Práctica 7 búsqueda local

Denisse Leyva

Abril 14, 2021

1. Introducción

En la séptima práctica implementamos una optimización heurística sencilla para encontrar máximos locales de funciones, los ejemplos siendo de Womersley[4] los matemáticamente inclinados pueden consultar su trabajo por métodos de búsqueda más sofisticados, guiados por métodos matemáticos de mayor rigor.

Buscamos minimizar la función unidimensional, a partir de un punto seleccionado al azar, realizando movimientos locales. Estando en x se seleccionará al azar un $\Delta x > 0$ pequeño, calculará los valores $f(x \pm \Delta x)$ y seleccionará el menor de los dos como el siguiente valor de x. Esto se repite k veces y aquel x que produjo el menor valor de $f(x)$ se regresa como el resultado. Se realizarán n réplicas y el menor de ellos es el resultado de la búsqueda en sí [2].

2. Objetivo

La tarea se trata de maximizar algún variante de la función bidimensional ejemplo $g(x, y)$ con restricciones $-3 \leq x, y \leq 3$, con la misma técnica del ejemplo unidimensional. La posición actual es un par x, y y se ocupan dos movimientos aleatorios, Δx y Δy, cuyas combinaciones posibles proveen ocho posiciones vecino, de las cuales aquella que logre el mayor valor para g es seleccionado [2]. Dibujado en tres dimensiones, $g(x, y)$ se ve así

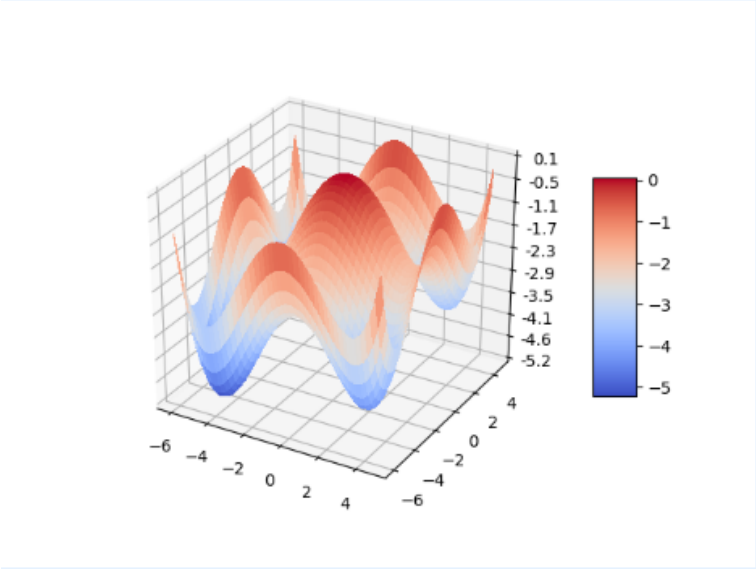


Figura 1: Gráfica 3D de ejemplo.

3. Código

Para este código se tomó en cuenta las 8 posiciones en combinación a los puntos cardinales de la posición inicial, también se tuvo que poner un limitante para que los agentes no se salgan de los límites. Para escoger el mayor se guardaron todos los resultados de las 8 posiciones en una lista de la cual se agarra la posición del mayor y modificamos la posición con el selector. Se utilizó el código base de Schaeffer [3] para obtener la visualización en una proyección en el plano x, y. El código completo esta en GitHub [1].

```
r = agentes.iloc[i]
xl = r.x - r.dx
xr = r.x + r.dx
yd = r.y - r.dy
yu = r.y + r.dy
if  xl < low+step:
    xl = r.x
if  xr > high-step:
    xr = r.x
if  yd < low+step:
    yd = r.y
if  yu > high-step:
    yu = r.y
g1 = g(xl, yu)
g2 = g(r.x, yu)
g3 = g(xr, yu)
g4 = g(xl, r.y)
g5 = g(xr, r.y)
g6 = g(xl, yd)
g7 = g(r.x, yd)
g8 = g(xr, yd)
lista = [g1,g2,g3,g4,g5,g6,g7,g8]
mayor = lista.index(max(lista))+1
if mayor == 1:
    agentes.at[i, 'x'] = xl
    agentes.at[i, 'y'] = yu
elif mayor ==2:
    agentes.at[i, 'x'] = r.x
    agentes.at[i, 'y'] = yu
elif mayor ==3:
    agentes.at[i, 'x'] = xr
    agentes.at[i, 'y'] = yu
...
```

Código 1: Realiza el movimiento de los agentes.

4. Resultados

En la figura dos se puede apreciar los picos más altos que se representan más adelante como el color amarillo en las figuras tres y cuatro haciendo que los agentes se posicionen en ese lugar, igualmente se muestran tres cajas bigote donde se puede apreciar el resultado del mejor valor de todas las replicas en diferentes pasos y con diferentes cantidades de replicas simultaneas, podemos observar como entre mas pasos mas se acerca el promedio al valor deseado.

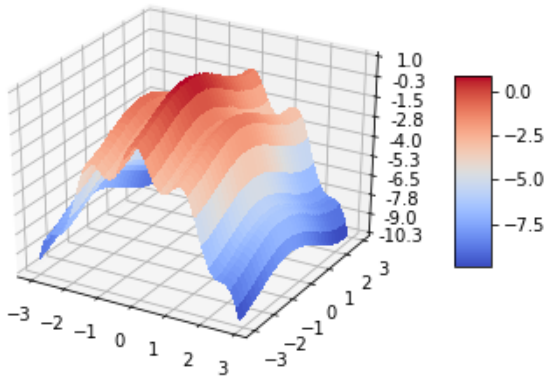
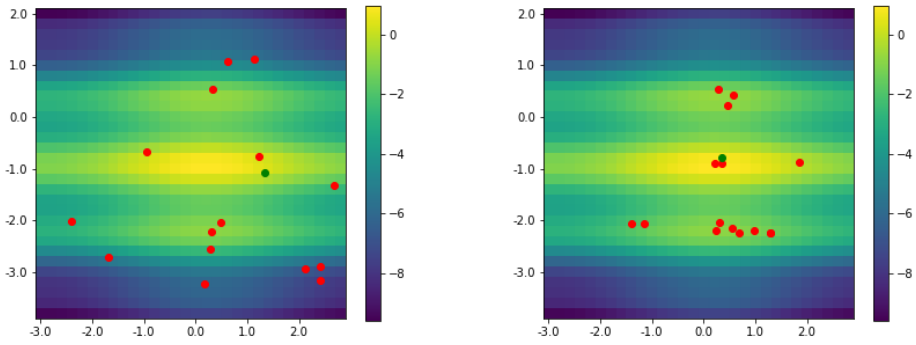


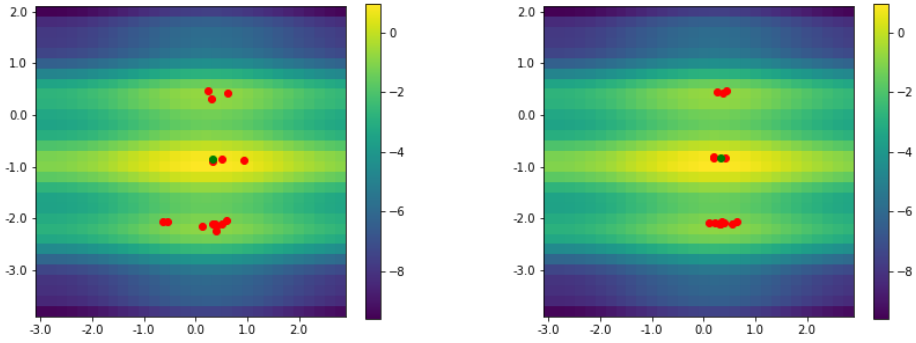
Figura 2: Gráfica 3d.



(a) Gráfica de posición inicial.

(b) Gráfica de posición en 4 pasos.

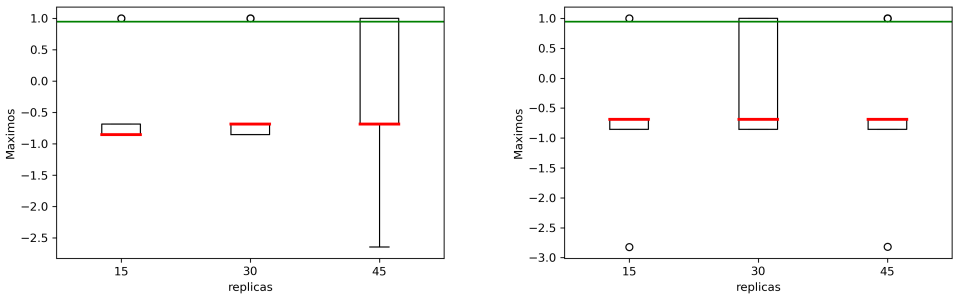
Figura 3: Gráfica que muestra la posición de los agentes



(a) Gráfica de posición en 8 pasos

(b) Gráfica de posición en 20 pasos

Figura 4: Gráfica que muestra la posición de los agentes.



(a) 100 pasos. (b) 1000 pasos.

Figura 5: Gráfica caja bigote comparando replicas.

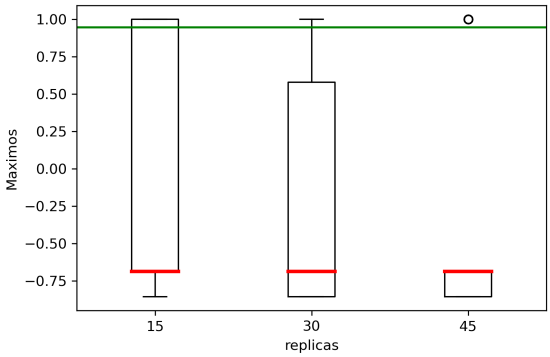


Figura 6: Gráfica caja bigote comparando replicas en 10000 pasos.

5. Reto 1

El primer reto es cambiar la regla de movimiento de una solución x (un vector de dimensión arbitraria) a la siguiente a la de recocido simulado, para optimizar una función f(x), se genera para la solución actual x un solo vecino $x' = x + \Delta x$ (algún desplazamiento local). Se calcula $\delta = f(X') - f(x)$ (para minimizar; maximizando la resta se hace al revés). Si $\delta > 0$, siempre se acepta el vecino x' como la solución actual ya que representa una mejora. Si $\delta < 0$ se acepta a x' con probabilidad $exp(\delta/T)$ y rechaza en otro caso. Aqui T es una temperatura que decrece en aquellos pasos donde se acepta una empeora; la reducción se logra multiplicando el valor actual de T con $\xi < 1$, como por ejemplo 0.995 [2].

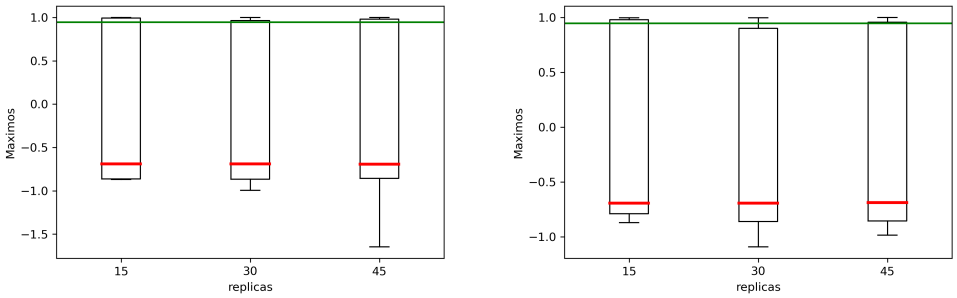
A continuación se muestra una parte del código, donde se realiza el movimiento de los agentes, el código completo se encuentra en GitHub [1], además de los gif que muestran el movimiento de los agentes.

```
r = agentes.iloc[i]
xp = r.x + r.dx
yp = r.y + r.dy

if  xp < low+step:
    xp = r.x
elif xp > high-step:
    xp = r.x
if  yp < low+step:
    yp = r.y
elif yp > high-step:
    yp = r.y

delta = g(xp, yp) - g(r.x, r.y)
prob = exp(delta/r.t)
if delta > 0:
    agentes.at[i, 'x'] = xp
    agentes.at[i, 'y'] = yp
else:
    if random() < (prob):
        agentes.at[i, 'x'] = xp
        agentes.at[i, 'y'] = yp
        agentes.at[i, 't'] = r.t * xi
mejor = g(r.x, r.y)
if mejor > best:
    best = g(r.x, r.y)
    bestx = r.x
    besty = r.y
if mejor > r.best:
    agentes.at[i, 'best'] = mejor
```

Código 2: Realiza el movimiento de los agentes con la regla del recocido simulado.



(a) Temperatura 10 con ξ 0.33

(b) Temperatura 10 con ξ 0.66

Figura 7: Gráfica caja bigote de 100 pasos en función a temperatura y ξ.

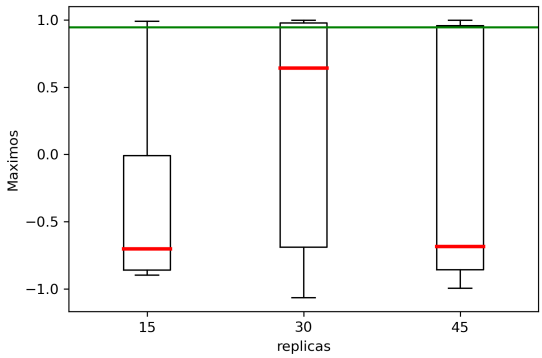
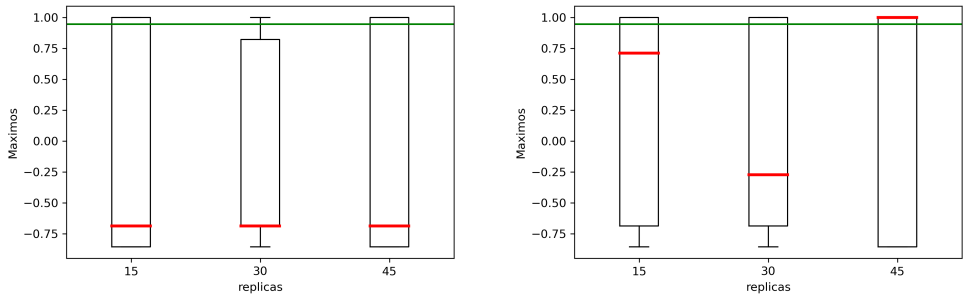


Figura 8: Gráfica caja bigote de 100 pasos en función a temperatura=10 y ξ=0.99.



(a) Temperatura 100 con ξ 0.33.

(b) Temperatura 100 con ξ 0.66.

Figura 9: Gráfica caja bigote de 10000 pasos en función a temperatura y ξ.

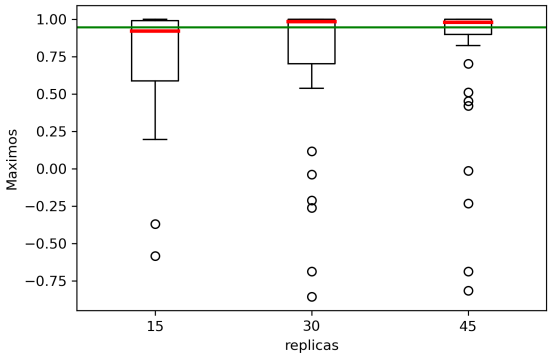
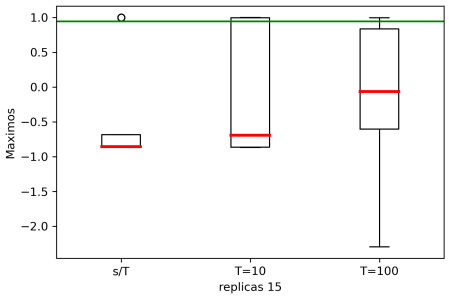


Figura 10: Gráfica caja bigote de 10000 pasos en función a temperatura=100 y ξ=0.99.

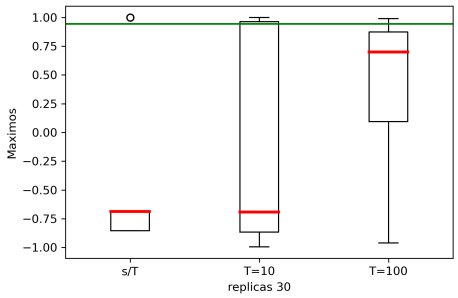
6. Reto 2

El segundo reto es comparar sí o no hay diferencia estadísticamente significativa entre el método de la tarea base y el método del primer reto en términos de la precisión del resultado obtenido (es decir, la diferencia entre el resultado reportado y el óptimo global) en función del número de iteraciones y el número de réplicas [2].

Al observar las siguientes gráficas podemos concluir que es mejor el metodo recocido simulado ya que entre mayor sea la temperatura y mayor sea ξ el resultado del promedio de las muestras esta mucho más cerca en el valor deseado.

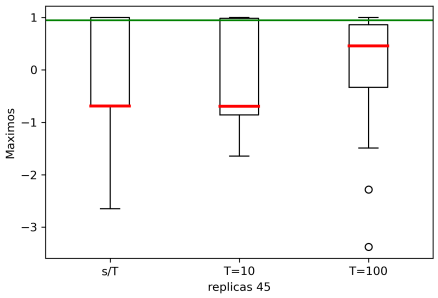


(a) 100 pasos.

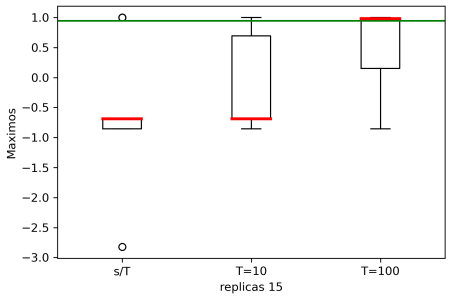


(b) 100 pasos.

Figura 11: Gráfica caja bigote de pasos comparando muestras sin y con temperatura y ξ.

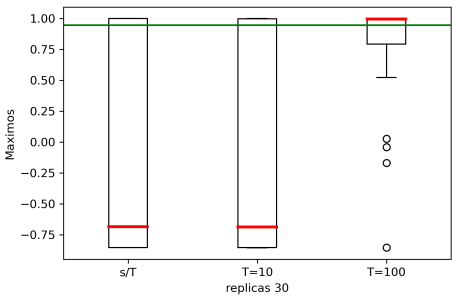


(a) 100 pasos.

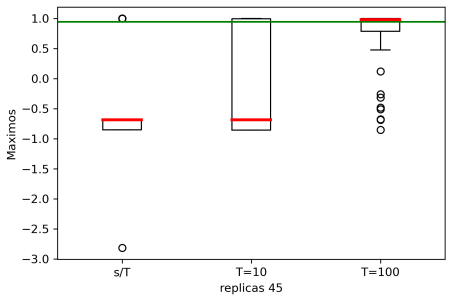


(b) 1000 pasos.

Figura 12: Gráfica caja bigote de pasos comparando muestras sin y con temperatura y ξ.

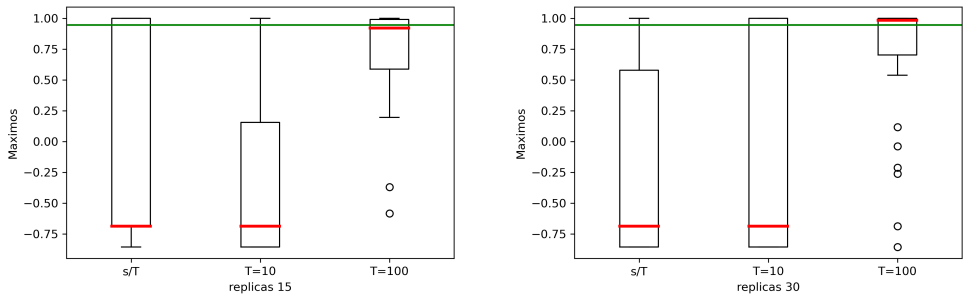


(a) 1000 pasos.



(b) 1000 pasos.

Figura 13: Gráfica caja bigote de pasos comparando muestras sin y con temperatura y ξ.



(a) 10000 pasos. (b) 10000 pasos.

Figura 14: Gráfica caja bigote de pasos comparando muestras sin y con temperatura y ξ.

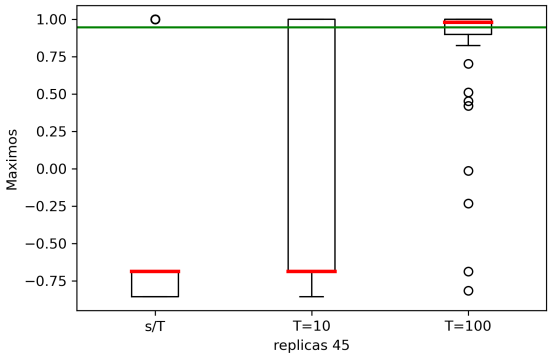


Figura 15: Gráfica caja bigote de 10000 pasos comparando muestras sin y con temperatura y ξ.

Referencias

[1] *Denisse Leyva Repositorio*. URL: `https://github.com/Denisse251/Simulation/tree/main/Tarea.7`.

[2] *Elisa Schaeffer Práctica 7*. URL: `https://elisa.dyndns-web.com/teaching/comp/par/p7.html`.

[3] *Elisa Schaeffer Repositorio*. URL: `https://github.com/satuelisa/Simulation`.

[4] *Rob Womersley: Local and Global Optimization, 2008*. URL: `http://web.maths.unsw.edu.au/~rsw/lgopt.pdf`.