Problema de Transferencias Mínimas usando Recocido Simulado

César Lara

Heurísticas de Optimización Combinatoria, UNAM

Noviembre 2019

1 Problema de Transferencias Mínimas

Se tiene una gráfica dirigida de transferencias bancarias, cada nodo de la gráfica representa a una persona dentro del sistema bancario. Cuando la persona j le debe X dinero a la persona i, esto es representado con una arista con peso X que sale de j a i. Una persona puede deber dinero y a la vez tener personas que le deban. El objetivo es proponer una gráfica de solución que **minimice** el numero de aristas en el sistema y que nadie pague mas o menos de lo que debe ni que nadie reciba mas o menos de lo que le deben. No importa quién pague a quién, lo que importa es que los balances de cada persona sean correctos.

1.1 Cotas

Para este problema se consideran instancias de a lo mas dos mil personas. No hay cota en la cantidad de dinero que fluye.

1.2 Conceptos

Un **acreedor** es aquel cuya suma de cantidades que le deben es mayor a la cantidad de dinero que debe.
Por el contrario un **deudor** es aquel cuya suma de deudas es mayor a la suma que se le debe.
Una persona cuyo balance es 0 (no es ni acreedor ni deudor) es denominado un **inútil**.
Error del acreedor i: Es la diferencia absoluta entre el dinero que un acreedor i tiene que recibir, contra el dinero que recibe.
Error: Es la suma de los errores de todos los acreedores.
Solución viable: es aquella en la que el error es 0. Es decir, ningún acreedor tendrá mas o menos dinero del que necesita.

1.3 Visto como gráfica bipartita

Ya que tenemos los conceptos de acreedor y deudor, el problema se puede ver como una gráfica bipartita dirigida donde un conjunto son los acreedores y el otro los deudores. Es fácil notar que en una solución óptima, ningún acreedor da dinero ni ningún deudor recibe, así que todas las aristas van a ir de un conjunto a otro.

2 Algoritmo de aproximación

El algoritmo que será presentado a continuación no está probado ser un algoritmo de aproximación en el sentido estricto del termino, no se tiene una cota de qué tan cercanas serán las soluciones pero se le denomina así porque empíricamente es como se comporta.

2.1 Problema de la asignación

Dados trabajadores, máquinas y el costo de que un trabajador opere una máquina para todos los trabajadores y todas las máquinas, encontrar el apareamiento (de trabajadores a máquinas) de costo mínimo.

2.2 Algoritmo Húngaro modificado

Dado que el problema descrito anteriormente (viéndolo como gráfica bipartita) se parece mucho al problema de la asignación. En este caso los dos conjuntos a emparejar son los deudores y acreedores y el costo entre cada par es el error que induce que el deudor le de todo el dinero de la deuda al acreedor. Si un conjunto es mas grande que el otro, simplemente le creamos fantasmas al conjunto pequeño y asignamos un número extremadamente grande a cada arista que los toque.
Si aplicamos el algoritmo que resuelve el problema de asignación (el algoritmo Húngaro) a una instancia del PTM, nos va a dar el emparejamiento que da el error mínimo posible cuando tiene tamaño $min(|deudores|, |acreedores|)$. Esto por supuesto no da una solución viable porque el error puede no ser 0, cuando éste es el caso, los acreedores que tienen más dinero que el que deberían, ahora pasan a ser deudores (porque le deben dinero al sistema), los acreedores que aun no tienen todo el dinero que deberían, siguen siendo acreedores pero ahora el dinero que necesitan es menor o igual al de la iteración anterior, finalmente si el balance de un acreedor pasa a ser 0, se hace inútil y se ignora. Se repiten estos pasos hasta que el error del sistema es 0, hasta que toda deuda queda saldada y todas las personas están satisfechas.

2.2.1 Complejidad

El algoritmo Húngaro tiene de complejidad $\mathcal{O}(n^3)$ con n el número de trabajadores o bien de $\mathcal{O}(n^4)$ dependiendo de la implementación, en nuestro caso es

cúbica. El algoritmo Húngaro tendrá que ser aplicado tantas veces hasta que el error sea 0, el máximo numero de veces que puede pasar es $n-1$ donde n es el número de personas, ya que por mínimo, eliminamos solo a un deudor por iteración (excepto al final que el sistema no puede quedarse con solo una persona, ahí eliminamos 2). Así que la complejidad final de este algoritmo es de $\mathcal{O}(n^4)$

2.3 Resultados

Para las siguientes instancias se sabe por diseño que el número mínimo de transacciones es el número de deudores.

2.3.1 Instancia 6

Deudores: 10
Acreedores: 3
Transacciones calculadas por el algoritmo de aproximación: 11
Porcentaje de error: 10%

2.3.2 Instancia 5

Deudores: 200
Acreedores: 36
Transacciones calculadas por el algoritmo de aproximación: 218
Porcentaje de error: 9%

2.3.3 Instancia 4

Deudores: 500
Acreedores: 89
Transacciones calculadas por el algoritmo de aproximación: 542
Porcentaje de error: 8.4%

2.3.4 Instancia 3

Deudores: 1000
Acreedores: 183
Transacciones calculadas por el algoritmo de aproximación: 1087
Porcentaje de error: 8.7%

2.3.5 Instancia 2

Deudores: 1000
Acreedores: 179
Transacciones calculadas por el algoritmo de aproximación: 1095
Porcentaje de error: 9.5%

2.3.6 Instancia 1

Deudores: 1000
Acreedores: 181
Transacciones calculadas por el algoritmo de aproximación: 1080
Porcentaje de error: 8%

2.4 Conclusión

Como se puede ver, empíricamente este funciona como un algoritmo de aproximación que no está equivocado por mas del 10% de la solución óptima, con tiempo de ejecución de menos de un minuto para las instancias más grandes. Esto es una aproximación bastante buena para los estándares de algoritmos de aproximación. Cabe aclarar de nuevo que es pura especulación que éste sea un algoritmo de aproximación, no sabemos en realidad.

3 Recocido Simulado

3.1 Modelado del problema

3.1.1 Solución inicial

Dada la gráfica bipartita de acreedores y deudores, hacemos que todos los deudores paguen toda su deuda de manera random. Por supuesto que el error será distinto de 0 y muy probablemente muy grande.

3.1.2 Función vecino

Dada una solución, una solución vecina será tomar un deudor al azar, dos acreedores al azar y si el deudor le esta pagando algo al primero, le pasamos todo o una parte de esa paga al segundo. De hecho, con probabilidad de 3/4 se le transferirá todo. De esta manera la instancia del problema sigue siendo la misma, no cambiamos ni deudores ni acreedores, solo aristas.
Por supuesto esto puede volverse un problema muy muy grande si la cantidad de dinero que se mueve es grande. Es un problema que estamos dispuestos a aceptar.

3.1.3 Función de costo

Para este problema tenemos dos objetivos: Minimizar el número de aristas y hacer que el error sea 0. La función de costo refleja justo esto, es una combinación lineal de una medida normalizada del error y una medida normalizada del numero de aristas. Actualmente es $error * n^2 + aristas * n^{(1.25)}$. Recordemos que n es el numero de personas en la instancia.

3.2 Experimentación

Fue muy difícil llegar a encontrar un tuneo que no diera resultados altamente malos. Se inició poniendo una temperatura de $100n$ y una función de costo que no tenía parámetros normalizados. Se tenía como solución una gráfica con 4000 aristas cuando la solución óptima son 200 (instancia 5). Pésimo. Noté que la temperatura podía ser reducida mucho más y al final se dejó en $n/4$. Normalizar ayudó un poco y también poner el tamaño de los lotes en $2000n$.
Lo que más ayudó a mejorar los resultados (que en este punto daban 2000 aristas para una solución óptima de 200) fue modificar la función de costo, se intentaron varios valores distintos en la combinación lineal del error y numero de aristas normalizados, nada ayudaba mucho o bien completamente se empeoraba. Finalmente llegue a que $error * n^2$ reducía rápidamente el error sin sacrificar tanto las aristas. Una vez con el error así, intente poner $aristas * n^2$ pero esto ocasionaba que el error se quedara muy alejado de cero en varias ocasiones así que después de jugar un poco con el sistema, se quedó en $aristas * n^{(1.25)}$.
Otra cosa que igual hizo gran diferencia fue introducir que se transfiriera todo el dinero que un deudor le estaba pagando a un acreedor a otro acreedor con probabilidad de 3/4. Así finalmente se obtuvieron resultados cercanos en la instancia 5, de al rededor de 280, cuando la óptima es de 200.

3.3 Resultados

Todos estos resultados representan soluciones viables.

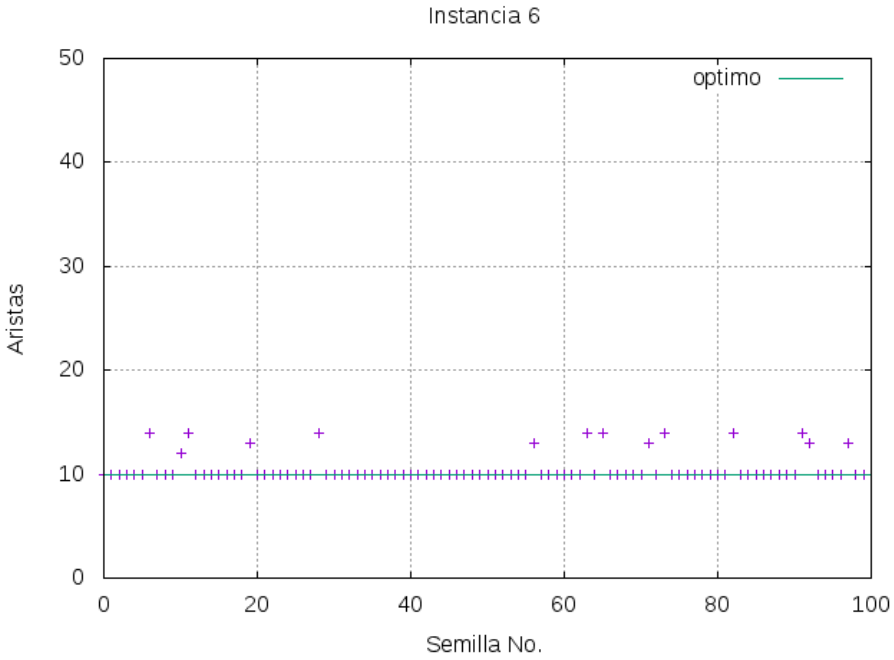

Figure 1: Resultados de la instancia 6

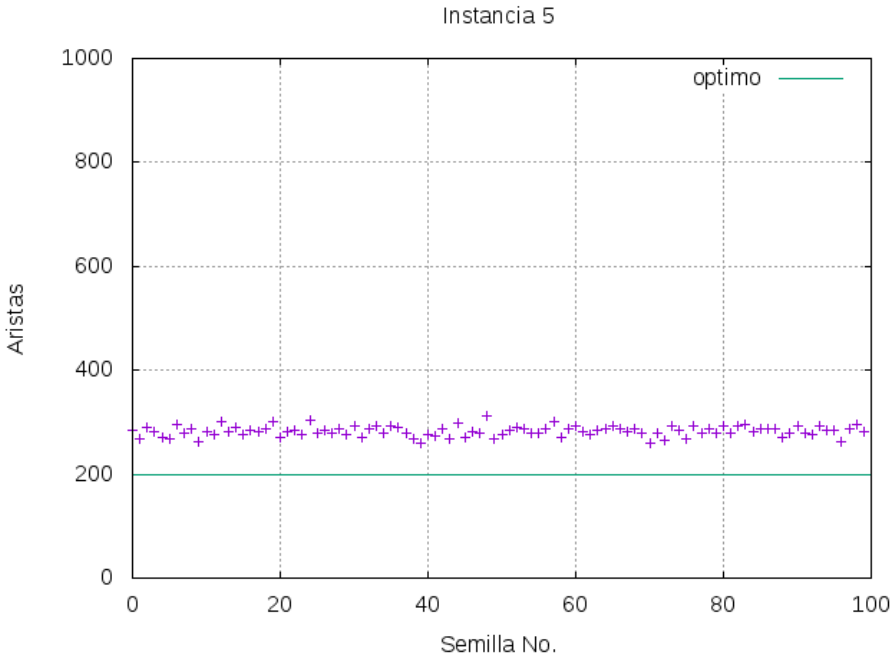

Figure 2: Resultados de la instancia 5

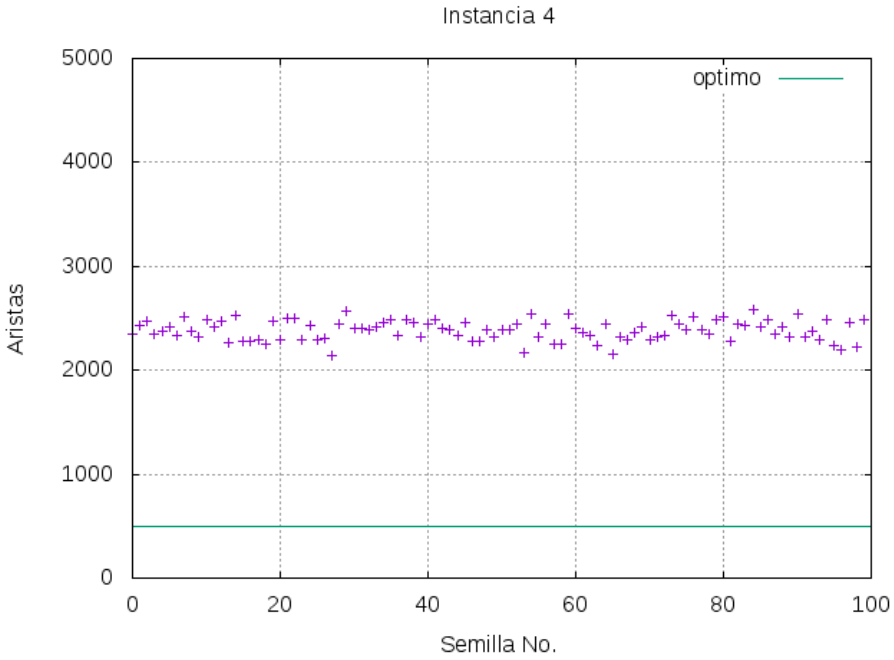

Figure 3: Resultados de la instancia 4

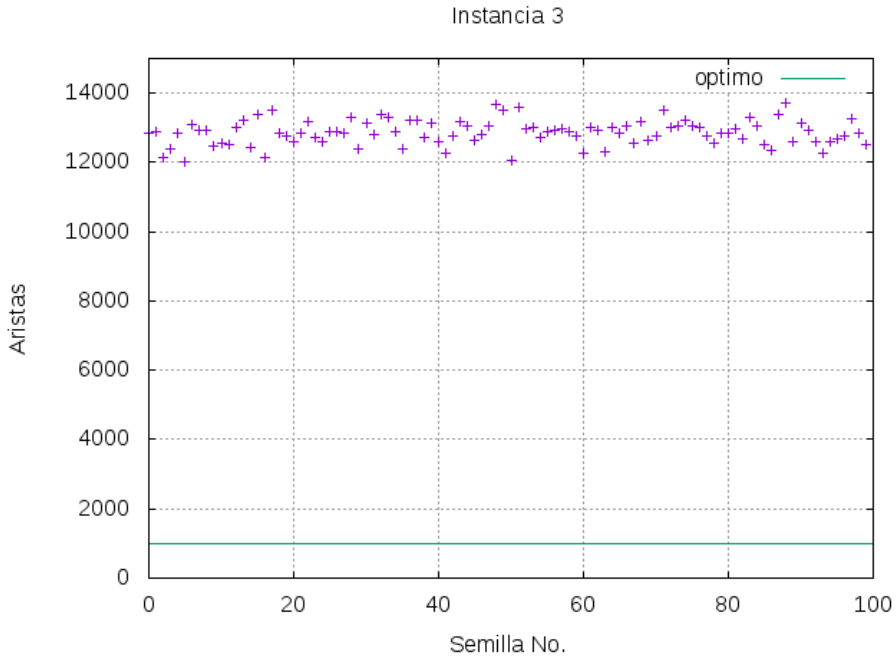

Figure 4: Resultados de la instancia 3

3.4 Conclusiones

Completamente están muy alejados los resultados obtenidos de la solución óptima, en general no esta bien tuneada la heurística e incluso puede que haya sido una mala elección modelarla de la manera en que se modeló. Igual cabe destacar que estos resultados se obtuvieron con tamaños de batches relativamente pequeños para que corrieran más rápido las soluciones y poder correr distintas instancias, igual no creo que con un batch mas grande se mejore demasiado la solución.
Lo que me da curiosidad es que todos los resultados parecen seguir una misma linea, no están esparcidos de una manera menos uniforme, no sé a qué se deba esto y definitivamente no esperaba algo así.
Igualmente no es alentador que incluso en una instancia tan chica como la instancia 6, no mas del 99% de semillas llegue a la solución óptima.

3.5 Experimentación futura

3.5.1 Solución inicial distinta

Una idea que podría funcionar bastante bien es aplicar el algoritmo Húngaro una vez a la solución inicial para reducir el error y en base a esa nueva solución aplicar un recocido con una temperatura pequeña. Esto probablemente pueda

resultar en en soluciones mucho mejores de las que se tienen actualmente. Desgraciadamente la falta de tiempo no permitió intentar esto.

3.5.2 Algoritmo Húngaro modificado estocástico

Otra idea es cambiar la instancia del problema inicial dándole todo el dinero de un deudor (o varios) a un(os) acreedor(es) con probabilidad inversamente proporcional al error que la operación induciría para después aplicar el algoritmo Húngaro modificado en esta nueva instancia, con suerte las soluciones obtenidas son mejores que si se le hubiera aplicado el algoritmo Húngaro modificado al inicio. Por supuesto esta técnica se tendría que intentar varias veces con semillas distintas para probar suerte. Esto no es una heurística como tal pero valdría la pena intentarlo.

References

[1] https://en.wikipedia.org/wiki/Assignment_problem