Universitat de Girona
Escola Politècnica Superior

Grau en Enginyeria Informàtica

PROJECTE FINAL DE GRAU

Implementació i millora d'un algoritme de Time-Interval-Related-Pattern (TIRP) Mining incremental per ser executat en paral·lel i en C++

Autor:
Pau Calvera Moliner

Tutor:
Beatriz López Ibañez

RESUM

Convocatòria:
Juny 2023

Departament :
Enginyeria Elèctrica, Electrònica i Automàtica

Projecte: Projecte Final de Grau
Document: Resum
Títol: Implementació i millora d'un algoritme de Time-Interval-Related-Pattern (TIRP) Mining incremental per ser executat en paral·lel i en C++
Autor: Pau Calvera Moliner
Data: Juny 2023

Estudi:
Grau en Enginyeria Informàtica
Universitat de Girona

Supervisor:
Beatriz López Ibañez
Universitat de Girona
Email: beatriz.lopez@udg.edu

Índex

Capítol 1

Resum

1.1 Introducció

Els algortimes de mineria de "Time-interval-related pattern (TIRP) consisteixen en trobar patrons de relacions temporals tal com "A comença B". El seu ús pioner era trobar patrons de compres, però també pot tindre utilitat en altres camps, com per exemple per fer detecció i classificació en les seqüencies d'ADN, detectar atacs i/o buscar signatures dels virus, prediccions de preus dels estocs, etc. TIRP mining és una família de mètodes de "sequential patern mining"que permeten minar els patrons amb relacions totals o parcials en intervals temporals d'esdeveniments. El problema que presenten aquesta classe d'algoritmes és que tenen un cost computacional elevat i és per això que es fa de manera incremental: en aquest cas al tractar-se d'una estructura de dades en arbre, es computa per nivells.

En l'article [1], s'introdueix un nou algoritme per minar TIRPS anomenat "vert-TIRP", que combina una representació eficient d'aquests patrons utilitzant les propietats de transitivitat que tenen, amb una estratègia d'aparellament que ordena les relacions temporals per així accelerar el procés de mineria. A més presenta una definició robusta de les relacions temporals que elimina les ambigüitats entre relacions fent ús d'un valor "èpsilon". En l'article esmentat es pot trobar un enllaç del repositori de l'algoritme implementat en python.

1.2 Objectius

Com s'explica en la secció anterior, aquesta classe d'algoritmes tenen un cost computacional molt elevat. L'objectiu del projecte és implementar l'algoritme "vert-TIRP"en C++ per disminuir el temps de càlcul i comparar els resultats amb el "vert-TIRP"original. El llenguatge C++ és conegut per ser ràpid i eficient, a més disposa de punters que millorarien significativament l'eficiència i diverses opcions de multithreading.

1.3 Procés de desenvolupament del PFG

Com que l'objectiu principal d'adaptar l'algoritme a C++ és millorar-ne l'eficiència, s'ha dedicat molt esforç a optimitzar el codi per reduir el temps d'execució. Això s'ha aconseguit utilitzant referències sempre que sigui possible, punters, estructures de dades específiques i altres tècniques. S'ha programat el codi des de zero perquè sigui més comprensible, mantenint l'estructura, noms de variables i classes originals. No obstant això, alguns aspectes poden ser més complicats de comprendre a causa de la nomenclatura del llenguatge, especialment en les classes que fan servir iteradors i comprovacions d'elements en mapes.

El procés de traducció d'un algoritme, requereix d'un anàlisi previ important. S'ha de conèixer bé com funciona, tant el codi com els llenguatges de programació que s'utilitzaran. Una vegada realitzat l'anàlisi, s'ha començat creant una base de l'estructura de l'algoritme i s'ha anat implementant classe per classe seguint una metodologia *Ágile* dividida en *sprints*. Una vegada traduït el codi, s'ha implementat la paral·lelització i s'ha optimitzat el codi per al nou llenguatge. En la figura 1.1 es mostra el diagrama de classes de la nova implementació en C++.

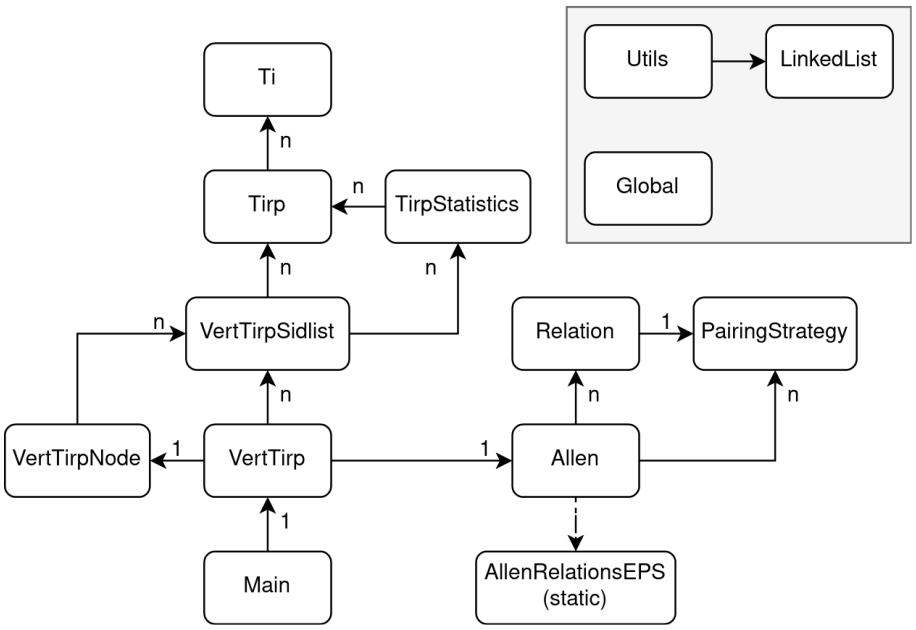

FIGURA 1.1: Diagrama de classes de l'algoritme en C++.

Per la paral·lelització, s'ha optat per *OpenMp*[2]. Es tracta d'una API per programar en paral·lel en C++. S'ha decidit utilitzar CPU en comptes de GPU a partir de l'anàlisi previ ja que s'ha considerat que beneficiaria més al temps d'execució de l'algoritme per l'estructura i funcionament d'aquest.

1.4 Resultats

1.4.1 Temps d'execució en seqüencial

Els resultats d'adaptar l'algoritme a C++, són bastant positius, ja que hi ha una diferència notable en el temps d'execució del codi. S'han fet diverses proves de rendiment en seqüencial i la millora de velocitat és de fins a un 11.814%, és a dir, el temps d'execució en C++ representa més o menys un 1% del temps que tarda l'algoritme en python sota les mateixes condicions. La figura mostra una comparació del temps entre els dos algoritmes amb el dataset *ASL*[3].

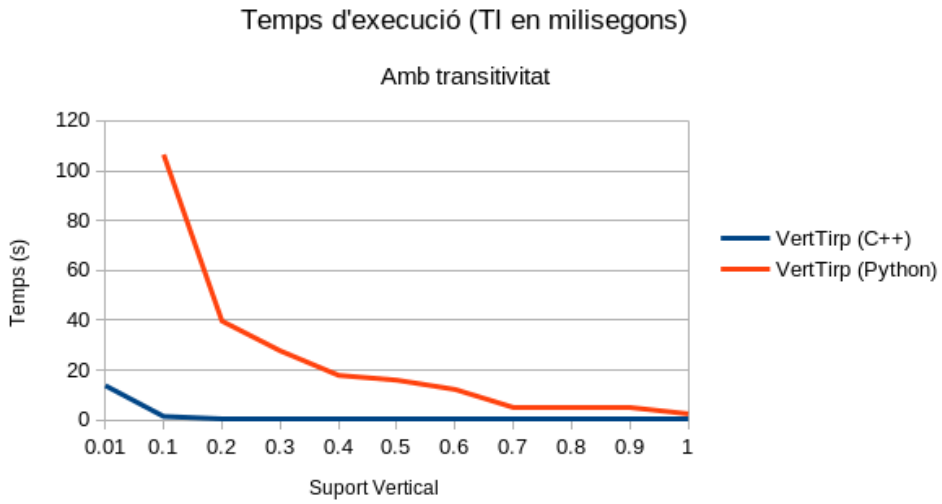

FIGURA 1.2: Temps d'execució del dataset *ASL* amb intervals de temps en milisegons i transitivitat. (per més informació consultar la memòria del projecte)

1.4.2 Temps d'execució en paral·lel

La parel·lelització de l'algoritme no ha resultat com s'esperava. Si bé en els millors casos, com mostra la figura 1.3, es pot observar una millora de fins al 90% , en altres casos la millora ha estat menor o fins i tot ha empitjorat el temps. A més, cal tenir en compte que aquesta millora del 90% és en el millor dels casos i amb un augment de 1 a 6 *threads*, per tant no és gaire substancial.

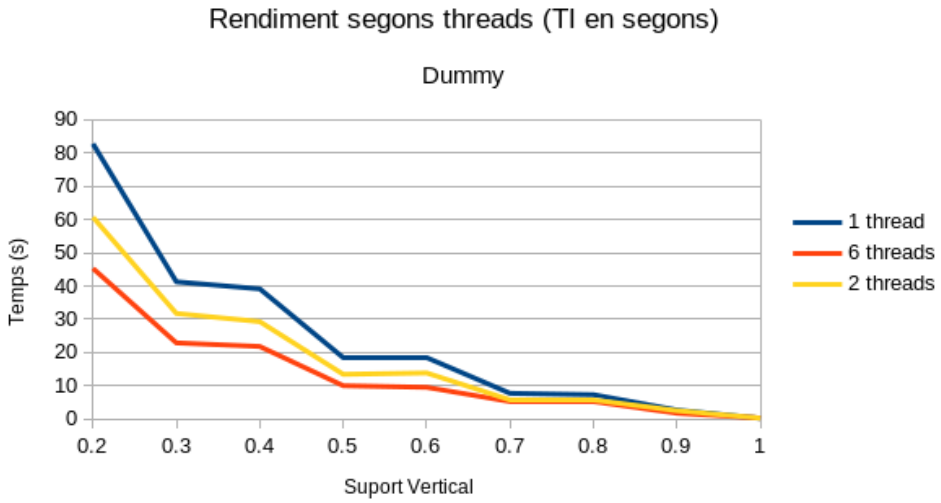

FIGURA 1.3: Diferència en el temps d'execució del vertTIRP en c++ amb el dataset *ASL* amb intervals de temps en segons i en *dummy*, en funció del nombre de threads. (per més informació consultar la memòria del projecte)

1.5 Conclusions

L'objectiu principal del projecte era millorar el temps d'execució de l'algoritme i els resultats han estat positius. Una de les principals complicacions, especialment al principi, va ser comprendre adequadament el codi. Abans de començar a programar, es va haver de fer un anàlisi en profunditat del funcionament i els conceptes, tant del codi original com de l'article que l'explica[1].

S'ha intentat que el codi sigui el més comprensible possible, seguint l'estructura original, amb els mateixos comentaris i noms de les variables. No obstant això, el C++ és un llenguatge més caòtic i alguna part pot ser complicada de comprendre. El codi està penjat en un repositori públic [4].

Els resultats de l'adaptació de l'algoritme són molt bons en termes de temps d'execució i ús de memòria. Es poden obtenir els mateixos resultats en només un 1% del temps requerit per l'original. No obstant això, en l'apartat de paral·lelització, la millora no és molt substancial i hi ha marge a millores. En conclusió, es considera que s'han assolit els objectius proposats i es mostra satisfacció amb els resultats.

Bibliografia

[1] Natalia Mordvanyuk, Beatriz López i Albert Bifet. "vertTIRP: Robust and efficient vertical frequent time interval-related pattern mining". A: *Expert Systems with Applications* 168 (2021). Consultat: 11 de maig de 2023, pàg. 114276.

[2] *OpenMp*. https://www.openmp.org/. Consultat: 11 de maig de 2023.

[3] Universitat de Boston. *ASLLRP Database*. https://www.bu.edu/asllrp/. Consultat: 11 de maig de 2023.

[4] Pau Calvera. *VertTIRP C++*. Available at https://github.com/Pcalvera/vertTIRP_c, Consultat: 11 de maig de 2023. 2022.