**Universidad Autónoma Metropolitana (UAM)**

**Unidad Iztapalapa**



**División de ciencias básicas e ingeniera (CBI)**

Inteligencia Artificial

**Practica 3 “ Hill Climbing, Steepest Hill Climbing y Steepest Hill Climbing Replacement"**

David Moreno Gutiérrez.

Corrales Andrade Fidel Francisco.

18-P

**INTRODUCCIÓN**

El algoritmo hill climbing, también llamado Algoritmo de Escalada Simple o Ascenso de colinas es una técnica de optimización matemática que pertenece a la familia de los algoritmos de búsqueda local. Es un algoritmo iterativo que comienza con una solución arbitraria a un problema, luego intenta encontrar una mejor solución variando incrementalmente un único elemento de la solución. Si el cambio produce una mejor solución, otro cambio incremental se le realiza a la nueva solución, repitiendo este proceso hasta que no se puedan encontrar mejoras. Suele llamarse a esta búsqueda algoritmo voraz local, porque toma un estado vecino "bueno" sin pensar en la próxima acción.

En está practica se analizaran tres versiones del algoritmo Hill Climbing

* Hill Climbing
* Steepest Hill Climbing
* Steepest Hill Climbing Replacement

**DESARROLLO**

Para realizar el código de éste algoritmo usamos una clase Hill Climbing y su clase VentanaHillClimbing quien hace uso de los servicios (métodos) de la clase Hill climbing.



En la clase Hill Climbing implementamos tres atributos cuya unica funcion es la de guardar el valor de "x" en cada una de las iteraciones para después ser utilizados en la clase VentanaHillClimbing a traves de un "getter" y colocarlo en una tabla. Tambien cuenta con los 3 algoritmos HillClimbing como método, 2 métodos que generan una solucion cuya funcion es la de complementar a los 3 algoritmos HillClimbing, y el método función.

La clase Ventana en cambio solo posee una interfaz gráfica para que el usuario pueda ser capaz de colocar una función de 2 disponibles, y pueda observar los resultados en una tabla y una gráfica, así como limpiar los datos de la tabla y salir del programa.

Los problemas que presentamos al realizar la práctica fueron el hecho de escoger una función de dos posibles, por lo que se implemento la variable "opción" y se pasó como parámetro en todos los métodos, está variable solo podia tener los valores 1 o 2 según el usuario, y establecía cual era la función a trabajar con los 3 algoritmos. El segundo problema fue el hecho de guardar las soluciones después de cada iteración para poder representarlos en la tabla y gráficamente, por lo que se implementaron 3 atributos llamados "iteración 1, 2 y 3" con sus respectivos Getters y Setters. El tercer problema fue el hecho de hacer una método que sume o reste aleatoriamente un número, para ello se utilizó un método de la clase Random de Java que genera un booleano aleatoriamente, después usamos un if-else, si el método era positivo sumaba, si no restaba.

**Resultados (graficas)**

A continuación se presentan las gráficas obtenidas de algunos casos experimentados con las tres versiones del algoritmo Hill Climbing

Resultados en la función f(x) = sen ((x\*x)/10) / ((x\*x) / 10) :



Resultados en la función f(x) = sen (x) + sen (10x/3) :



Al analizar los resultados, observamos como este algoritmo encontraba óptimos locales y en raras ocasiones el optimo global, esto debido a que el algoritmo esta pensado en buscar el mejor resultado después de 3 iteraciones.

**CONCLUSIÓN**

La relativa simplicidad de este algoritmo lo hace una primera elección popular entre los algoritmos de optimización. Es usado ampliamente en inteligencia artificial, para alcanzar un estado final desde un nodo de inicio. La elección del próximo nodo y del nodo de inicio puede ser variada para obtener una lista de algoritmos de la misma familia. Aunque algoritmos más avanzados tales como simulated annealing o búsqueda tabú pueden devolver mejores resultados, en algunas situaciones hill climbing opera sin diferencias. El hill climbing con frecuencia puede producir un mejor resultado que otros algoritmos cuando la cantidad de tiempo disponible para realizar la búsqueda es limitada, por ejemplo en sistemas en tiempo real. El algoritmo puede devolver una solución válida aún si es interrumpido en cualquier momento antes de que finalice.

**Opinion Personal**

El Algoritmo Hill climbing es interesante para encontrar un óptimo local (una solución que no puede ser mejorada considerando una configuración de la vecindad) pero no garantiza encontrar la mejor solución posible (el óptimo global) de todas las posibles soluciones (el espacio de búsqueda). La característica de que sólo el óptimo local puede ser garantizado puede ser remediada utilizando reinicios (búsqueda local repetida), o esquemas más complejos basados en iteraciones como recocido simulado y programación genética.