RELAZIONE SUL PROGETTO DI IMPLEMENTAZIONE DEL METODO DEL SIMPLESSO

I. Introduzione

Il progetto si propone di implementare il metodo del simplesso per la soddisfacibilità di un insieme di vincoli lineari con l'obiettivo di fornire una strategia efficiente per determinare se un insieme di vincoli lineari può essere soddisfatto o meno. Il metodo del simplesso è un algoritmo di ottimizzazione utilizzato per risolvere problemi di programmazione lineare; si basa sulla ricerca di soluzioni ammissibili attraverso iterazioni di pivotaggio, dove le variabili vengono scambiate. Nel caso di metodo del simplesso per la soddisfacibilità, ci si concentra sulla ricerca di una soluzione che soddisfi tutti i vincoli. A tale scopo, vengono assegnati valori alle variabili in modo iterativo, con l'obiettivo di trovare una configurazione alle variabili che soddisfi simultaneamente tutte le equazioni del sistema di vincoli.

II. Scelte implementative

Il linguaggio scelto per l'implementazione dell'algoritmo è stato Java. Sono state usate strutture dati tali che:

- **'double [][] tableau'**: rappresenta il tableau del metodo del simplesso.
- **'int [] nonBasicVariables'** e **'int [] basicVariables'**: contengono rispettivamente l'insieme delle variabili non di base e di base, rappresentati come indici.
- **'double [] b'**: contiene i termini noti del sistema.
- **'List<Integer> nonBasicIndex'**: è stata utilizzata per tenere traccia degli indici delle varibili non di base.
- **'HashMap<Integer, Double> beta'**: contiene una mappa indice -> valore, usata per memorizzare i valori di beta associati alle variabili sia di base che non di base e non solo, ma anche gli indici di tali varibili.

Un file è stato definito per memorizzare il sistema di vincoli iniziale, usando una classe di supporto per la lettura dei dati di input (matrice, termini noti, variabili di base e non di base).

III. Prototipo

Come detto in precedenza, l'obiettivo principale è quello di ottenere un assegnamento alle variabili che soddisfi simultaneamente tutti i vincoli del sistema lineare. L'algoritmo utilizza il metodo del simplesso per migliorare iterativamente la soluzione fino a raggiungere tale assegnamento, o concludere il problema se è insoddisfacibile.

Il prototipo dell'algoritmo proposto è presentato di seguito attraverso **uno pseudocodice ad alto livello,** offrendo una visione chiara e concisa delle principali logiche e passaggi fondamentali che caratterizzano questo metodo per la soddisfacibilità di vincoli lineari. Per l'implementazione, sono state progettate diverse classi:

- La classe **"Simplex"** è il cuore dell'implementazione. Fornisce un'astrazione chiara ed organizzata dei principali componenti e metodi necessari per risolvere il problema. In questa classe, la matrice dei coefficienti, gli array delle variabili di base e non di base, i termini noti e altri elementi cruciali, sono gestiti in modo efficiente permettendo la risoluzione iterativa del problema attraverso i metodi come **"pivotAndUpdate"** e **"check"**.
- La classe **"Main"** rappresenta il punto di ingresso del programma. All'interno di essa viene stampata la soluzione in base al risultato ottenuto.
- Nella classe **"ReadMatrix"** invece, viene fornito un metodo **"readMatrixAndConstantsFromFile"** che legge la matrice dei coefficienti da un file e restituisce un oggetto **"MatrixAndConstants"**. Quest'ultima facilita l'operazione di acquisizione dei dati necessari per eseguire il metodo del simplesso.
- La classe **"PivotResult"** rappresenta il risultato dell'operazione di pivoting nella classe **"Simplex".** Contiene inoltre le informazioni aggiornate sul tableau e sugli insiemi di variabili di base e non di base dopo l'operazione di pivoting.

Pseudo codice ad alto livello:

Classe PivotResult:

- A: matrice dei coefficienti

- N: array delle variabili non di base

- B: array delle variabili di base

Metodo costruttore PivotResult(A, N, B):

Inizializza i membri della classe con i valori forniti

Classe Simplex:

- tableau: matrice dei coefficienti

- nonBasicVariables: array delle variabili non di base

- basicVariables: array delle variabili di base

- b: array dei termini noti

- nonBasicIndex: lista degli indici delle variabili non di base

- beta: mappa dei valori beta associati alle variabili di base e non di base

```
Metodo costruttore Simplex(tableau, b, nonBasicVariables, basicVariables, beta):
    Inizializza i membri della classe con i valori forniti

Metodo update(xi, v):
    Aggiorna i valori di beta in base a xi e v

Metodo pivotAndUpdate(xi, xj, v):
    Calcola il theta e aggiorna i valori di beta e tableau
    Crea un nuovo oggetto PivotResult da passare al metodo pivot
    Chiama il metodo update per aggiornare i valori di beta
    Chiama il metodo pivot per ottenere un nuovo risultato

Metodo pivot(currentResult, enteringIndex, leavingIndex):
    Calcola una nuova matrice tableau dopo l'operazione di pivoting
    Restituisce un nuovo oggetto PivotResult

Metodo check(lowerBounds, upperBounds):
    Ciclo while:
        Conta il numero di variabili di base che soddisfano i vincoli;
        Se tutte le variabili di base soddisfano i vincoli, restituisci true;
        Altrimenti:
            Seleziona la prima variabile di base che viola i vincoli;
            Per ogni variabile non di base;
                Seleziona la prima variabile non di base adatta;
                Esegui il pivoting e interrompi il ciclo;
                Se nessuna variabile non di base è adatta, restituisci false

Metodo getAssignmentB():
```

Restituisce una stringa con l'assegnamento alle variabili di base

Metodo getAssignmentN():

Restituisce una stringa con l'assegnamento alle variabili non di base

Classe Main:

Metodo main:

Legge la matrice e le costanti da un file utilizzando ReadMatrix

Inizializza lowerBounds e upperBounds

Crea un oggetto Simplex con i dati letti

Risolve il problema invocando il metodo check con lowerBounds e upperBounds

Stampa la soluzione in base al risultato ottenuto

Classe ReadMatrix:

Metodo readMatrixAndConstantsFromFile(filePath):

Legge la matrice e le costanti da un file e restituisce un oggetto matrixAndConstants

IV. Esperimenti, risultati, tempi di esecuzione

Per valutare l'efficacia dell'algoritmo implementato, sono stati eseguiti una serie di esperimenti su diversi sistemi lineari. Di seguito, sono riportati i risultati ottenuti includendo la fonte del problema, la risposta e i tempi di esecuzione associati.

Problema	**Sistema**	**Fonte**	**Risposta**	**Tempo di esecuzione**
1.	x + 2y >= 1 2x + y >= 1 x +y <= 1	Appunti a lezione	Soddisfacibile	30 millisecondi
2.	x >= 1 2x <= 1	Appunti a lezione	Insoddisfacibile	29 millisecondi
3.	x + y >= 2 2x – y >= 0 -x + 2y >= 1	Appunti a lezione	Soddisfacibile	30 millisecondi
4.	x1 − 2x2 − 1/10x3 + 5x4 + x5 = 0	Paper:	Soddisfacibile	

	7/10x1 − 3/10x2 − 1/100x3 + 19/50x4 + x6 = 0 x1 + x2 + x3 + x4 + x7 = 5 x1 + 2x2 + 3x3 + x4 + x8 = 10 xj ≥ 0, j = 1…8.	https://arxiv.org/pdf/2101.01805.pdf.		35 millisecondi
5.	x1 − 32x2 − 4x3 + 36x4 + x5 = 0 x1 − 24x2 − x3 + 6x4 + x6 = 0 xj ≥ 0, j = 1…6.	Paper: https://arxiv.org/pdf/2101.01805.pdf.	Soddisfacibile	30 millisecondi
6.	x1 + x2 + 1/3x5 + 1/3x6 = 2 9x2 + x3 − 9x4 − 2x5 − 1/3x6 + x7 = 0 x2 + 1/3x3 − 2x4 − 1/3x5 − 1/3x6 + x8 = 2 xj ≥ 0, j = 1…8.	Paper: https://arxiv.org/pdf/2101.01805.pdf.	Soddisfacibile	27 millisecondi
7.	1/40x1 + 1/400x2 + 3x3 + 2x4 + x5 = 0 1/20x1 + 9/200x2 − 1/2x3 + 2/25x4 + x6 = 0 xj ≥ 0, j = 1… 6.	Paper: https://arxiv.org/pdf/2101.01805.pdf.	Soddisfacibile	40 millisecondi
8.	x1 − 2x2 − 1/10x3 + 5x4 + x5 = 0 7/10x −3/10x2 −1/100x3 +19/50x4 +x6 = 0 xj ≥ 0, j = 1…6	Paper: https://arxiv.org/pdf/2101.01805.pdf.	Soddisfacibile	35 millisecondi
9.	3x + 2y >=10 -x + 4y = 5	Internet	Soddisfacibile	35 millisecondi
10.	2x – 3y <= 7 x + 4y >=5 -3x + 2y = 1	Internet	Soddisfacibile	29 millisecondi
11.	0.5x1 − 5.5x2 − 2.5x3 + 9x4 + x5 = 0 0.5x1 − 1.5x2 − 0.5x3 + x4 + x6 = 0 x1 + x2 + x3 + x4 + x7 = 1 xj ≥ 0, j = 1…7.	Paper: https://arxiv.org/pdf/2101.01805.pdf.	Soddisfacibile	36 millisecondi
12.	x1 − 2x2 − 1/10x3 + 5x4 + x5 = 0 7/10x1 − 3/10x2 − 1/100x3 + 19/50x4 + x6 = 0 x1 + x2 + x3 + x4 + x7 = 5 x1 + 2x2 + 3x3 + x4 + x8 = 10 xj ≥ 0, j = 1…8.	Paper: https://arxiv.org/pdf/2101.01805.pdf.	Soddisfacibile	34 millisecondi
13.	3x + 2y <= 12 -2x +y >= 3 x – y <= 5 2x + 3y >= 8	Internet	Soddisfacibile	33 millisecondi
14.	0.4x1 + 0.2x2 − 1.4x3 − 0.2x4 + x5 = 0 −7.8x1 − 1.4x2 + 7.8x3 + 0.4x4 + x6 = 0 −20x2 + 156x3 + 8.0x4 + x7 = 1	Paper: https://arxiv.org/pdf/2101.01805.pdf.	Soddisfacibile	30 millisecondi

15.	2x – 2y + z = -3 x + 3y -2z = 1 3x – y – z = 2	Internet	Soddisfacibile	35 millisecondi

V. Osservazioni sugli esperimenti

Durante gli esperimenti, si è notato che la permutazione delle variabili non cambia il risultato (soddisfacibile/non soddisfacibile) ma vengono modificati gli insiemi di base e non di base con i rispettivi assegnamenti. Non vengono inoltre modificati i tempi di esecuzione e si è notato che il crescere del numero di vincoli porta ad un aumento delle dimensioni della matrice dei coefficienti; il che rende l'operazione di risoluzione del sistema più onerosa in termini di risorse computazionali.