# Memoria Práctica 1(G07P1)

Integrante: Gallardo Cruzado Mario Daniel

Cobertura: pr1 básica + cruce BLX- + 5 funciones(DeJong n5, Easom, CrossIT, Drop wave, Bukin n6)

Github: <https://github.com/danisilver/pevolutiva>

## Instrucciones de Uso

Atención: si se quiere fijar una semilla propia para la generación de números pseudoaleatorios, hay que desactivar la casilla random seed, pegar y seguidamente presionar enter en la casilla.

En el panel izquierdo tenemos las opciones, en el panel General seleccionamos en el desplegable de funciones, la que deseamos ejecutar.

En el centro tenemos 4 pestañas, [grafico de] Aprendizaje, Problem View, Log, Ayuda

En el gráfico de aprendizaje vemos la evolución del algoritmo genético

En Problem view, vemos la interfaz del problema, en esta práctica disponemos de la visualización 3d de las funciones, con doble click podemos hacer rotar la gráfica 3d, y manteniendo click rotamos manualmente.

En Log, se muestran los resultados a medida que progresa el algoritmo.

En ayuda se pretende poner una descripción de las opciones, pero está incompleto a falta de tiempo

### Atajos de teclado:

Alt+j Ejecutar

Alt+r Limpiar log

Alt+d Random

Alt+f Elegir función

Alt+a Elegir tipo cromosoma

Alt+s Elegir selección

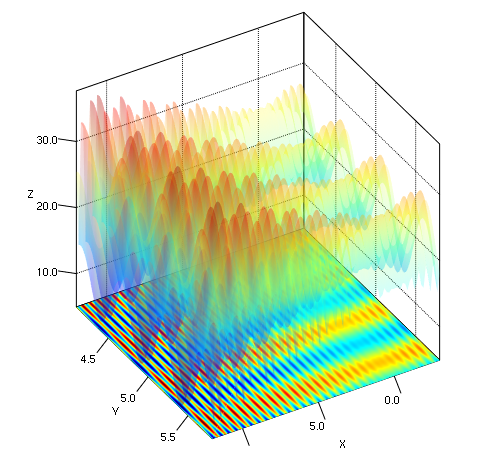
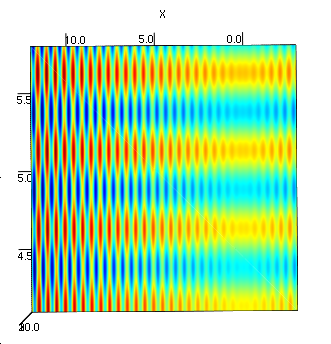
Alt+c Elegir cruce

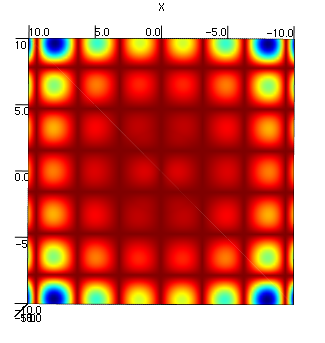
Alt+m Elegir mutación

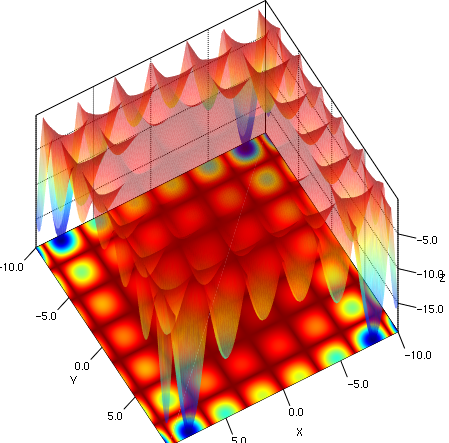
## Vista previa

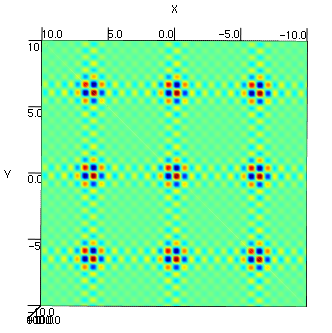
Función 1:

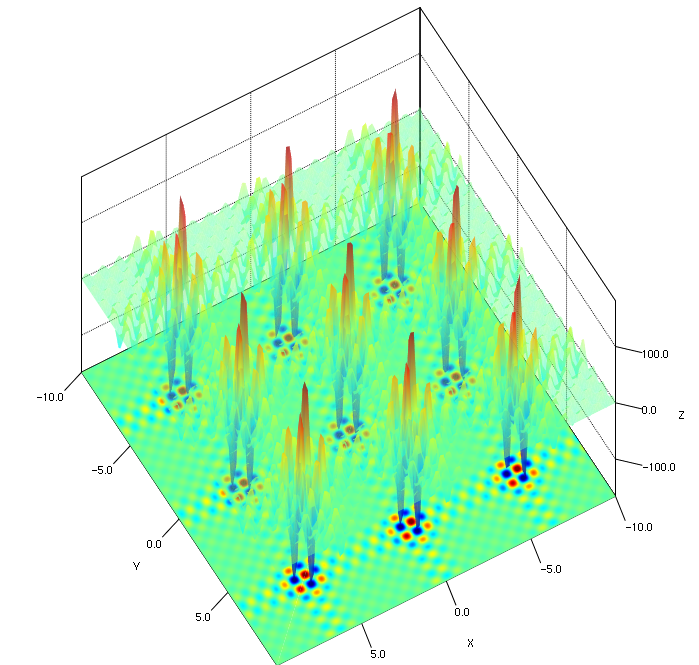
que presenta un máximo de 38.809 en 11.625 y 5.726

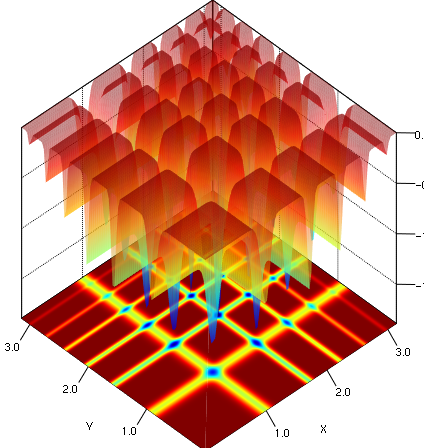
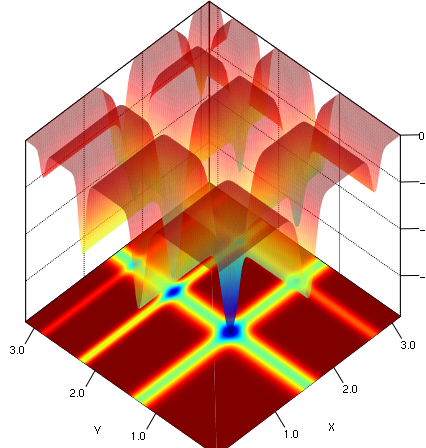
tiene la siguiente gráfica en la pestaña Problem View

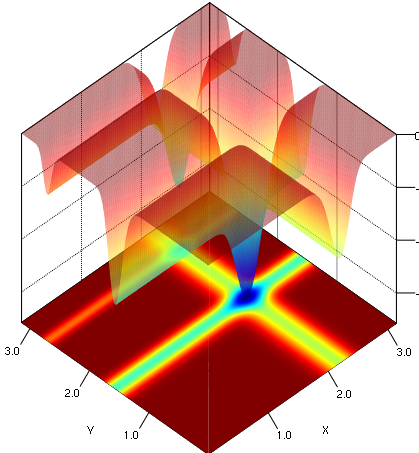
HOLDER TABLE:



SCHUBERT:



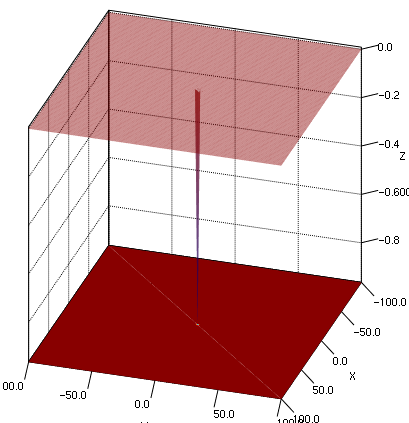
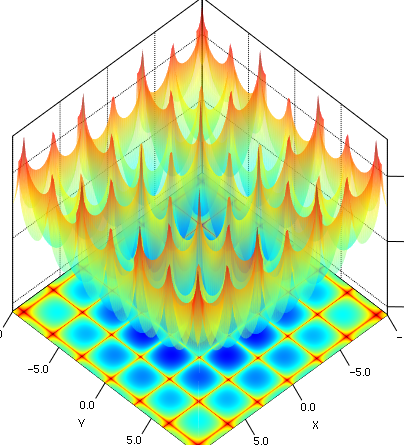
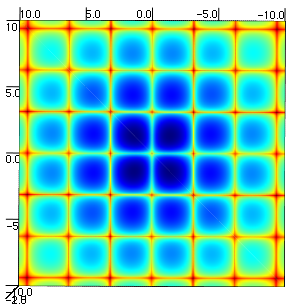
Michalewicz:



Esta función se ha dibujado con las últimas dos variables de con el resto puesto a 0, para poder mostrarla en 3d, en el cálculo de la función de evaluación se consideran todas las variables.

### DE JONG n5: <https://www.sfu.ca/~ssurjano/dejong5.html>

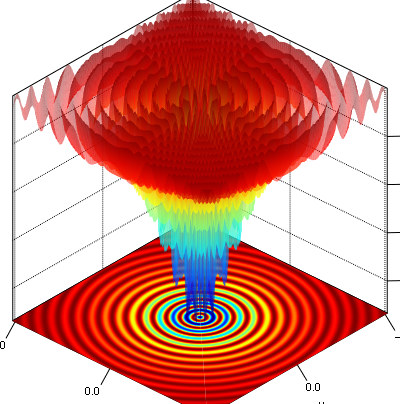
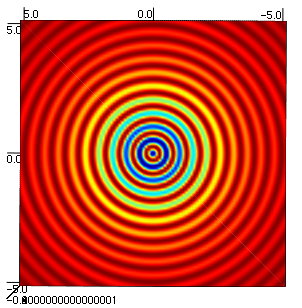
### EASOM: <https://www.sfu.ca/~ssurjano/easom.html>

Mínimo en   
CROSS-IN-TRAY: <https://www.sfu.ca/~ssurjano/crossit.html>

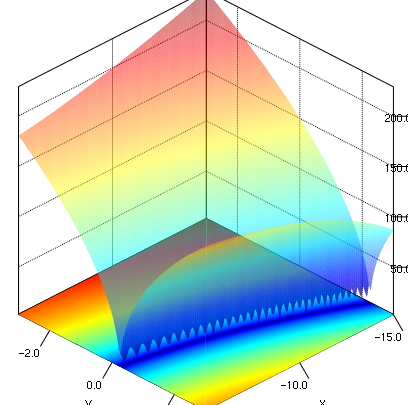
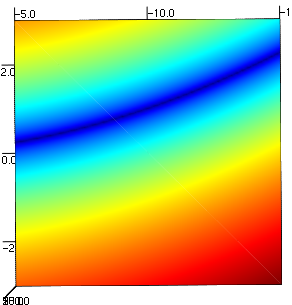
mínimo

### DROP-WAVE: <https://www.sfu.ca/~ssurjano/drop.html>

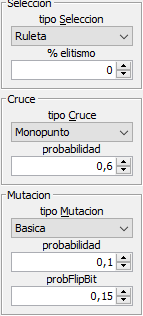
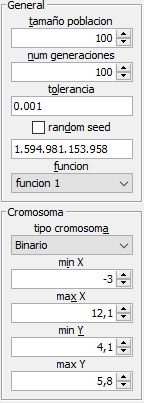
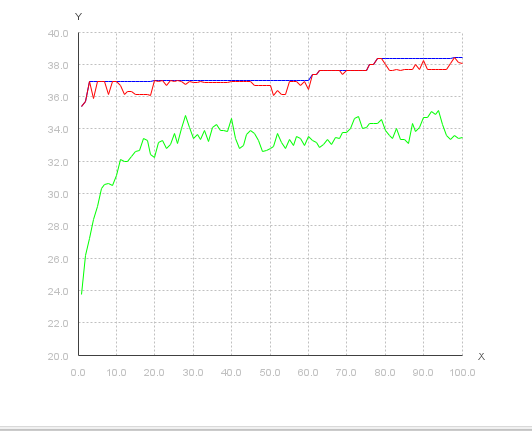
mínimo en



### BUKIN n6: <https://www.sfu.ca/~ssurjano/bukin6.html>

Mínimo en

## Ejecución del algoritmo evolutivo

A continuación, se muestran ejecuciones del algoritmo con su correspondiente semilla para volver a replicar la ejecución:

Seed: 1.594.981.153.958, mejor global:x:11.619, y:5.32, value2optimize:38.4141

Seed: 1.594.980.669.528, mejor global:x:11.618, y:5.72, value2optimize:38.7509

Seed: 1.594.980.797.045, mejor global:x:11.617, y:5.72, value2optimize:38.7970

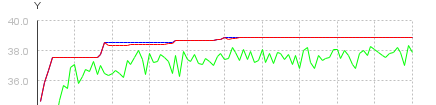
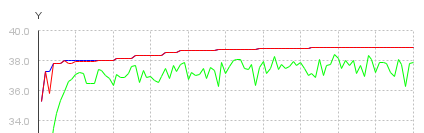
Con elitismo en el primero llegamos una solución mejor:

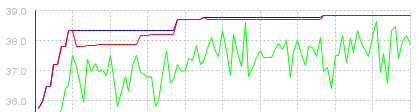
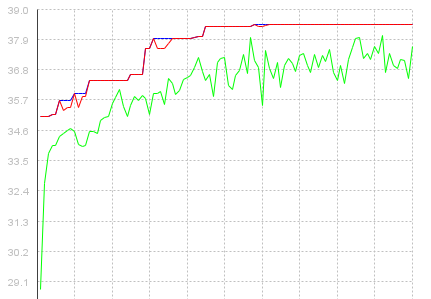
Seed: 1.594.981.153.958, mejor global:x:11.625, y:5.72, value2optimize:38.8497

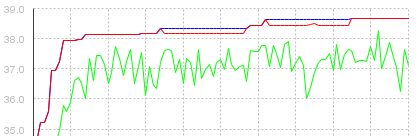
Seed: 1.594.980.669.528, mejor global:x:11.629, y:5.72, value2optimize:38.8349

Seed: 1.594.980.797.045, mejor global:x:11.625, y:5.72, value2optimize:38.8497

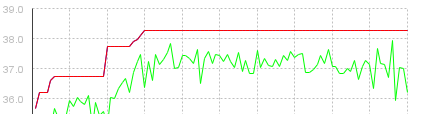
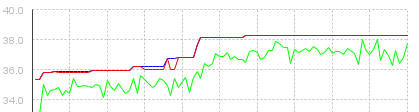
Vemos que se pasa, esto puede ser debido a la precisión, aka tolerancia.

 podemos observar que con selección por torneo los mejores de cada iteración se parecen mucho a los globales, ya que las líneas rojas y azules casi se solapan, esto es debido a que la selección por tornero es muy elitista



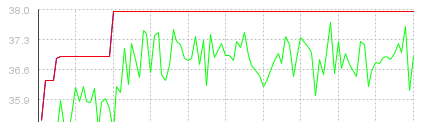
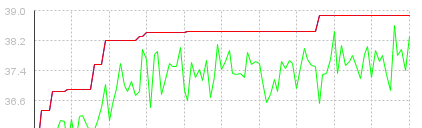


Con *selección estocástica universal*, podemos ver el mismo solapamiento, con esta selección podemos cambiar el número de individuos(*kindividuos*) que se eligen a forma de rastrillo, igualmente espaciados.



K=3

K=2

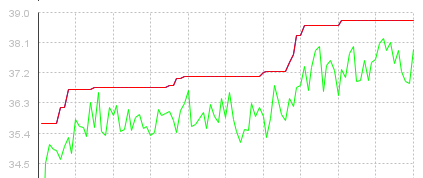


K=4

K=6

K=4

mejor global:x:11.62 y:5.72 value2optimize:38.845

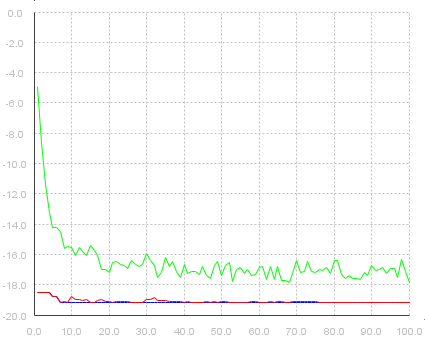
Con selección por Truncamiento:

Seed: 1.594.987.085.066, mejor global:x:11.62, y:5.72, value2optimize:38.8493

Seed: 1.594.987.162.987, mejor global:x:11.62, y:5.62, value2optimize:38.7494

trunc=0.5

Holder Table: minimización

Seed: 1.594.988.059.383,

mejor global:

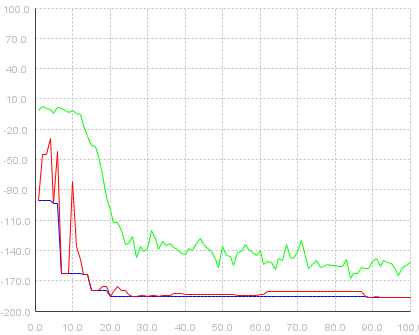
x:-8.0535294656209 y:9.66063417462691 value2optimize:-19.208321534080596

Seed: 1.594.988.120.854,

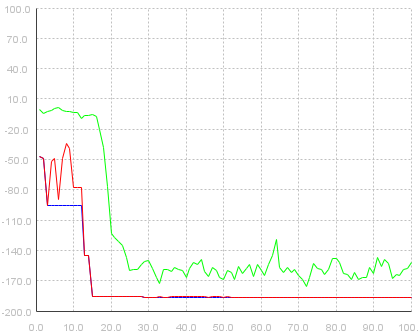
mejor global:

x:8.0626850184637 y:-9.66917935728019 value2optimize:-19.207702358745692

vemos que son dos puntos distintos de la mesa (holder *table*)

Schubert: minimización

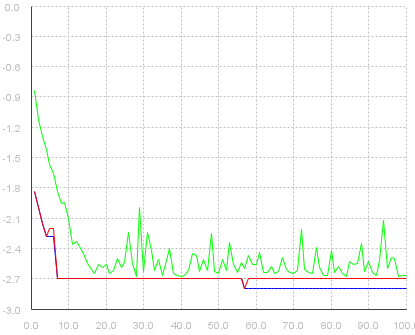
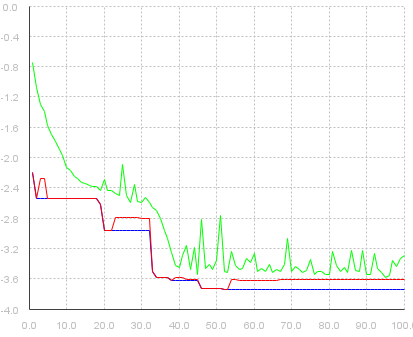
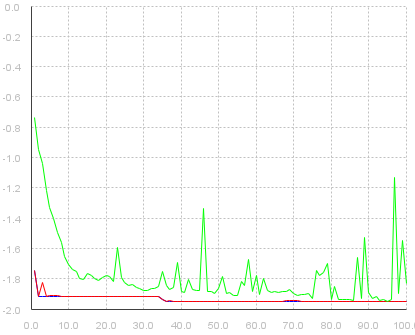
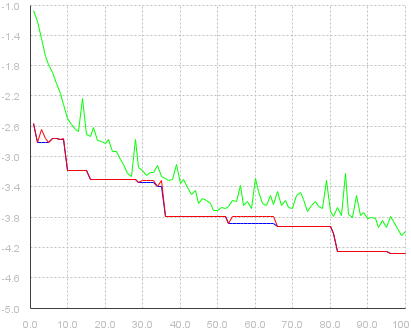
Seed:1.594.988.468.243, mejor global: x:-1.4255195776238292 y:-0.8053834650715661

value2optimize:-186.6738823445401

seed: 1.594.988.652.294, mejor global: x:-7.095248268074587 y:-7.70683919797357

value2optimize:-186.42076259942286

Al igual que holder table nos da múltiples puntos porque como vimos en la visualización hay múltiples mínimos.

Michalewicz: minimización

variables:4

variables:3

variables:2



variables:5

variables:6

Seed: 1.594.997.363.508:

mejor global:[1.56, 1.29] value2optimize:-1.9505423586745683

mejor global:[1.54, 1.29, 1.09] value2optimize:-2.7933649366837274

mejor global:[1.57, 1.26, 1.08, 1.72] value2optimize:-3.7364475154505286

mejor global:[1.57, 1.28, 1.10, 0.97, 0.94] value2optimize:-4.286485872082538

mejor global:[1.59, 1.31, 2.50, 0.99, 2.03, 1.44] value2optimize:-5.109238247151112

mejor global:[1.56, 1.29, 1.09, 0.98, 2.01, 1.44, 1.35] value2optimize:-6.4177004317110935