Práctica 7
Búsqueda local

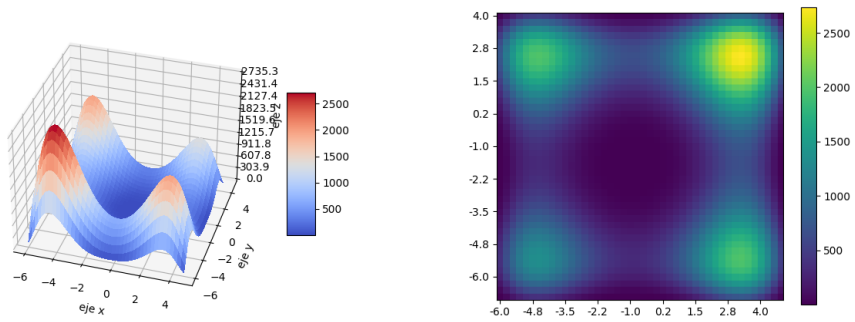
Edson Edgardo Samaniego Pantoja
Materia: Simulación computacional
Fecha 14 de abril de 2021

1. Introducción

En la séptima práctica se implementa una optimización heurística para encontrar máximos locales de la función:

$$f(x,y) = (((x+0{,}5)^4 - 30*x^2 - 20*x + (y+0{,}5)^4 - 30*y^2 - 20*y)^2)/100$$

De la función modificada, se realiza su visualización en tres dimensiones 1a pero para fines prácticos y de mejor identificación de los puntos máximos en la función se utiliza la vista superior en dos dimensiones de la misma como se puede observar en la figura 1b.



(a) (b)

Figura 1: a) Función graficada en tres dimensiones. b) Función de tres dimensiones vista superior.

2. Metodología

Para la práctica son dados ejemplos qué se pueden consultar en la página de Schaeffer [2] donde se puede representar el movimiento de un punto en rojo a la posición más baja en una función vista en 2 dimensiones, a la cual se modifica de manera qué el punto ahora busque el punto más alto como se puede observar en la figura 2.

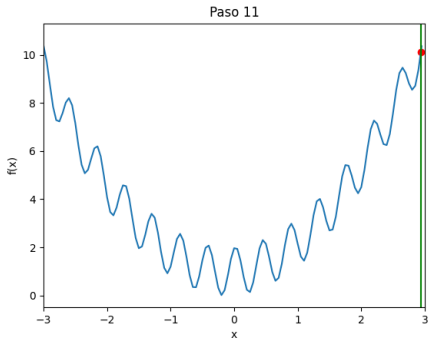
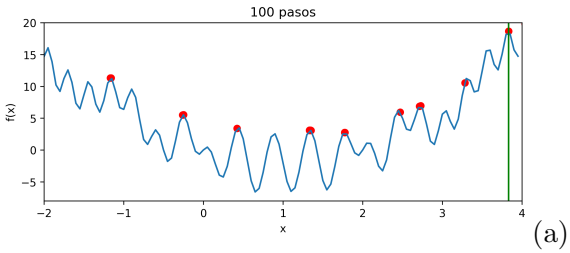
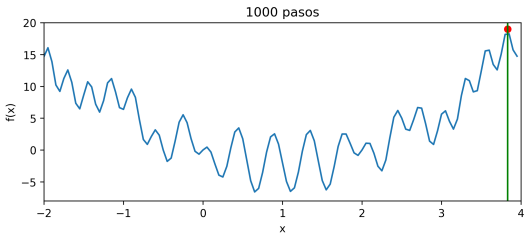


Figura 2: Función dos dimensiones un solo punto.

La siguiente variante dada en la página [2] es generar varios puntos a lo largo de la función que estarán buscando el punto más alto a un número de réplicas dado para lograr visualizar en que replica coinciden todos en el mismo punto, las siguientes figuras 3 a y b muestran el comportamiento.



(a)



(b)

Figura 3: a) Función dos dimensiones y recorrido de puntos a 100 pasos. b) Función dos dimensiones y recorrido de puntos a 1000 pasos.

3. Objetivo

El objetivo principal es crear una visualización animada de cómo proceden quince réplicas simultáneas de la búsqueda con la lógica modificada anteriormente, se realiza este comportamiento en la gráfica en tres dimensiones visualizada desde la parte superior ya que si se utiliza en tres dimensiones los puntos no se apreciarían de la mejor manera.

4. Simulación

La programación es modificación del programa dado por Schaeffer [2] y el programa puede ser consultado en el github de Samaniego [1]. La primera parte modificada en el programa es el ciclo `for` inicial en el que contiene treinta ciclos y dentro de el se declara dos variables `currx y curry` que dan un número al azar entre `low y high` para los dos ejes y dos variables mas llamadas `bestx y besty` almacenarán las variables anteriores respectivamente. Posteriormente entra a un ciclo `for` mas que realiza otras 30 iteraciones por cada ciclo del for global. Esta lógica realiza el camino de un punto, treinta iteraciones y al siguiente ciclo sera el camino de otro punto y así sucesivamente treinta puntos.

```
for ciclo in range(tmax):
```

```
currx = uniform(low, high)
curry = uniform(low, high)
[bestx, besty] = [currx, curry]
for iteracion in range(tmax):
```

Dentro del segundo ciclo lo que se realiza es determinar los movimientos de izquierda, derecha, arriba y abajo respectivamente a cada eje `x` y `y`, restando `deltax` siendo un dato al azar entre cero y paso qué vale 0.20 qué es el máximo avance por punto, este dato se le resta a la variable `currx y curry` qué es la posición donde estará el punto. Para cada una de las cuatro posiciones se limitan con funciones `if` para que no sobrepasen los rangos de la gráfica.

```
deltax = uniform(0, step)
leftx = currx - deltax
leftx = low if leftx < low else leftx
rightx = currx + deltax
rightx = high if rightx > high else rightx

deltay = uniform(0, step)
lefty = curry - deltay
lefty = low if lefty < low else lefty
righty = curry + deltay
righty = high if righty > high else righty
```

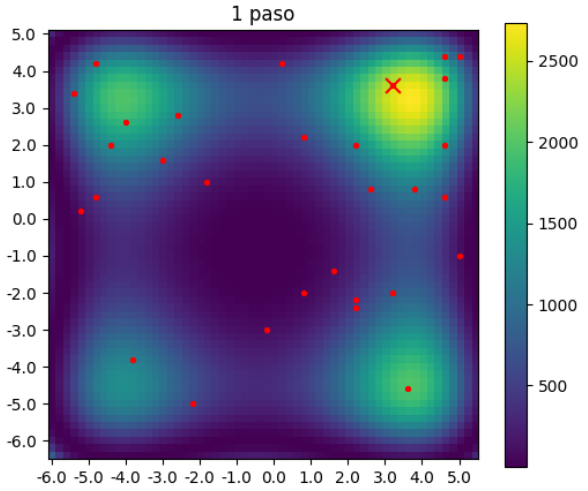
La función `if` que se ve a continuación, decide sí `currx y curry` en la función `g` generada es mayor a `bestx y besty` en la función g entonces estas ultimas variables tomaran el valor de `currx curry`.

```
if g(currx, curry) > g(bestx, besty):
    [bestx, besty] = [currx, curry]
```

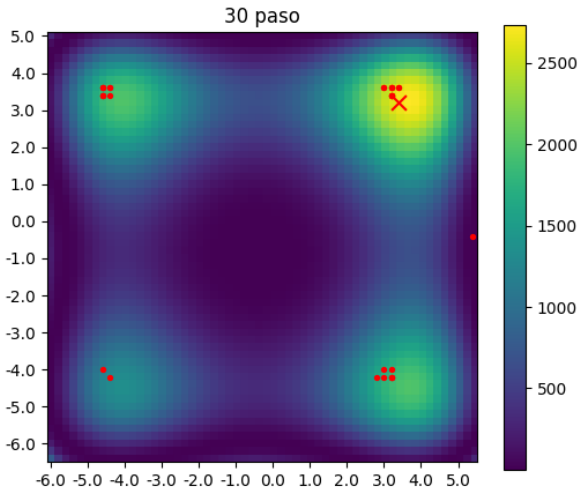
De esta manera se asegura que la posición a la que se moverá el punto en la función será una mejor posición y va a tender a una posición más alta sin que descienda. La última parte de la programación solo consiste en como graficar estos resultados en réplicas de como se desplazan los puntos, se puede consultar en github [1].

5. Resultados

Los resultados de la programación archiva imágenes de las réplicas las cuales fueron tomadas para crear un gif con la página Giphy [3] y puede ser visualizado en github [1]. El inicio del gif se puede observar en la figura 4a, en el qué se visualiza como se distribuyen los puntos a lo largo de la función de manera al azar. Para el último paso o réplica del gif es mostrada en la figura 4b, se observa que los puntos tienden a irse a los picos más altos en la gráfica siendo el más alto del lado superior derecho.



(a)



(b)

Figura 4: a) Primera distribución de puntos. b) Paso final con puntos aglomerados en puntos altos.

6. Conclusión

La tendencia de cada punto a ir al pico alto dependerá mucho de en que posición inicien al azar porque sí el punto cae muy cerca en cualquiera de los cuatro picos su tendencia es ir hacia el pico mas cercano aún qué no sea el más alto de los cuatro picos, de esta manera la mayoría de los puntos están en estos cuatro puntos y teniendo una mayor cantidad de puntos rojos en el pico mayor.

Referencias

[1] Samaniego E. Práctica 7, 2021. URL https://github.com/edson-samaniego/simulation-2021/tree/main/Pr%C3%A1ctica-7.

[2] Schaeffer E. *Práctica 7*, 2021. URL https://elisa.dyndns-web.com/teaching/comp/par/p7.html.

[3] Cooke J. Giphy, 2013. URL http://www.giphy.com/.