Solución del problema de producción "Temprana-Tardía" (Earliness/Tardiness) con los métodos de discretización

José Eduardo Díaz Moreno
Trabajo dirigido por Alina Fedossova, Ph. D.
Departamento de Matemáticas, Universidad Nacional de Colombia
24 de junio de 2022

Abstract

Se estudia el modelo matemático para el problema de producción 'temprano-tardía' (earliness/tardiness) como un problema de Optimización Semi-Infinita (SIP), buscando la construcción de un algoritmo de discretización para la solución del problema. Luego se realizarán experimentos numéricos para comparar los resultados con los obtenidos con otros métodos. El algoritmo que se utilizará es el de SQP el cual es la base de la rutina *fseminf* de Matlab. Se aplicará a datos originales del artículo inicial de estudio y datos nuevos de una empresa colombiana.

1 Preliminares

La Optimización Semi-Infinita (*SIP*) trata problemas en los cuales existen una cantidad finita de variables e infinitas restricciones. Muchos problemas de ingeniería y logística contienen por lo menos una restricción y un parámetro, como el tiempo, que varía en dominio.

En una empresa o negocio de cualquier índole cada vez es más grande la necesidad de tener una correcta logística para la creación y entrega de productos a un cliente. Un cliente es un recibidor de un producto, ya sea cliente intermedio o final. Una forma acertada de pensar en la producción típica de una empresa de manufactura es que inicia con la fase de preparación de producción, seguido del proceso de manufactura y finaliza con un estadio de ensamblaje.

La solución a la que queremos llegar tiene como principal objetivo aplanar la curva de crecimiento de recursos requeridos para cada tarea realizada, pues al tener usos no adecuados, no mesurados o no controlados, esta curva puede llegar a crecer hasta superar la brecha de los recursos disponibles inicialmente, lo que requiere una nueva inversión no prevista de capital para terminar el trabajo satisfactoriamente.

Este método se conoce como Just-In-Time (JIT) y viene de los años 60 en el Japón en un escenario perteneciente a la posguerra en el que los recursos, aún limitados, debían ser administrados de la mejor manera posible para asegurar un rendimiento e inversión acordes. El método es mencionado en una gran cantidad de artículos y libros de negocios convirtiéndose en más que una sugerencia, llegando a ser una norma aunque no siempre se aplique de manera correcta.

El problema *SIP* se plantea a continuación. El objetivo es minimizar la función f dada una función g (extendible a más restricciones).

$$f(x) \to min_x, \quad x \in X$$
$$\text{t.q. } g(x,y) \leq 0, \quad \forall y \in Y$$

Donde $f : \mathbb{R}^n \to \mathbb{R}$ y $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ y $X \subseteq \mathbb{R}^n$ y $Y \subseteq \mathbb{R}^m$, $|Y| < +\infty$. Y, además, f y g son continuas y diferenciables.

Existen diferentes métodos para tratar el problema y serán expuestos más adelante, pero aquí nos quedaremos con la solución por el método SQP (Sequential Quadratic Programming) (Programación Cuadrática Secuencial). Es un método iterativo que usa la idea del método de Newton y lo generaliza. En él, se escoge primero un vector solución inicial que funge de semilla, luego se calcula el lagrangiano de f, se determina su gradiente, se procede a igualar a cero y se busca resolver esta ecuación.

$$\nabla L(x, \lambda) = 0$$

Que es equivalente a:

$$\begin{bmatrix} \nabla^2_{xx}\,\mathcal{L} & \nabla J_g^T(x) \\ \nabla g(x) & 0 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta\lambda \end{bmatrix} = - \begin{bmatrix} \nabla_x\,\mathcal{L} \\ g(x) \end{bmatrix}$$

Las iteraciones se realizan para actualizar x_ky λ_k al aparecer δ_x y δ_y ya que son las variables de incrementos:

$$\begin{pmatrix} x_k \\ \lambda_k \end{pmatrix} = \begin{pmatrix} x_{k-1} \\ \lambda_{k-1} \end{pmatrix} + \begin{pmatrix} \delta x \\ \delta\lambda \end{pmatrix}$$

A partir de aquí se actualiza el vector inicial y se repite este proceso una cantidad suficiente de veces, en algunos casos sencillos, menos de cien. Pueden ocurrir casos en los que el vector inicial no haya sido bien escogido y las iteraciones aumenten mucho haciendo costosa su solución. Como también puede suceder que no se encuentre solución local ya que este método comparte esta característica con el de Newton, la de que la solución hallada es local. Otra característica importante a mencionar es que la convergencia de las soluciones es cuadrática, como en el método de Newton o de los cuasi-Newton.

¿Qué es SQP?

El método SQP (Sequential Quadratic Programming) es la base de *fseminf*, la rutina que hace parte del paquete de optimización de Matlab, la herramienta computacional que se usa en este trabajo para resolver el problema de Optimización Semi-Infinita (*SIP*).

Lo primero a tener en cuenta es que el sistema de ecuaciones no debe ser lineal, ya que para problemas lineales se procede con otros métodos del álgebra lineal. Sea f nuestra función a minimizar. El planteamiento general es el mismo de arriba:

$$f(x) \to \min_x, \quad x \in \mathbb{R}^n$$
$$\text{t. q.} \quad g(x) \leq 0$$

Con f siendo a función objetivo y $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$. Además, como vimos anteriormente, puede existir un número finito de ecuaciones de restricción. Para $n, m \in \mathbb{N}$. Suponemos que f y g son continuas y diferenciables en todo punto. Además, de modo general, suponemos que el conjunto Ξ de las restricciones (recordando que es una posibilidad tener n restricciones finitas o infinitas restricciones, incluso), forma un conjunto de las restricciones activas: $\Xi_{act} := \{g \in \Xi \mid g(x) = 0\}$ y a su vez forma un conjunto linealmente independiente $\{\nabla_x g(x) \mid g \in \Xi_{act}\}$. Las condiciones **KKT** aplicadas al método dicen que el Lagrangiano de f debe ser igual a cero, a lo que se sigue resolver esto para obtener puntos críticos.

2 El problema

Vamos a considerar el caso de una empresa manufacturera en la que se recibe una cantidad específica n correspondiente a órdenes de ciertos productos que deben ser fabricados en un tiempo determinado $[0, T]$. La **i-ésima** orden tiene cierto ciclo de producción L_i, interpretable como el tiempo que se demora la empresa en manufacturar el producto, y este a su vez cuenta con una fecha de entrega final específica d_i.

El propósito es el de implementar de una forma fiel el método o filosofía Just In Time (JIT). Para ello se debe garantizar que el tiempo para completar la producción sea lo más cercano al tiempo de entrega al cliente final dado que en este ambiente no sirve que un producto sea terminado antes de tiempo pues existiría la necesidad de almacenarlo y esto acarrearía un gasto adicional; de igual forma, si el producto se entrega tarde pueden llegar a existir cargos extra por la tardanza, además de que ninguna empresa quisiera demora. Es por esto que cualquier alejamiento del tiempo acordado se debe corregir y se puede abordar introduciendo el concepto de función de penalización cuadrática.

Esta función actúa dando un "peso" o "precio" a los tiempos de producción que no correspondan a lo esperado y los penaliza hasta que alcancen un valor exacto o un rango de valores aceptables. Indiscutiblemente, no se puede apuntar a la precisión perfecta en el tiempo de la entrega de productos, pero sí a minimizar esta función de penalidad.

El planteamiento del problema *SIP* nos permite implementar esta penalización con la restricción correspondiente. Como nuestro objetivo es minimizar, lo expresamos de la siguiente forma:

$$\min_x \sum_{i=1}^{n} \alpha_i (x_i - d_i)^2$$
$$\text{t. q. } \sum_{i=1}^{n} R_i(t, x_i) \leq G(t), \qquad t \in [0, T]$$
$$0 < L_i \leq x_i \leq T, \qquad i = 1, 2, \ldots, n$$

donde α_i corresponde al peso de la orden i-ésima y que usualmente es proporcional al valor de la orden i.

$$R_i(t, x_i) = a_i \exp\left(\frac{-(t - x_i + b_i)^2}{c_i}\right), \qquad i = 1, 2, \ldots, n.$$

Por las propiedades de la distribución normal, los coeficientes para $R_i(t, x_i)$ son:

$$a_i = \frac{P_i}{(\sqrt{2\pi} L_i/4)}$$
$$b_i = \frac{L_i}{2}$$
$$c_i = \frac{L_i^2}{8}$$

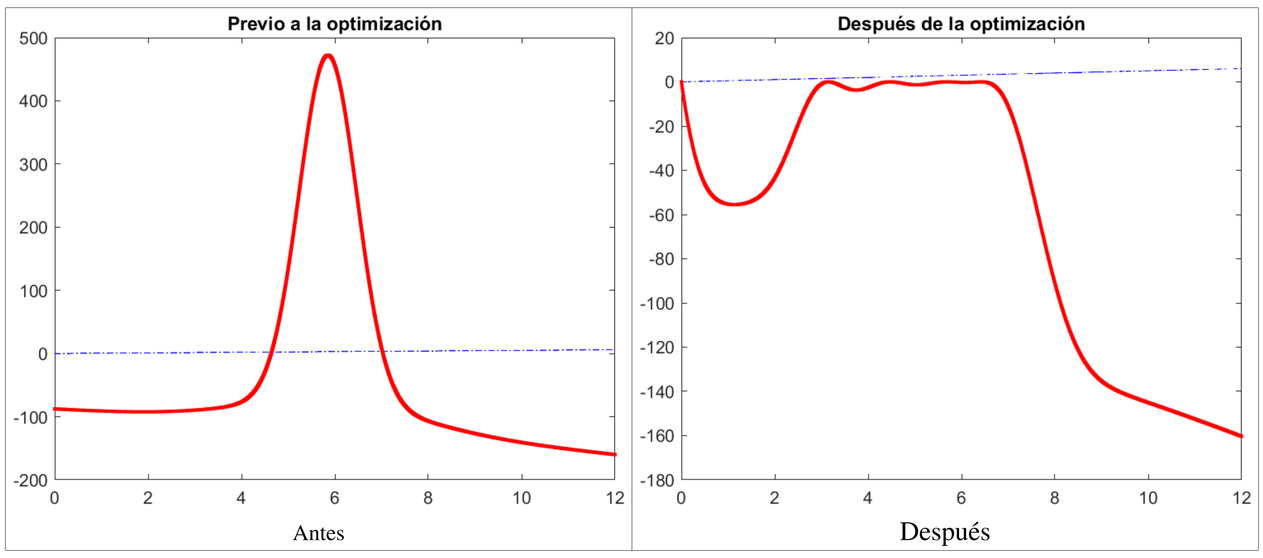
La función $G(t)$ planteada arriba, la cual nos muestra el uso de recursos, se modela entonces como:

$$G(t) = g_0\, exp(\beta t) - \sum_{i=1}^{n} Q_j(t), \qquad t \in [0, T]$$

Donde $Q_j(t)$ se define similar a $R_i(t)$.

3 Resultados

Esta aplicación es el caso de una empresa de colombiana llamada Zelo Vending la cual tiene 10 pedidos para rellenar máquinas de vending. El modo de operación consiste en realizar pedidos de productos a las fábricas correspondientes (productos como gaseosas, aguas, papas fritas, galletas, entre otros), al recibirlos se empacan en canastas, una por máquina. Posteriormente, el repartidor las lleva a cada una de las locaciones planeadas. Aquí es importante notar que no se tiene en cuenta nada que tenga que ver con el problema del transporte (rutas), sino solo tiempos de llegada a llenar las máquinas.



En la gráfica podemos observar que antes de la aplicación del método los recursos no tienen una cota sino que se usan de manera indiscriminada. En la segunda gráfica que es g_0, el cual es el limitador de recursos al que le apuntamos. La minimización del tiempo de entrega real (vector resultado x) ayuda a que se cumplan los tiempos al cliente final y los recursos no sean malgastados, lo que beneficia a la empresa.
Una de las corridas del programa nos da este vector solución:

$$X_8 = \begin{bmatrix} 11.4436 & 4.4319 & 7.0655 & 5.3389 & 8.3072 & 0.9999 & 3.3154 & 9.7748 & 7.4575 & 6.0705 \end{bmatrix}$$

Junto con el valor solución de 1.53×10^4. Estos datos minimizan la función objetivo presentada arriba.

4 Conclusiones

Este método de solución simplifica el trabajo de otros al converger de manera rápida (cuadrática). El costo computacional comparado con otros métodos es menos por la cantidad menor de pasos y operaciones que hay que realizar. Otros métodos llevan más tiempo por esta razón.
El algoritmo de SQP es relativamente sencillo, contrario a otros métodos como algoritmos genéticos o SMETH.ACTIV.

References

[1] Y. Li and D. Wang. A Semi-Infinite Programming Model for earliness/Tardiness Production Planning with Simulated Annealing., 35-42., Pergamon, 1997

[2] D. Wang and S.-C. Fang. A Semi-Infinite Programming Model Earliness/Tardiness Production Planning with a Genetic Algorithm. 95-106., Pergamon, 1996

[3] J. C. Valencia., Earliness/Tardiness Production Problem using Semi-Infinite Optimization., 2019