Технически Университет - София

Теория на графите и приложения, 2019-2020

Семестриален проект

Цели:

Този практически насочен проект цели да затвърди уменията на студентите да работят с графи и да имплементират програми, чиято основа е обработката на графи и релевантните към тях структури данни. Въпреки че изкуствени данни генерирани на случаен принцип биха могли да бъдат използвани за целта на проекта, работата с реални данни би била по-интересна. Накратко, при дадена матрица на корелациите, програмата трябва да генерира граф с тегло, ребрата с малки тегла да бъдат премахнати, покриващо дърво да бъде намерено, и накраая, резултатите да бъдат върнати в xml формат. Студентите следва да демонстрират работещата програма, и да обяснят своя сорс код и получените резултати. По-долу не само е описан прокета, но има и някои препоръки как да бъде изпълнен добре.

Описание на входните данни:

Налични са два независими един от друг входни файла във csv формат – първият е матрица на корелациите на логаритмичната възвръщаемост (log-returns) на 25 акции на компании с голяма пазарна капитализация, а вторият съответства на матрицата на корелациите на волатилностите на същите 25 акции, пресметнати за периода 01/01/2019 до 14/12/2019. Всяка от тези квадратни матрици има по 25 реда и колони и може да бъде разглеждана като матрица на съседство на напълно свързан граф с тегло. **Нека също обърнем внимание, че едиснтвения фокус на този проект е Теорията на графите и тяхната практическа имплементация.** Как точно тези матрици са пресметнати ще бъде обяснено накратко по време на лекциите и упражненията (т.е. какво е корелация, log-returns, волатилност, дискретни времеви серии), и оценките няма да зависят от познанието по тази материя. При все това, окуражаваме всички студенти, които проявяват интерес към темата да направят свое собствено проучване.

И двета матрици са симетрични с по 25 реда и колони. Техните елементи са реални числа между -1 и 1, а всички коефициенти по диагоналите са равни на 1.0 (свойство на коефициентите на корелация). И двете матрици са достъпни в CSV (comma separated value) формат със сепаратор “,” .

Изисквания за имплементацията

1. Език за програмиране: Java или C# са препоръчителни; все пак, студентите имат възможност да изберат друг език по тяхно желание, напр. C++, Python, R, и т.н.

2. Матриците дадени като входящи данни трябва да бъдат прочетени с помощта на стандартни file reading/writing техники и библиотеки (функционалност поддържана от повечето езици). Студентите имат свободата да изберат какви типове колекции и структури данни да ползват за временно съхранение на данните, като напр. Maps/Dictionaries, Lists of Lists, two-dimensional arrays, и т.н.

3. Студентите могат да изберат как да имплементират структураата за техните графи, включваща върховете, ребрата, getters, setters, релевантни алгоритми, или също да правят пресмятанията директно върху двумерни масиви. Първата опция би била по-лесна за студенти с познания по компютърни науки и програмиране, съответно е препоръчителна. Имплементираната структура на граф трябва да бъде попълнена с данните от CSV файловете. Бележка: интерпретирайте графите като ненасочени, следователно, вместо да построявате едно ребро с начало връх *i* и край връх *j* и още едно ребро с обратна посока, просто построете едно ненасочено ребро между *i* and *j*.

4. След като напълно свързаните графи са заредени, ребрата с теглови коефициенти съответстващи на слаба корелация трябва да бъдат преманати. **За всеки връх, да се запазят само ребрата съответстващи на 3-те теглови коефициента с най-голяма абсолютна стойност.**

   Бележки:

   * Интересуваме се от силните корелации и анти-корелации (с коефициенти между 0 и -1), затова използвайте **абсолютните стойности** на тегловите коефициенти, тъй като стойности близки до 0 означават слаба статистическа взаимовръзка на върховете.

   ** за веки връх, целим да запазим свързаните с него ребра с най-голяма абсолютна стойност на теглото: така, итерирайки върху множеството от върховете, можем да съхраним всички ребра отговарящи на изискването във временна *„white-list“* структура (напр. HashMap/Dictionary съдържащ информация за върховете начало и край на всяко избрано ребро). След това, може да се направи итерация върху множеството от всички ребра в графа, и да се изтрият тези, които не са включени в описаната горе *„white-list“* структура. **Забележете, че в получения подграф на началния граф, някои върхове могат да имат повече от 3 съседа.**

5. Получените два подграфа на двата начални графа трябва да бъдат записани в xml файл (**GEXF format**, четим от софтурера Gephi) т.е. трябва да се генерират два текстови файла за всеки от под-графите, съответстващи на GEXF XSD схемата (спецификация: https://gephi.org/gexf/format/schema.html , по-долу има пример).

6. Горната програма да бъде разширена с функционалността за пресмятане на максимални покриващи дървета **(maximum spanning trees)** извлечени от описаните горе подграфи.

   Бележки:

   * Обърнете внимание, че намирането на ***максимално покриващо дърво*** е аналогично на откриването на *минимално покриащо дърво*, единствено трябва да се направи инверсия на тегловите коефициенти (или да се инвертира компаратора, в случай че ползвате такъв). Алгоритъма на Kruskal би бил добър избор в случая.

   ** За да установите дали операцията добавяне на ребро образува цикъл, припомнете си, че алгоритмите *Deep First Search* / *Breadth First Search* могат да бъдат от полза.

7. Подобно на заданието за т. 5, получените две покриващи дървета да бъдат запазени в xml файл (**GEXF format**) т.е. да се генерира текстови файл за всяко дърво, в съответствие с GEXF XSD xml схемата.

8. Накрая, отворети получените 2 + 2 XML файла с помощта на софтуера Gephi и опитайте да подобрите ръчно визуализацията на получените графи. От менюто на Gephi, изберете да експортирате графите в PNG формат.

Описание на данните на изхода:

Имплементираната програма трябва да поддържа функционалността за експортиране на генерираните графи в в GEXF формат, който описва типа граф, множеството от върховете и техните обозначения, както и ребрата, заедно с техните теглови коефициенти, ID на върха-начало и ID на върха-край. Макар, че формата GEXF може да поддържа комплексна информация като цветове и позиция на върховете, динамични вариращи във времето графи и т.н., настоящия проект не изисква използването на тези възможности. Единственатите типове информация нужна за този проект може да видите в долния пример:

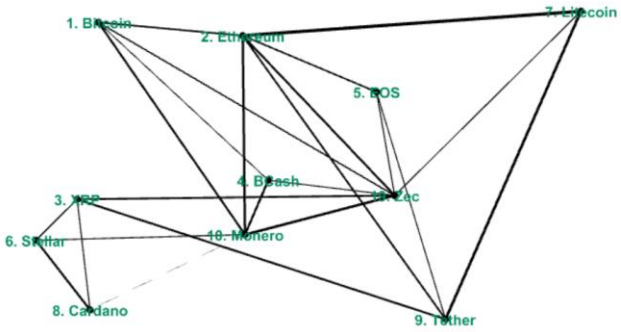
```
<?xml version="1.0" encoding="UTF-8"?>
<gexf xmlns="http://www.gexf.net/1.3" version="1.3" xmlns:viz="http://www.gexf.net/1.3/viz"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.gexf.net/1.3http://www.gexf.net/1.3/gexf.xsd">
   <graph defaultedgetype="undirected" mode="static">
      <nodes>
         <node id="0" label="Label A">
         </node>
         <node id="1" label="Label B">
         </node>
         …
         <node id="25" label="Label N">
         </node>
      </nodes>
      <edges>
         <edge id="0" source="0" target="1" weight="0.634025257"></edge>
         <edge id="1" source="0" target="6" weight="0.63573986"></edge>
         <edge id="2" source="0" target="9" weight="0.653174749"></edge>
         …
         <edge id="50" source="7" target="9" weight="0.541349961"></edge>
         <edge id="51" source="9" target="10" weight="0.737370515"></edge>
      </edges>
   </graph>
</gexf>
```

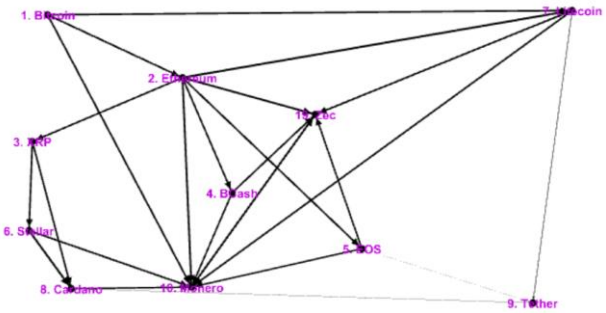
След като генерирате XML представянето на графите, уверете се, че Gephi може успешно да отвори получените файлове (в противен случай има грешка, която трябва да се коригира). Експортнете визуализираните с помощта на Gephi графи, както е описано по-горе, в PNG картинен формат.

Рапорт:

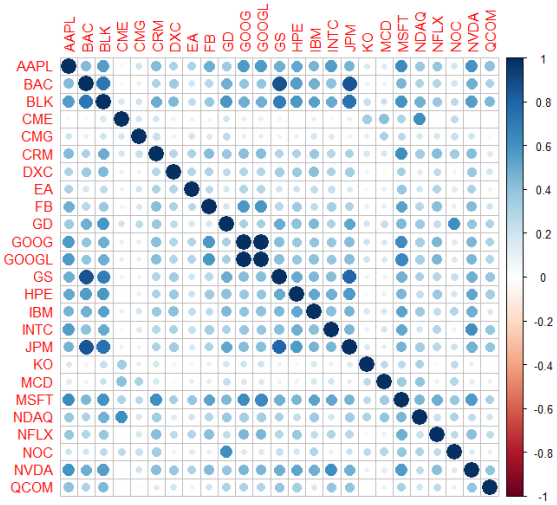
Студентите могат да работят в групи от 2 до 4 души, като всяка група трябва да изпрати рапорт (2-4 страници) включващ описание на свършената работа и получените графи в PNG формат. **Добавете описание на имплементирания сорс код.** Изпратете заедно с репорта целия сорс код; програмите, които сте написали ще бъдат изпълнени с цел да се провери тяхнта коректност. Оценките няма да зависят от времето за изпълнение, все пак опитайте да направите имплементацията си добра откъм алгоритмична сложност.

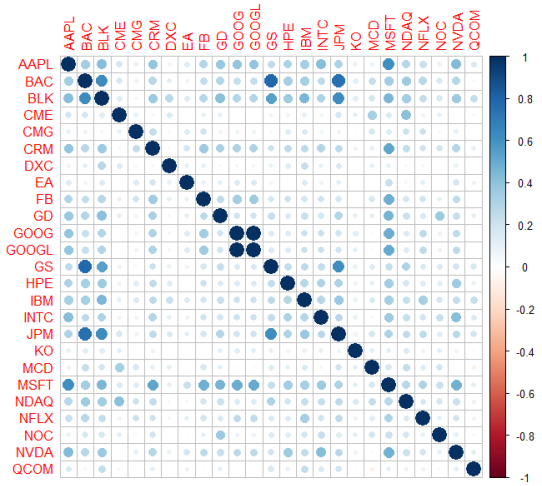
Примерни графи (генерирани върхо подобни данни за криптовалути):





Визуализация на входните данни от CSV файловете, използвани за генерирането на графите (корелации на log-returns вляво, volatility вдясно):





Списък на наименованията на колоните и редовете - съответно върховете на графите (ticker symbols):

Ticker	Company Name
AAPL	Apple Inc.
BAC	Bank of America Corp
BLK	BlackRock
CME	CME Group Inc.
CMG	Chipotle Mexican Grill
CRM	Salesforce.com
DXC	DXC Technology
EA	Electronic Arts
FB	Facebook, Inc.
GD	General Dynamics
GOOGL	Alphabet Inc Class A
GOOG	Alphabet Inc Class C
GS	Goldman Sachs Group

Ticker	Company Name
HPE	Hewlett Packard Enterprise
IBM	International Business Machines
INTC	Intel Corp.
JPM	JPMorgan Chase & Co.
KO	Coca-Cola Company
MCD	McDonald's Corp.
MSFT	Microsoft Corp.
NDAQ	Nasdaq, Inc.
NFLX	Netflix Inc.
NOC	Northrop Grumman
NVDA	Nvidia Corporation
QCOM	QUALCOMM Inc.