Práctica 7

Jose David Martinez Ruiz

26 de septiembre de 2017

1. Introducción

Para esta práctica 7 se implementó un algoritmo de búsqueda local para encontrar el máximo de una función en una región dada. Se procedió a partir de uno implementado para búsqueda local en el plano bidimensional; se modificó para que el algoritmo funcionara para una función en el plano tridimensional.

Para el reto uno, se hizo una representación gráfica de cómo la función se va moviendo a un punto mejor hasta que encuentra su máximo en esa área o bien alcanza su número de pasos totales.

Para el segundo reto consistió en modificar la función que busca el máximo de la función de modo que buscara utilizando el método del recocido simulado. Donde se puede aceptar un vecino que genera un valor peor para la función que el valor que genera la posición actual, con una probabilidad dada a partir de una temperatura inicial y la diferencia que hay entre el valor de la función en el punto actual y el vecino. La temperatura irá bajando cada vez que se escoja un valor peor para la función, dicha temperatura bajará al multiplicarse con un un valor épsilon menor a uno. Se examinará la efectividad de aplicar esta técnica en comparación de la técnica anterior y también que efecto tiene la temperatura y épsilon.

2. Desarrollo

Se busca encontrar el máximo de una función bivariada. A partir del método proporcionado para obtener el mínimo para una función univaluada. La función que queremos trabajará de la siguiente manera, primero se genera un numero aleatorio dentro del rango donde se buscara, después de se generan vecinos a partir del primer punto donde se verificaran si aumentan el valor de la función que queremos maximizar, después se verificará cuál de los vecinos genera el valor de la función más grande y se compara el valor de la función en ese vecino y se compara con el valor que tiene la función en el primer punto que se generó, si resulta que el vecino hace más grande a la función, la posición cambia a ese punto, en caso contrario el punto se queda en donde se encuentra.

Para poder generar la función que necesitamos, primero se modificó la función que se nos proporcionó en un principio, dicha función busca en un área el punto que hace más pequeña el valor de una función. Para nuestros propósitos, se adaptó la función ya proporcionada para que funcionara con una función bivariada y buscara el máximo, para esto primero se cambió el criterio para que es escogieran valores que hicieran a la función más grande, después de eso modifico para que se generaran cuatro vecinos, uno hacia nada punto cardinal con respecto al punto donde se encuentra actualmente. Se compararon esos cuatro puntos por pares, de modo que se comparan los puntos que se encuentran "arriba" y "abajo" del punto donde nos encontramos y también los puntos que se encuentra en el lado "derecho" e "izquierdo". Para ambos casos se obtiene el punto que haga más grande a la función, luego esos dos puntos se compararán para obtener el más grande. Ya con ese punto se comparará con el punto en donde ya nos encontramos, si hace más grande el valor de la función, nos moveremos al punto nuevo, en caso contrario nos quedaremos en el punto donde estamos. La función recibe como argumento la cantidad de veces que el proceso de búsqueda se va a realizar y retorna el par de puntos que general el valor de la función más grande que pudo encontrar. Se espera que mientras más sean los procesos de búsqueda, la función arroje un valor más exacto del máximo real que se encuentra en esa área.

De modo que los puntos no se salgan del área que nos interesa se delimitó el área y se le agregaron unas condiciones extra de modo que los puntos se comportaran como si se estuvieran moviendo en un toroide. Debido a que la función crece mucho y muy rápido después de ciertos valores, se fijó el total de búsqueda al cuadrado que es de lado ocho y que va de cuatro negativo a cuatro positivo en ambos ejes.

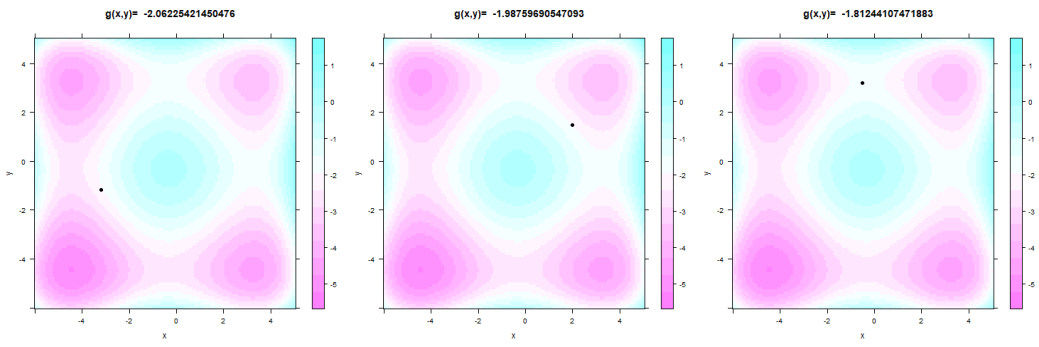
Se le realizaron cien replicas para tres diferentes tamaños de procesos de búsqueda que hará la función. El resultado más grande obtenido de todos los máximos que la función pudo encontrar se encontró en la muestra de tamaño diez mil y con valor de 0.0666821.

3. Reto 1

Para el primer reto se debe de hacer una visualización del recorrido que se hace desde el punto aleatorio donde inicia hasta el punto más grande que pueda encontrar. Para esto se modificó la función que se utilizó anteriormente para que fuera guardando todos los puntos por donde pasa, es decir todos los puntos, desde donde empezó, seguido por su siguiente movimiento hacia un punto mejor y así sucesivamente hasta que llegara al valor más grande que pudo encontrar en la cantidad de procesos que se le pidió buscar.

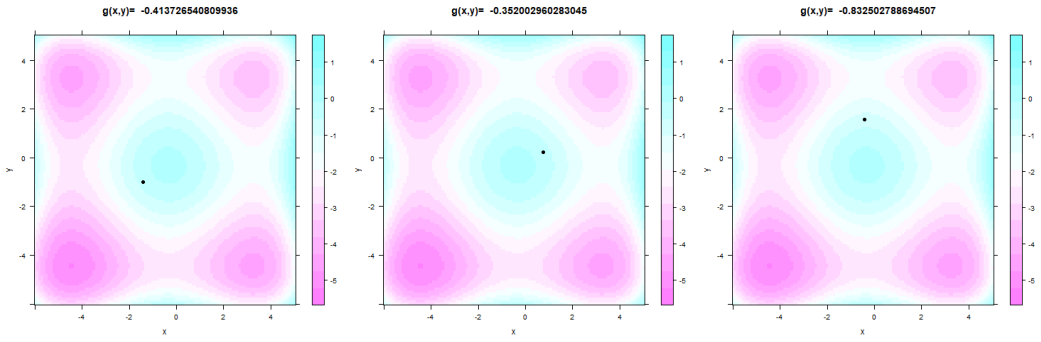
La función está estructurada de manera que siempre escoja un valor que mejore la función en cada iteración, si no logra encontrar un valor mejor, se quedará en el mismo lugar hasta que encuentre un vecino que mejore el valor de la función.

Al igual que en caso anterior, se utilizaron tres diferentes tamaños de procesos de búsqueda. Se hizo solo una réplica para cada caso, de modo que se pueda visualizar y comparar de una manera más sencilla. Para esto último se colocó el valor de la función que buscamos maximizar en cada gráfica del punto moviéndose en el área de búsqueda. Se anexará un gif del recorrido en busca del máximo en el área para cada tamaño de proceso de búsqueda al repositorio. A continuación, se mostrara parte del gif de cada uno.



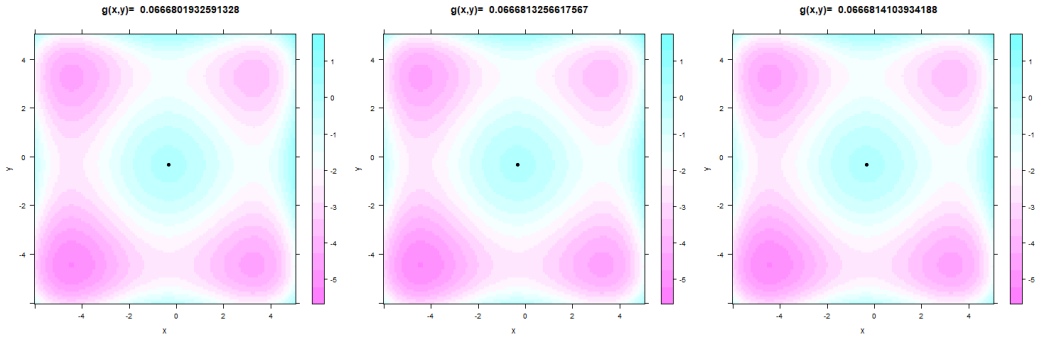
(a) tamaño de 100 (b) tamaño de 1000 (c) tamaño de 10000

Figura 1: Primera posición



(a) tamaño de 100 (b) tamaño de 1000 (c) tamaño de 10000

Figura 2: Posición intermedia



(a) tamaño de 100 (b) tamaño de 1000 (c) tamaño de 10000

Figura 3: Posición final

4. Reto 2

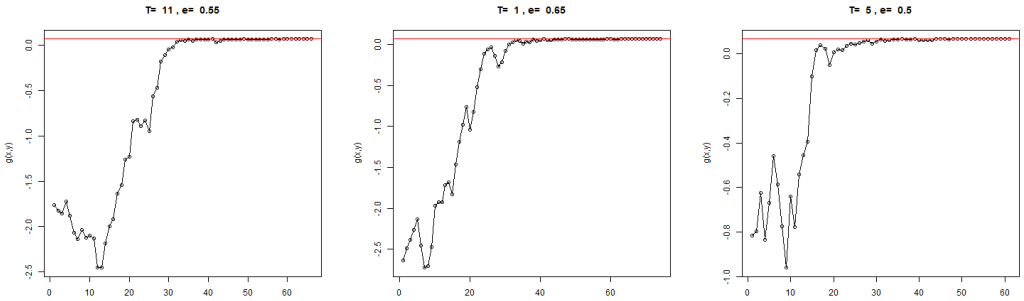
Para el segundo reto se cambió la función de búsqueda del máximo de manera que se implementara el método de búsqueda del recocido simulado. Para esto solo se generó un vecino y se comparará el valor de la función en ese punto vecino con el valor de la función en el punto donde nos encontramos. Si el punto vecino resulta ser mejor que el primero, nos movemos a esa posición. En caso contrario se calculará la diferencia que hay entre la función evaluada en el punto vecino menos el punto evaluada en el punto actual, dicha diferencia la llamaremos delta. Se calculará la función exponencial evaluada con el cociente del valor de delta y una temperatura dada, este resultado se tomará como una probabilidad. Se generará un numero aleatorio entre creo y uno, si este valor es menor al de la exponencial calculada anteriormente, nos moveremos al valor del vecino que nos da un valor de la función peor al que tenemos en el punto actual. Además de lo anterior, si se acepte el vecino "malo" como una solución mejor, la temperatura dada bajará, para esto, se multiplicará por un valor épsilon menor que uno, de manera que el resultado sea una temperatura más baja, al ocurrir esto el proceso de búsqueda se vuelve más estricto, es decir, si se escoge un valor peor, en la siguiente búsqueda será más difícil escoger un valor peor. Los parámetros que la función recibe son la cantidad de procesos de búsqueda que hará, el valor inicial de la temperatura y el valor de épsilon; la función retornará los puntos del recorrido del recocido simulado.

Lo interesante de este método es que hay una posibilidad de aceptar valores que no mejoren el valor de la función que nos interesa, esto nos abre la posibilidad de explorar otras áreas que podrían llevarnos a soluciones aún mejores para función objetivo.

Se mantuvo fijo el número de procesos de búsqueda en diez mil, ya que este valor nos arrojó el mejor valor de la función y más cercano al valor real. Debido a que nos interesa obtener mejores resultados, los valores tanto de la temperatura como de épsilon no fueron variados con números

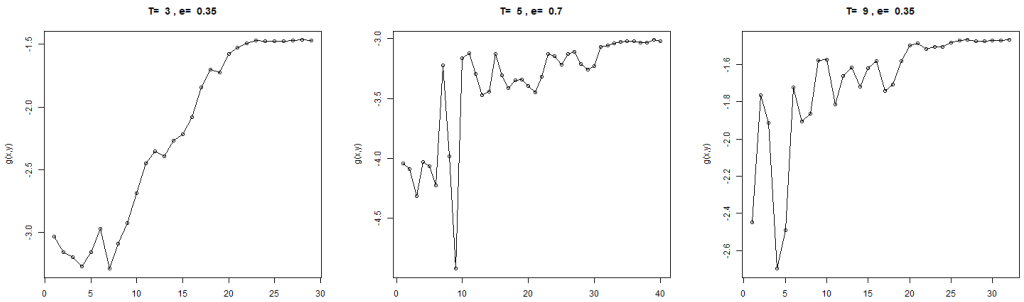
grandes, si no con números pequeños para que el recocido simulado fuera relativamente más estricto desde el principio que si utilizara valores más grandes. La temperatura varió desde uno hasta once con pasos de valor dos, y épsilon varió desde 0.30 hasta 0.70 con pasos de 0.05.

Los valores obtenidos, variando lo anterior mencionado, se evaluaron los pares de puntos en la función en cuestión y, para cada caso, se graficó el comportamiento que se obtuvo, es decir, se graficó como va moviéndose, por así decirlo, el valor de la función conforme los pasos de búsqueda van aumentando. se marcará con una línea roja el valor máximo real de la función en esa área. Se mostrarán algunos de los gráficos obtenidos, el resto se anexará en el repositorio.



(a) T= 11, e= 0.55 (b) T= 1, e= 0.65 (c) T= 5, e= 0.5

Figura 4: Casos Favorables



(a) T= 3, e= 0.35 (b) T= 5, e= 0.7 (c) T= 9, e= 0.35

Figura 5: Casos no tan Favorables

5. Conclusiones

De la tarea base se concluye que, aunque se obtuvieron valores algo variados, en cada tamaño de procesos de búsqueda, algunos de los valores de las réplicas se acercaban al valor máximo original, sin embargo, con el tamaño de proceso más grande se pudo obtener el valor más grande y más preciso del máximo local de la función. Debido a lo anterior se consiguió satisfactoriamente aplicar el método de búsqueda local para esta función bivariada.

Del primer reto se concluye que, la función implementada para la búsqueda local, siempre buscará obtener un mejor valor en cada iteración, de lo contrario no se moverá. El camino hecho en los tres casos es bastante similar, todos buscaron el valor mayor, que se encontraba en el centro de la gráfica, la diferencia radica en que unos hicieron más pasos que otros y ahí es donde se puede notar que mientras más grande sea el tamaño del proceso de búsqueda, más cerca llega al valor máximo real de la función que es de 0.0666822.

Del segundo reto, la implementación del recocido simulado nos ayudó a obtener en su mayoría valores muy cercanos al máximo real en cada uno de los casos, sin embargo, en otros no se estuvo nada cerca, esto puede deberse a que al aceptar un valor "malo" para la función, esto no le permitió buscar un vecino que lo ayudara a acercarse al valor máximo real, y se quedó estancado en el valor más grande que pudo encontrar. Sin embargo, en su mayoría se obtuvieron resultados bastante adecuados debido a que, llegando a cierto punto de la exploración, buscaron mejorar y acercarse lo más posible al valor real de la función. En conclusión, el recocido simulado es una buena opción de búsqueda ya que proporciona resultados bastante precisos en la mayoría de los casos.