Universidad de La Habana

Facultad de Matemática y Computación

Problema de optimización libre del contexto

Daniel Abad Fundora C-411
Anabel Benítez González C-411
Juan Carlos Espinoza Delgado C-411
Raudel Alejandro Gómez Molina C-411
Alex Sierra Alcalá C-411

Proyecto en github

Introducción

Los problemas de optimización en enteros (POE) son una rama de la optimización matemática que se ocupa de encontrar soluciones óptimas a problemas donde las variables de decisión solo pueden tomar valores enteros. Estos problemas se encuentran en una amplia variedad de áreas, como:

- Ingeniería: diseño de estructuras, planificación de producción, gestión de proyectos.
- Finanzas: selección de inversiones, planificación financiera, gestión de riesgos.
- Ciencias de la computación: criptografía, diseño de algoritmos, aprendizaje automático.

Los POE son generalmente más difíciles de resolver que los problemas de optimización continuos. Esto se debe a que la discreción de las variables enteras añade complejidad al problema.

En el presente trabajo abordaremos una propuesta de solución a problemas de optimización en enteros, con variables binarias (que cumplen ciertas restricciones) usando herramientas de teoría de lenguajes formales y autómatas.

Antecedentes

Este trabajo se basa en los resultados obtenidos en [1], en el cual se plantea un algoritmo polinomial para resolver el problema de la satisfacibilidad booleana libre del contexto, el cual se basa en la teoría de lenguajes formales y autómatas. En este trabajo se propone una adaptación a este problema para resolver problemas de optimización en enteros.

En [1], se transforma una fórmula booleana en una lista de instancias de variables, donde se asume que 2 instancias de una variable no tienen por qué tener el mismo valor de verdad. Luego se define un autómata que dada una cadena de 0 y 1 y una fórmula booleana, determina si se obtiene un valor de verdad para la fórmula booleana donde cada instancia de una variable toma el valor de verdad que se corresponde con la cadena de 0 y 1. Entonces para verificar que 2 instancias de una variable tengan el mismo valor de verdad se intersecta dicho autómata con una gramática libre del contexto obteniendo una autómata de pila. Después de esto se plantea un algoritmo para dado este autómata de pila, generar todas las posibles cadenas de 0 y 1 que se corresponden con una asignación de valores de verdad y como consecuencia se obtiene un generador de todas las posibles asignaciones de valores de verdad para una fórmula booleana. Luego para determinar si la fórmula es satisfacible solo queda verificar si este conjunto de soluciones es no vacío.

Una fórmula booleana se considera libre del contexto si para cualquier par de instancias de una variable x_i y x_j con $i < j$ se cumple que si existe otra variable con instancia x_k con $i < k < j$ entonces todas las instancias de esta nueva variable ocurren entre x_i y x_j.

El problema antes planteado ha sido generalizado en [2] y [3] donde en vez de una gramática libre del contexto se usa una gramática de concatenación de rango y gramáticas matriciales simples respectivamente, para ser intersectadas con el autómata booleano y al igual que en [1] obtener una mecanismo generador de soluciones que satisfacen la fórmula booleana.

Planteamiento del problema

Para utilizar el algoritmo propuesto en [1] para resolver problemas de optimización en enteros, una fórmula booleana puede ser interpretada como una restricción de un problema de optimización en enteros. Por ejemplo sea la fórmula booleana:

$$x_1 \vee x_2 \vee x_3 \vee x_4 \tag{1}$$

La restricción para que sea satisfacible puede ser interpretada de la siguiente manera:

$$x_1 + x_2 + x_3 + x_4 \geq 1 \tag{2}$$

donde los x_i son variables binarias.

Problema de optimización libre del contexto

Sea $x_1, x_2, ..., x_n$ un conjunto de variables binarias. Se desea minimizar la función objetivo:

$$f(x_1, x_2, ..., x_n) = c_1x_1 + c_2x_2 + ... + c_nx_n \tag{3}$$

Sujeto a las restricciones:

$$\sum_{i=1}^{n} b_{j,i}x_i \geq 1, \quad j = 1, 2, ..., m \tag{4}$$

donde los $b_{j,i}$ son 0 o 1. Además se cumple para todo k entre 1 y n que si existen i, j, z y p tal que $b_{i,k} = 1$, $b_{j,k} = 1$ y $b_{z,p} = 1$, tal que $i \leq z \leq j$ y $p \neq k$ entonces no existe q con $q < i \vee j > q$ tal que $b_{q,p} = 1$, en el caso de que $i = z$ se debe cumplir que $p > k$ y en el caso de que de que $j = z$ se debe cumplir que $p < k$.

Las restricciones de dicho problema de optimización se pueden plantear con un SAT libre del contexto (resultado obtenido en [1]) de la siguiente manera:

$$x_{a_{j,1}} \vee x_{a_{j,2}} \vee ... \vee x_{a_{j,k}}, \quad \forall i : b_{1,i} \neq 0, \quad j = 1, 2, ..., m \tag{5}$$

para este SAT se obtiene una gramática libre del contexto que genera todas las posibles asignaciones de valores de verdad para las variables x_i, que para nuestro problema de optimización se traduce en todas las posibles asignaciones de valores que satisfacen las restricciones.

Dicha gramática como se demuestra en [1] es un grafo acíclico dirigido, donde cada arista representa una producción y a su vez un valor de verdad para cada instancia de una variable. Luego para obtener el valor mínimo de la función objetivo se puede ponderar cada arista que represente la primera instancia de una variable con el valor de la función objetivo para dicha instancia y el resto de las aristas con 0. Luego identificar los estados nodos que representan los estados finales de la gramática y usar un algoritmo para minimizar el costo de llegar desde el distinguido a los estados finales.

El algoritmo propuesto será el algoritmo de Dijkstra, donde se considera el costo de llegar a un estado nodo como el valor de la función objetivo acumulado. En el caso de tener aristas negativas podemos utilizar la modificación del algoritmo para aristas de costo negativo (podemos garantizar el correcto funcionamiento de esta transformación ya que no existen ciclos y por tanto no existen ciclos de costo negativo) o también pudiéramos considerar el uso de Bellman Ford aunque esto implica una mayor complejidad.

Generalización del problema libre del contexto

El problema descrito en la sección anterior fue una transformación explícita del problema planteado en [1] para un problema de optimización. Ahora planteemos un nuevo problema de optimización con menos restricciones que el anterior y usaremos la misma idea que en [1] pero cambiando el autómata.

Sea $x_1, x_2, ..., x_n$ un conjunto de variables binarias. Se desea minimizar la función objetivo:

$$f(x_1, x_2, ..., x_n) = c_1x_1 + c_2x_2 + ... + c_nx_n \tag{6}$$

Sujeto a las restricciones:

$$\sum_{i=1}^{n} b_{j,i}x_i \geq a_j, \quad j = 1, 2, ..., m \tag{7}$$

donde los $b_{j,i} \in Z$. Además se cumple para todo k entre 1 y n que si existen i, j, z y p tal que $b_{i,k} \neq 0$, $b_{j,k} \neq 0$ y $b_{z,p} \neq 0$, tal que $i \leq z \leq j$ y $p \neq k$ entonces no existe q con $q < i \vee j > q$ tal que $b_{q,p} \neq 0$, en el caso de que $i = z$ se debe cumplir que $p > k$ y en el caso de que de que $j = z$ se debe cumplir que $p < k$.

La restricción eliminada no nos obliga a que los $b_{j,i}$ tomen valores 0 o 1. Por tanto el autómata booleano planteado en [1] no nos sirve para reconocer si una cadena de 0 y 1 satisface una restricción pero si hemos mantenido el orden libre del contexto de los x_i.

Ahora diseñemos un autómata que reconozca el lenguaje de las cadenas binarias que satisfacen una restricción del tipo:

$$\sum_{i=1}^{n} b_i x_i \geq a \tag{8}$$

Para ello respetemos el diseño seguido en [1] para la construcción del autómata booleano, la cual exige conocer para cada instancia de una variable x_i el estado que esta representa en el autómata. Ahora los estados del nuevo autómata representarán tuplas (i, k) donde i es el índice de la variable actual y k es la suma acumulada hasta el momento, luego para cada variable x_i el autómata tendrá $p + q$ estados donde

$$p = max(\sum_{j=1}^{n} b_j x_j \forall j < i) \tag{9}$$

$$-q = min(\sum_{j=1}^{n} b_j x_j \forall j < i) \tag{10}$$

para todas las posibles cadenas binarias. El estado inicial sería $q_{0,0}$ y las transiciones se definen de la siguiente manera:

$$f(q_{0,0}, 1) = q_{1,b_1} \tag{11}$$

$$f(q_{0,0}, 0) = q_{1,0} \tag{12}$$

$$f(q_{i,k}, 1) = q_{i+1,k+b_{i+1}} \tag{13}$$

$$f(q_{i,k}, 0) = q_{i+1,k} \tag{14}$$

y los estados finales serían los $q_{n,k} \forall k \geq a$.

El autómata planteado anteriormente tiene una cantidad de estados en el orden de $n \cdot (p+q)$, donde p y $-q$ son los valores de la suma máxima y mínima para todas las cadenas respectivamente, por lo que la complejidad de temporal de la construcción de dicho autómata es $O(n \cdot (p+q))$ lo cual impone otra restricción a nuestro problema ya que debemos garantizar que el valor de $p + q$ no sea un número muy grande ya que el algoritmo de construcción del autómata es un algoritmo pseudo-polinomial.

Ahora procedemos igual que en [1] a intersectar el autómata con una gramática libre del contexto y obtener un mecanismo generador de soluciones factibles para nuestro problema de optimización y siguiendo el método planteado en la sección anterior obtener el valor mínimo de la función objetivo mediante la ponderación del grafo resultante y la ejecución del algoritmo de Dijkstra.

Transformación de un problema de optimización en un problema de optimización libre del contexto

Sería interesante obtener un algoritmo que permita reordenar las restricciones de un problema de optimización general a un problema de optimización libre del contexto planteado en la sección anterior. En este trabajo no expondremos ningún algoritmo para ello pero si ejemplificaremos un caso particular donde esto es posible.

Sea $x_1, x_2, ..., x_n$ un conjunto de variables binarias. Se desea minimizar la función objetivo:

$$f(x_1, x_2, ..., x_n) = c_1x_1 + c_2x_2 + ... + c_nx_n \tag{15}$$

Sujeto a las restricciones:

$$\sum_{i=1}^{n} b_{j,i}x_i \geq a_j, \quad j = 1, 2 \tag{16}$$

podemos reordenar las restricciones del problema de la siguiente manera:

$$b_{1,1}x_1 + b_{1,2}x_2 + ... + b_{1,n}x_n \geq a_1, \tag{17}$$

$$b_{2,n}x_n + b_{2,n-1}x_{n-1} + ... + b_{2,1}x_1 \geq a_2, \tag{18}$$

como puede observarse con este reordenamiento garantizamos que las instancias de los x_i sigan un orden libre del contexto, luego podemos proceder a encontrar el mínimo valor de la función objetivo siguiendo el procedimiento descrito en la sección anterior.

Análisis con métodos clásicos

En esta sección se mostrará un análisis experimental sobre los métodos clásicos existentes para resolver los problemas de optimización antes planteados, para ello se utilizará **python** como lenguaje de programación y la biblioteca pulp del mismo que cuenta con varios métodos optimización implementados (el código fuente se encuentra en el repositorio de github).

Primeramente se trató de crear un generador de problemas de optimización libres del contexto, aunque se logró una implementación del mismo en la mayoría de los casos el conjunto de restricciones factibles era vacío por lo que se optó por crear ejemplos de manera manual.

En todos los ejemplos del problema de optimización generar el algoritmo empleado por pulp logró resolver el problema de forma rápida y eficiente, por lo que por esta parte no se encontró una mejora a los métodos clásicos por parte del algoritmo propuesto en este trabajo.

Por otro lado si se logró programar un generador de problemas de optimización para el caso del reordenamiento de variables expuesto en la sección anterior. En este caso se generaron 1000 matrices aleatorias de 2x1000 junto a sus respectivas restricciones y función objetivo, en todos los casos nuevamente el algoritmo propuesto en este trabajo no presenta mejores sobre los métodos clásicos.

Con lo experimentado claramente se demuestra que en la práctica lo planteado en este trabajo no es realmente factible ya que los problemas de este estilo no son realmente comunes y en caso de aparecer los métodos clásicos son eficientes para resolverlos.

Recomendaciones

Para futuros trabajos sobre el tema sería interesante plantear los problemas de optimización que se derivan de [2] y [3], los cuales serían generalizaciones del problema descrito en este trabajo, ya que el mecanismo reconocedor empleado en

ambos casos tiene un poder superior a la gramática libre del contexto empleada en [1] y además el autómata planteado en este trabajo se ajusta a las características del autómata booleano empleado en [2] y [3].

Además sería interesante diseñar algoritmos de reordenamiento de las restricciones para transformarlas en un orden libre del contexto como se analiza en este trabajo o cualquier otro mecanismo reconocedor de la teoría de lenguajes que permita seguir las mismas ideas.

Referencias

[1] Alina Fernández Arias, "El Problema de la Satisfacibilidad Booleana Libre del Contexto. Un Algoritmo polinomial.," La Habana, 2007.

[2] Manuel Aguilera López, "Problema de la Satisfacibilidad Booleana de Concatenación de Rango Simple". Jornada Científica Estudiantil MATCOM 2016

[3] José Jorge Rodríguez Salgado, Informe de las Prácticas Laborales: Gramáticas Matriciales Simples. Primera aproximación para una solución al problema SAT, 2019