Programación con Restricciones

Laura Rodríguez Lardiés y Carlota Sánchez González

Septiembre 2023

Resumen

En este documento trabajaremos la definición de la programación con restricciones y trataremos las diferencias entre este método de programación y la programación lineal. Desarrollaremos un ejemplo práctico que, además de explicarlo aquí, también crearemos un repositorio en GitHub que puede visitar **aquí**.

Índice

1. Introducción

La programación con restricciones es un paradigma para resolver problemas combinatorios que se basa en una amplia gama de técnicas de inteligencia artificial, informática e investigación operativa. Este tipo de programación se ha aplicado con éxito a problemas tan diversos como el análisis de la estructura del ADN, la programación horaria de los hospitales y la programación de la industria. En esta, un problema se ve como una serie de limitaciones sobre lo que podría ser una solución válida.

Vamos a profundizar en el concepto de programación de restricciones (CP), distinguiendo, además, entre problemas de satisfacción y problemas de optimización. Analizaremos las diferencias fundamentales entre la programación lineal (LP) y la programación con restricciones (CP), dos enfoques clave en la optimización y toma de decisiones, y examinaremos ejemplos ilustrativos. Llevaremos a cabo un caso práctico de programación con restricciones, a través del cual veremos cómo aplicar los conceptos y técnicas explicadas anteriormente.

2. Programación de Restricciones

La programación basada en restricciones (CP) es un paradigma de programación declarativo, es decir, en lugar de indicar cómo queremos que se hagan las cosas, nos limitamos a decir qué queremos que pase y el sistema se encargará de hacerlo. La CP es una técnica para encontrar la solución que respete un conjunto de restricciones predefinidas. Es decir, este método emplea algoritmos que encuentran todas las soluciones que quedan dentro de la región factible definida por las restricciones para un conjunto de variables de decisión.

La CP se conoce como una técnica para resolver eficientemente problemas de optimización discretos (números enteros). El CP ha encontrado aplicaciones en diversas áreas como en la optimización combinatoria o los problemas de asignación y programación. En este tipo de programación podemos utilizar restricciones especializadas, sirven para reducir dominios de variables de manera eficiente durante una búsqueda. Los solucionadores de CP no fuerzan el problema con una búsqueda exhaustiva, sino que combinan heurística y búsqueda combinatoria.

Depende de como enfoquemos la CP, obtenemos dos problemas diferentes: problemas de satisfacibilidad o de optimización.

2.1. Problemas de Satisfacibilidad

Si le damos importancia a la satisfacibilidad, en la solución del problema obtendremos un conjunto de valores que satisfacen las restricciones del problema, pero no significa que todas las soluciones sean óptimas. Es decir, buscamos sólo que la solución cumpla las restricciones dadas, no que sea la mejor de todas. Las tres técnicas más populares para los problemas de satisfacción de restricciones son el retroceso, la propagación de restricciones y la búsqueda local.

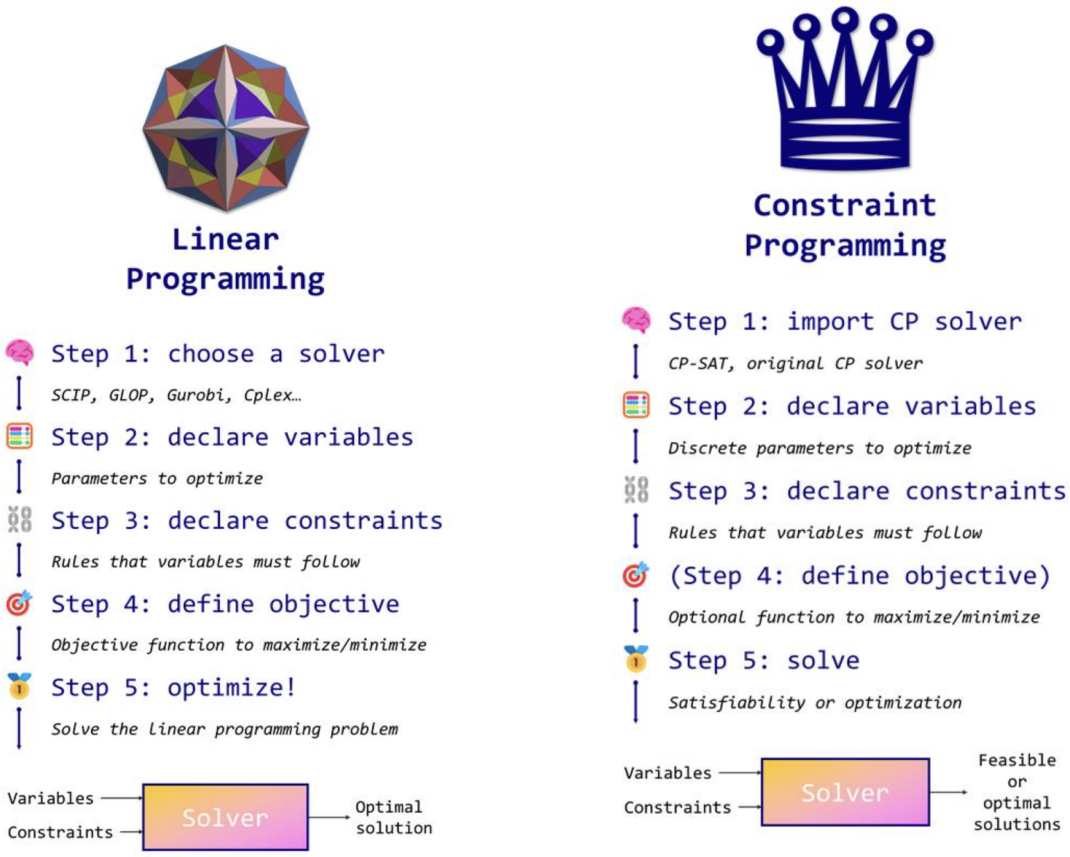
2.2. Problemas de Optimización

Los problemas de optimización de CP son como la optimización lineal (LP), en donde se busca la solución óptima dentro de un área definida por restricciones, la región factible. A diferencia de los problemas de satisfacibilidad, en los problemas de optimización se va más allá, puesto que no sólo buscamos que la solución cumpla las restricciones, sino que también buscamos la solución óptima de entre todas aquellas que quedan dentro de la región factible.

3. Diferencias entre LP Y CP

3.1. Objetivos y enfoque

La Programación de Restricciones (CP) y la Programación Lineal (LP) son dos enfoques fundamentales en la optimización y la toma de decisiones. A pesar de que ambos abordan problemas de optimización, difieren en su enfoque y en la manera en que resuelven los problemas. La CP se basa en la viabilidad en lugar de la optimización y se centra en las restricciones o dominios variables en lugar de la función objetivo. La diferencia principal es el algoritmo que se emplea para resolver un problema de una manera u otra:



A la hora de resolver un problema con programación lineal o programación con restricciones la sintaxis es bastante parecida. Estas dos técnicas están bastante cerca la una de la otra, ya que las dos manejan variables con restricciones y realizan optimización usando matemáticas y heurística. Sin embargo, CP se limita a parámetros discretos, como hemos mencionado antes, mientras que LP utiliza parámetros continuos. Por otro lado, la CP puede implementar restricciones especializadas como "todos diferentes", esto significa que no habrá dos variables con el mismo valor. Aquí hay una tabla de las principales diferencias entre las dos tecnologías:

	Programación matemática	**Programación con restricciones**
Solución óptima	Sí	Sí
Prueba de optimalidad	Cota inferior	Fuerza bruta
Tipo de problema	Todos	Discretos
Definición del problema	Requerido (LP, MIP)	No requiere

3.2. Ejemplos

3.2.1. Ejemplo Programación Lineal

Un ejemplo sencillo:

Supongamos que una empresa produce dos tipos de productos, A y B. El beneficio por unidad de producto A es de \$6, y el beneficio por unidad de producto B es de \$9. La empresa tiene restricciones en la cantidad de recursos disponibles, como horas de trabajo y materias primas. El objetivo es maximizar los beneficios totales.

La función objetivo sería:

$$\text{Maximizar } Z = 5A + 7B$$

Sujeto a restricciones lineales:

$$2A + 3B \leq 120$$

$$A, B \geq 0$$

3.2.2. Ejemplo Programación de Restricciones

Un ejemplo sencillo resuelto por CP es:

Al establecer un horario de clases en una escuela hay un conjunto de restricciones que deben cumplirse, como la disponibilidad de profesores y de aulas. El objetivo es programar las clases de manera que todas las restricciones se cumplan.

En este caso, se busca una asignación de horarios que sea viable y cumpla con todas las restricciones dadas.

3.3. Algoritmo de Resolución

Para resolver problemas de programación lineal, se utilizan algoritmos específicos como el método símplex. Estos algoritmos encuentran la solución óptima al problema, es decir, los valores de las variables que maximizan o minimizan la función objetivo.

Por otro lado, la programación con restricciones utiliza técnicas como la búsqueda y propagación de restricciones para encontrar soluciones que cumplan con todas las restricciones. No se trata de encontrar la solución óptima, sino de encontrar una solución factible.

En resumen, la programación lineal (LP) se enfoca en la optimización de una función objetivo sujeta a restricciones lineales, mientras que la programación con restricciones (CP) se enfoca en la satisfacción de restricciones sin una función objetivo. La elección entre LP y CP depende de la naturaleza del problema y de si el objetivo es encontrar la mejor solución o una solución viable que cumpla con todas las restricciones.

4. Ejemplo práctico

Para poner en práctica los conceptos que hemos visto a lo largo de este documento sobre la programación con restricciones, hemos desarrollado un código que resuelve el famoso **problema de las n-reinas**. Antes de comenzar a ver el código, expliquemos en qué consiste dicho problema.

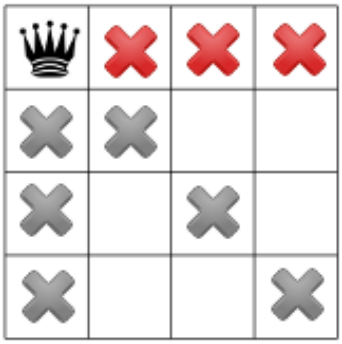
4.1. Problema de las N-reinas

El problema de las N-reinas es un problema combinado basado en el juego de ajedrez. Sabiendo que en el juego del ajedrez las reinas se mueven de adelante a atrás y en diagonal, el problema al que nos enfrentamos nos propone colocar N-reinas en un tablero NxN en donde ninguna reina esté amenazada. Por lo tanto, buscamos que ninguna de las reinas estén en la misma fila, columna o diagonal. Se trata de un problema de CP ya que no buscamos la solución óptima, queremos encontrar todas las posibles soluciones de este problema.

4.2. Enfoque

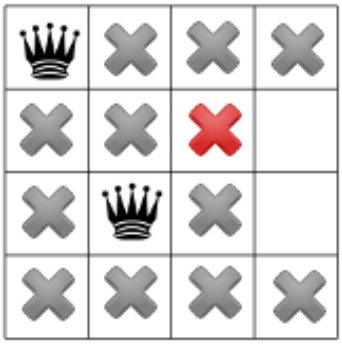
En la programación con restricciones, hay dos elementos clave a la hora de la búsqueda de soluciones: la propagación y el retroceso.

- Cada vez que el solucionador asigna un valor a una variable (en nuestro caso sería darle una posición a la reina), se agregan más restricciones a la siguiente variable y lo mismo ocurre con la siguiente. Es decir, existe una propagación de restricciones, las restricciones se propagan a futuras variables. Veamos el elemento de la propagación en la práctica, en nuestro ejemplo de las N-reinas, vemos que al colocar la primera pieza se crean tres nuevas restricciones para las siguientes piezas.

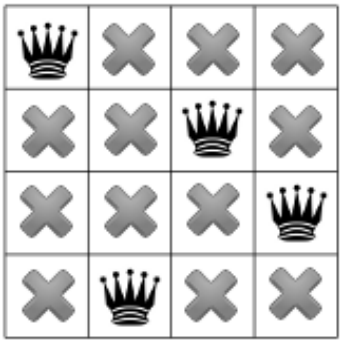


Como podemos ver en la figura, la siguiente pieza no podrá colocarse ni en la primera fila, ni la primera columna, ni la diagonal marcada. La primera pieza no contaba con estas restricciones y, además, la siguiente pieza que coloquemos añadirá más restricciones para la siguiente pieza.

- El retroceso ocurre cuando la solución no es factible, es decir, las restricciones establecidas no permiten que nuestro problema tenga las soluciones que buscamos. Por lo tanto, se retrocede a un paso anterior y se renombra la última variable definida. De esta manera las restricciones cambian y podemos encontrar nuestras soluciones.



En esta figura podemos ver que colocando la segunda reina en la tercera fila, hacemos que no se pueda colocar ninguna reina en la tercera columna, debido a que las restricciones lo impiden. En este caso se hace un retroceso, el cual cambia de posición la reina de la segunda columna para que así podamos tener soluciones factibles. Cambiando la posición de dicha reina, obtendríamos el siguiente resultado:



Ahora nos da problemas para colocar la última pieza, por lo que habría que hacer retroceso hasta la primera pieza y colocarla en otro lado.

4.3. Solución

Trabajaremos con las librerías sys, time y OR-Tools, que tiene un agente de resolución para programación con restricciones llamado CP-SAT. Por lo tanto, nuestro primer paso es importar dichas librerías.

```
import sys
import time
from ortools.sat.python import cp_model
```

El siguiente paso es declarar el modelo que vamos a usar, que en nuestro caso es CP-SAT.

```
modelo = cp_model.CpModel()
```

Una vez declarado el modelo, es hora de definir nuestras variables.

```
reinas = [modelo.NewIntVar(0, tablero - 1, f"x_{i}") for i in range(tablero)]
```

Agregamos las restricciones. El código usa el método AddAllDifferent, que requiere que todos los elementos de un conjunto de variables sean diferentes.

```
modelo.AddAllDifferent(reinas)
modelo.AddAllDifferent(reinas[i] + i for i in range(tablero))
modelo.AddAllDifferent(reinas[i] - i for i in range(tablero))
```

Veamos las restricciones que acabamos de codificar:

- Ninguna reina en la misma fila. Empleamos el método AddAllDifferent, que obliga a que las reinas de cada columna (reinas[j]) estén en filas diferentes (tengan una i diferente).
- Ninguna reina en la misma columna. Esto no queda codificado en la última parte de código que hemos visto, ya que dicha restricción ya viene dada por la manera en la que definimos la variable reinas.
- Ninguna reina en la misma diagonal. De nuevo, empleamos el método AddAllDifferent. En este caso, obtiene

Creamos una clase que va imprimiendo todas las soluciones del problema de las N-reinas a medida que las resuelve.

```
class Impresora(cp_model.CpSolverSolutionCallback):
    """Imprime las soluciones"""

    def __init__(self, reinas):
        cp_model.CpSolverSolutionCallback.__init__(self)
        self.__reinas = reinas
        self.__num_soluciones = 0
        self.__hora_comienzo = time.time()

    def num_soluciones(self):
        return self.__num_soluciones

    def on_solution_callback(self):
        hora_ahora = time.time()
        print(
            f"Solucion {self.__num_soluciones + 1}, "
            f"tiempo = {hora_ahora - self.__hora_comienzo} s"
        )
        self.__num_soluciones += 1

        todo_reinas = range(len(self.__reinas))
        for i in todo_reinas:
            for j in todo_reinas:
                if self.Value(self.__reinas[j]) == i:
                    print("R", end=" ")
                else:
                    print("_", end=" ")
            print()
        print()
```

Por último llamamos al agente de resolución o solucionador y calculamos los resultados.

```
solucionador = cp_model.CpSolver()
impresora = Impresora(reinas)
solucionador.parameters.enumerate_all_solutions = True
solucionador.Solve(modelo, impresora)
```

Pueden consultar y ejecutar el código completo en nuestro repositorio de GitHub, haciendo click **aquí**.

5. Bibliografía

- Aula virtual.
- Programación con Restricciones.
- ¿Qué es la Programación Basada en Restricciones?
- Restricciones Especializadas.
- Problema de las N-Reinas.