UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Industrias

Investigación de Operaciones

2^{do} Semestre 2020 | Programa Diurno | Casa Central - Vitacura

---

Tema de la tarea

Tarea Computacional 1

Autores:
Miguel Bustamante
201610501-8, Ingeniería Civil Matemática
Pablo Calcumil
201673563-1, Ingeniería Civil Matemática

Profesor:
Rodrigo Mena
Rafael Favereau
Pablo Escalona

Ayudantes:
Katherine Alvaréz
Ignacio Cabezas
Carla Rojas
Vincenzo Rosati
Eduardo Villanueva

23 de diciembre de 2020

Resumen

El presente trabajo trata de la resolución del *Single Source Weber Problem*, el cual consiste en la minimización de una suma ponderada de las distancias entre un nuevo servidor y los puntos de origen o destino, así seleccionando la mejor ubicación para un servidor en cualquier parte de un plano.

Para la resolución de este problema, se utilizan ciertos métodos de descenso conocidos y algunos algoritmos especializados para la resolución de este problema, los cuales deben ser implementados en el lenguaje de programación Python y el software AMPL. Una vez implementados estos métodos, se desea hacer una comparación entre estos a través de los resultados obtenidos de la función objetivo, de su tiempo de ejecución y de la cantidad de iteraciones que realiza cada método para llegar al resultado final, donde los datos de prueba para el problema están dados de manera aleatoria, algunos de distribución uniforme continua y otros de uniforme discreta.

Tabla de Contenidos

1. Introducción

El *Single Source Weber Problem* es uno de los problemas más famosos en la teoría de la localización, de la cual existe una amplía gama de métodos de resolución que van desde las heurísticas, hasta los métodos numéricos, como los de descenso que se presentarán en este trabajo.

En la resolución de este problema se implementarán algunos métodos de descenso en Python, como lo son el método del gradiente y el método de Newton, los cuales contarán con el método de Backtracking Line Search, además de un método especializado para la resolución de este problema el cual es el algoritmo de Weiszfeld y el solver MINOS dado por AMPL. El objetivo de tener estos métodos, es el de poder comparar tales algoritmos por medio de los tiempos de ejecución, la cantidad de iteraciones para cumplir cierto criterio (como una cantidad máxima de iteración o el error de la solución), y el óptimo de la función objetivo, que tales métodos brindarán para una cantidad de datos de 100, 1000 y 10000, que se obtienen aleatoriamente con distribución uniforme continua y discreta.

2. Formulación del modelo

El *Single Source Weber Problem*, trata de un problema de optimización convexo irrestricto, entonces es posible resolverlo a través de solver's como MINOS o aplicando condiciones de primer orden, es por ello que se consideran los métodos del gradiente, método de Newton, solver MINOS de AMPL y el algoritmo especializado para este problema Weiszfeld.

Donde el modelo se describe de la siguiente manera:

Conjuntos

- I $= \{1, \ldots, n\}$: conjunto de números para dar índice a los parámetros y así agruparlos. Este tendrá las siguientes cardinalidades $|I| = 100$, 1000 y 10000.

Parámetros

- (x_i, y_i) : Ubicación de punto origen o destino i-esimo.
- w_i : Peso asociado al punto (x_i, y_i).

Variables

- $(\bar{x}, \bar{y})$: Ubicación de un servidor.

Función Objetivo

$$\min_{(\bar{x},\bar{y})} \sum_{i=1} w_i \sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2}$$

Sujeto a

$$(\bar{x}, \bar{y}) \in \mathbb{R}^2$$

Se sabe que los parámetros (x_i, y_i) son los puntos de origen o destino y que w_i es el peso de este punto, donde esto se puede interpretar de la forma en que se debe realizar un envío desde $(\bar{x}, \bar{y})$, hacía los puntos (x_i, y_i), donde w_i es el costo por la distancia entre estos puntos, es por esto, que se debe buscar tal punto $(\bar{x}, \bar{y})$, para minimizar el costo de tales envíos.
Dicha esta analogía, se puede ver que el objetivo de este problema es la minimización de la ponderación de las distancias entre los puntos de origen o destino y un nuevo servidor, y sus pesos.

3. Enfoque de Solución

Para el *Single Source Weber Problem*, se han preparado 4 métodos de resolución, de las cuales 2 son métodos de descenso (Gradiente y Newton), un método especializado para este problema (algoritmo Weiszfeld) y uno dado por el solver MINOS de AMPL. Los datos a trabajar, puntos de origen o destino (x_i, y_i) y sus respectivos pesos w_i, en el problema son adquiridos de forma aleatoria con distribución uniforme gracias a las herramientas que tiene Python, como lo es la librería numpy, el cual fue el programa que se utilizó para este trabajo.

Describiendo Algoritmos

Antes de describir los algoritmos, cabe mencionar que los valores w_i, x_i e y_i que son creados aleatoriamente con distribución uniforme, se crean en un archivo llamado `Script.py`, en donde al correr tal programa este desarrolla todas las funciones, entregando los valores de las funciones objetivos para cada cantidad de datos, pidiendo los valores de α, β, iteración máxima k_{max} y la tolerancia de error. Por otro lado, dado que estos algoritmos tienen parámetros parecidos, a diferencia de algunos pocos, se definirán ahora, para no definirlos en cada algoritmo:

\# *ArregloW es un arreglo con los valores w_i creados aleatoriamente,*
\# *ArregloX es un arreglo con los valores x_i creados aleatoriamente,*
\# *ArregloY es un arreglo con los valores y_i creados aleatoriamente,*
\# *numero es un entero, es la cantidad de datos a trabajar en la función (largo de los arreglos),*
\# *tolerancia es el valor mínimo permitido entre $|f(x_{n+1}) - f(x_n)|$,*
\# *kmax es el valor de iteración máxima permitida para el algoritmo,*
\# *alfa es el valor que pertenece a $(0, 0{,}5)$, que está descrito en Backtracking,*
\# *beta es el valor que pertenece a $(0, 1)$, que está descrito en Backtracking.*

Algoritmo Weiszfeld: Este es un algoritmo especializado para el *Single Source Weber Problem*. El algoritmo descrito en pseudocódigo queda de la siguiente manera:

`Procedimiento Función Weiszfeld(ArregloW,ArregloX,ArregloY,numero,tolerancia,kmax)`
(x_k, y_k) `← valor semilla en x e y`
d_i `← distancias de todos los datos al punto` $(x_k,\ y_k)$
Z_k `← valor función objetivo con valores` $(x_k,\ y_k)$
`error ← 1000` *error absurdo para iterar*, `contador ← 0` *contador de iteraciones*
`Repetir`
...... (x_k, y_k) `← nuevo valor dado el algoritmo` $\left(\frac{\sum_i \frac{w_i \cdot x_i}{d_i}}{\sum_i \frac{w_i}{d_i}}, \frac{\sum_i \frac{w_i \cdot y_i}{d_i}}{\sum_i \frac{w_i}{d_i}}\right)$
...... d_i `← distancias de todos los datos para el nuevo punto` $(x_k,\ y_k)$
...... Z_k `← nuevo valor de función objetivo con los nuevos valores` $(x_k,\ y_k)$
...... `error ←` $|Zk1 - Zk|$, `contador ← contador +1`
`Hasta que contador >= kmax OR error < tolerancia`
`VOptimo ← valor función objetivo con valores` $(x_k,\ y_k)$
`Fin procedimiento`

Método del gradiente con backtracking: Este es un algoritmo de método de descenso. El algoritmo descrito en pseudocódigo queda de la siguiente manera:

Procedimiento Gradiente BCT(ArregloW,ArregloX,ArregloY,alfa,beta,num,toleran,kmax)
(x_k, y_k) ← valor semilla en x e y
f_{xk} ← valor función objetivo con valores (x_k, y_k)
df_{xy} ← valor del gradiente con valores (x_k, y_k), con -1, dado que minimizamos
f_{xk1} ← valor función objetivo con valores (x_k, y_k) + df_{xy}
error ← $|f_{xk} - f_{xk1}|$, contador ← 0 *contador de iteraciones*
Repetir
...... t ← 1
...... $funcion_x$ ← suma de funcion incluyendo alfa gradiente y t, para ver condición
......Repetir
.... t ← $t\cdot$ beta
.... f_{xk1} ← valor función objetivo con valores (x_k, y_k) + $t \cdot df_{xy}$
.... $funcion_x$ ← suma de funcion incluyendo alfa gradiente y t, *Actualiza t*
...... Hasta que $f_{xk1} < funcion_x$
...... t ← $fractbeta$ adecuamos
...... (x_k, y_k) ← (x_k, y_k) + $t \cdot df_{xd}$ actualizamos con t y df_{xd}
...... df_{xy} ← actualizamos gradiente con valores (x_k, y_k), con -1 ya que minimizamos
...... f_{xk1} ← valor función objetivo con valores (x_k, y_k) actualizados
......error ← $|f_{xk} - f_{xk1}|$, contador ← contador $+1$
...... f_{xk} ← f_{xk1} valor actual de funcion objetivo optimo
Hasta que error $<$ tolerancia
Fin procedimiento

Método de Newton con Backtracking: Dada la expresión explícita de la función, se puede determinar su matriz Hessiana calculando sus componentes usando derivadas parciales

$$f(\overline{x}, \overline{y}) = \sum_{i \in I} w_i \sqrt{(x_i - \overline{x})^2 + (y_i - \overline{y})^2} = \sum_{i \in I} w_i d_i$$

De aquí se determinan fácilmente la gradiente y la Hessiana de la función objetivo, y para llegar a las siguientes expresiones se utilizan las reglas básicas de la diferenciación (teniendo en cuenta que d_i depende de $\overline{x}$, $\overline{y}$).

$$\nabla f(\overline{x}, \overline{y}) = \left(\sum_{i \in I} w_i \frac{\overline{x} - x_i}{d_i}, \sum_{i \in I} w_i \frac{\overline{y} - y_i}{d_i} \right)$$

$$\nabla^2 f(\overline{x}, \overline{y}) = \begin{pmatrix} \sum_{i \in I} w_i \frac{d_i - (\overline{x} - x_i)^2 d_i^{-1}}{d_i^2} & -\sum_{i \in I} w_i \frac{(\overline{x} - x_i)(\overline{y} - y_i)}{d_i^3} \\ -\sum_{i \in I} w_i \frac{(\overline{x} - x_i)(\overline{y} - y_i)}{d_i^3} & \sum_{i \in I} w_i \frac{d_i - (\overline{y} - y_i)^2 d_i^{-1}}{d_i^2} \end{pmatrix}$$

Teniendo esta información, la implementación del algoritmo posee básicamente la misma estructura que el BLS, con la diferencia que para el cálculo de la dirección de descenso se emplea la Hessiana y el criterio de parada involucra un $\lambda \in \mathbb{R}$ calculado a partir de la gradiente y la Hessiana.

Solver MINOS usando AMPL:

4. Conclusión

En las simulaciones se utilizaron los parámetros $\alpha = 0{,}5$, $\beta = 0{,}8$ y un techo de iteraciones del orden de las centenas. A partir de los datos extraídos de tiempos, óptimos e iteraciones en promedios y máximos, expuestos en las tablas abajo se llega a las siguientes conclusiones:

1. Los algoritmos llegan a los óptimos de función con igual cercanía, en promedio. En $I = 10000$, los óptimos de la función objetivo son prácticamente idénticos.

2. A pesar de la aseveración anterior, en casos particulares el algoritmo de Newton con Backtracking se aleja bastante de BLS y Weiszfeld, cuyos máximos mantienen cercanía con su promedio. Cuando aumenta el número de obeservaciones, los máximos parecen acercarse.

3. Respecto de los tiempos, Weiszfeld y BLS mantienen ventaja siendo el segundo más rápido, por otro lado, NB puede llegar a demorarse hasta 10 veces más que Weiszfeld en casos pequeños y casos grandes, su tiempo está en el orden de las docenas de minutos. Esto contrasta con el hecho de que WF y BLS en ningún caso llegan a sobrepasar el minuto.

4. Teniendo en cuenta las iteraciones, Weisfeld logra cumplir su condición de calificación en menos iteraciones que BLS. Nuevamente, NB es el menos eficiente entre los tres debido a que en muchas ocasiones alcanza el techo de iteraciones impuesto por el grupo para la simulación.

Promedios

$\|I\| = 100$	FO óptimo	CPU time	Iteraciones
Weiszfeld	114624.9451381926	0.0007459473609924316	8.98
Gradiente	114624.95404140829	0.0017334365844726563	22.19
Newton	117302.42234208819	0.13790767669677734	285.16
Minos			

$\|I\| = 1000$	FO óptimo	CPU time	Iteraciones
Weiszfeld	114626196.70756154	0.0007304668426513672	11.42
Gradiente	114626196.76514491	0.005035958290100098	36.85
Newton	114766934.8790328	1.153316206932068	300.0
Minos			

$\|I\| = 10000$	FO óptimo	CPU time	Iteraciones
Weiszfeld	114816694429.10657	0.002910807132720947	14.54
Gradiente	114816694454.28467	0.03545771360397339	73.52
Newton	114819204822.87225	11.155067825317383	300.0
Minos			

Máximos

$\|I\| = 100$	FO óptimo	CPU time	Iteraciones
Weiszfeld	126956.82421530616	0.03744983673095703	19
Gradiente	126956.82953347823	0.0030210018157958984	37
Newton	178876.50672485668	0.3072376251220703	300
Minos			

$\lvert I\rvert = 1000$	FO óptimo	CPU time	Iteraciones
Weiszfeld	118721338.38098419	0.0017211437225341797	14
Gradiente	118721338.5063968	0.006013154983520508	45
Newton	118820263.85204731	1.2746937274932861	300
Minos			

$\lvert I\rvert = 10000$	FO óptimo	CPU time	Iteraciones
Weiszfeld	116300232751.3486	0.004023075103759766	16
Gradiente	116300232775.26857	0.04574108123779297	79
Newton	116301244957.0392	11.668357133865356	300
Minos			