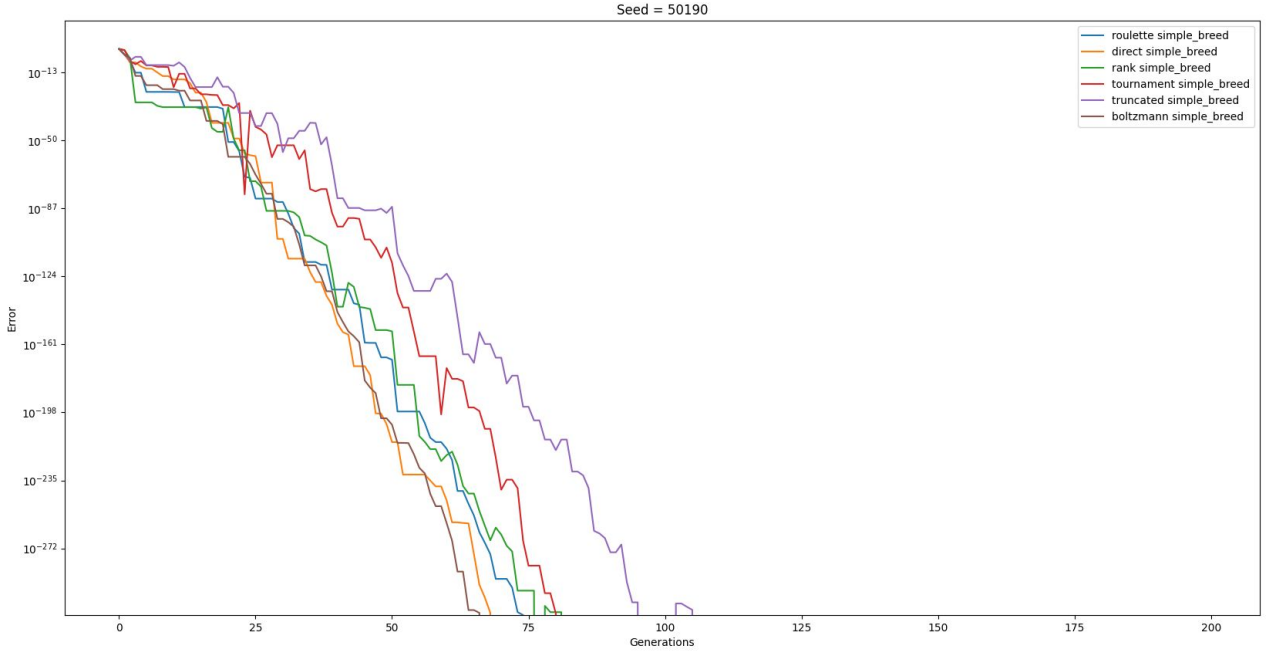
Generations = 600 - Uniform Breed



Conclusiones - diferentes breeds

- Como vemos los breeds no cambian en general el comportamiento del sistema, ya que los distintos métodos de selección mantienen un orden para los distintos breeds
- También vemos como los métodos más elitistas llegan en general a una mejor solución, esto es indicio de una baja dificultad en el problema

Población inicial chica, mutación alta

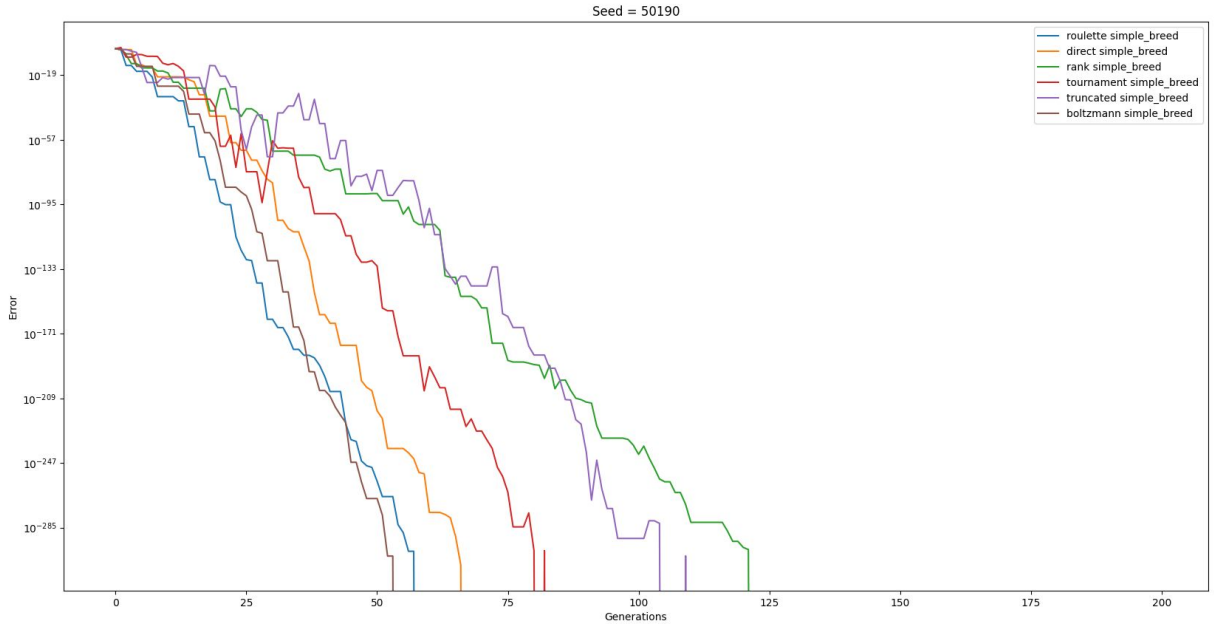


Población inicial: [0, 1]

Mutación: 0.5

desviación: 5

Población inicial dispersa - mutación alta

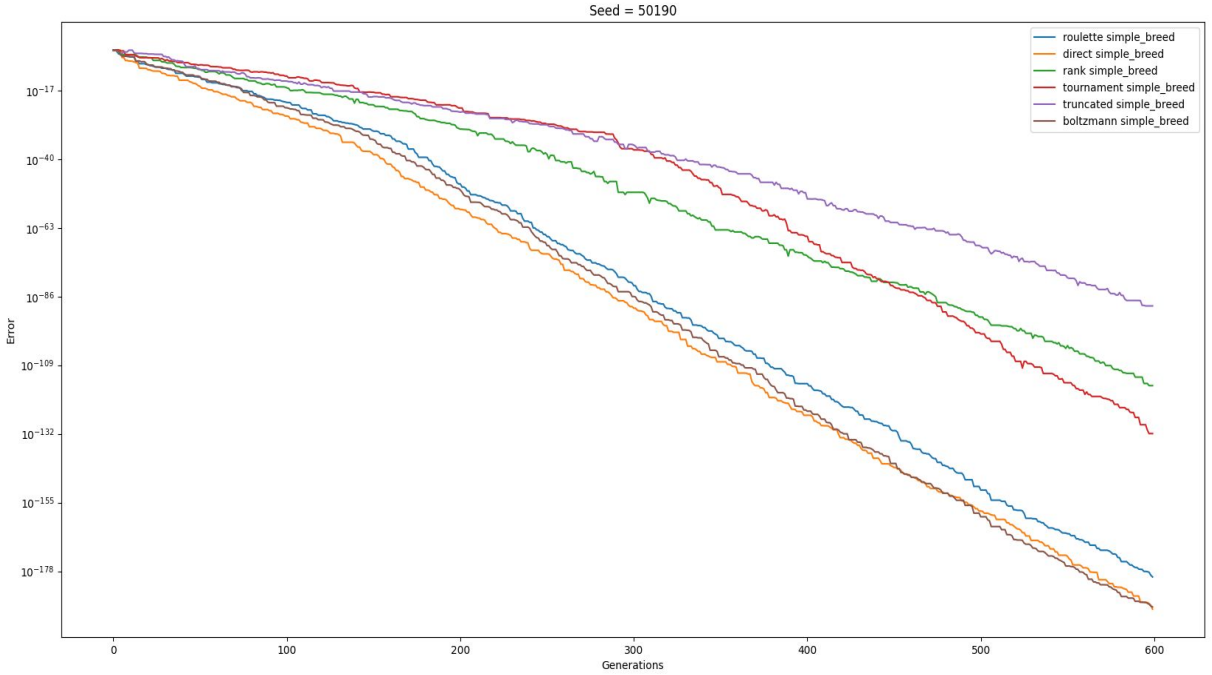


Población inicial: [-10, 10]

Mutación: 0.5

desviación: 5

Población inicial dispersa - mutación default



Población inicial: [-10, 10]

Mutación: 0.1

desviación: 1

Conclusiones generales

- Al incrementar la mutación se obtienen mejores resultados ya que no se cae en máximos locales.
- El rango de la población inicial no modifica mucho el resultado
- Mientras más simple es el problema, menos diversidad es necesaria
- Al ser un problema simple, el método elitista nos sirve para llegar al error mínimo ya que hay poco riesgo de caer en máximos locales