УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

Одабрани алгоритми теорије графова и линеарне алгебре прилагођени великим количинама података

ДОКТОРСКА ДИСЕРТАЦИЈА

Ментор:

др Ервин Варга

Кандидат:

Давор Шутић

Нови Сад, 2023. Године

УНИВЕРЗИТЕТ У НОВОМ САДУ **ОБРАЗАЦ – 5а**

НАВЕСТИ НАЗИВ ФАКУЛТЕТА ИЛИ ЦЕНТРА

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА[1]

Врста рада:	Докторска дисертација
Име и презиме аутора:	Давор Шутић
Ментор (титула, име, презиме, звање, институција)	Др Ервин Варга, ванредни професор, Факултет техничких наука, Универзитет у Новом Саду
Наслов рада:	Одабрани алгоритми теорије графова и линеарне алгебре прилагођени великим количинама података
Језик публикације (писмо):	Српски (ћирилица)
Физички опис рада:	Унети број: Страница 100 Поглавља 6 Референци 63 Табела 5 Слика 14 Графикона 0 Прилога 0
Научна област:	Електротехничко и рачунарско инжењерство
Ужа научна област (научна дисциплина):	Примењено софтверско инжењерство
Кључне речи / предметна одредница:	Рачунарство, информатика, дистрибуирани системи
Резиме на језику рада:	Предмет ове дисертације је прилагођавање комплексних алгоритама, који се користе у индустријским наменама, захтевима које диктирају потребе велике количине података (енгл. Big Data). Потребно је одређене алгоритме прилагодити дистрибуираном извршавању и великим улазним подацима на такав начин, да се пораст величине проблема може компензовати скалирањем, са акцентом на хоризонтално скалирање, тј. повећање броја дистрибуираних станица, рачунарске моћи система. Конкретно, у фокусу су следећи проблеми: 1. Проблем тока снага у електроенергетском систему 2. Проблем анализе испада у електроенергетском систему 3. Проблем спектралне спарсификације графова у општем смислу Резултати извршавања указују на прихватљива времена извршавања које постиже програмска подршка отвореног кода, која се лако скалира и подстиче даља истраживања превазилажењем тренутних ограничења. У ширем смислу, иако су проблеми наизглед везани искључиво за електроенергетске системе, у њиховој сржи се налазе општи математички проблеми који су примењиви у далеко ширем опсегу области.

[1] Аутор докторске дисертације потписао је и приложио следеће Обрасце:
5б – Изјава о ауторству;
5в – Изјава о истоветности штампане и електронске верзије и о личним подацима;
5г – Изјава о коришћењу.
Ове Изјаве се чувају на факултету у штампаном и електронском облику и не кориче се са тезом.

Датум прихватања теме од стране надлежног већа:	30.06.2022.
Датум одбране: (Попуњава одговарајућа служба)	
Чланови комисије: (титула, име, презиме, звање, институција)	Председник: др Дарко Чапко, редовни професор, Факултет техничких наука, Универзитет у Новом Саду Члан: др Игор Тартаља, ванредни професор, Електротехнички факултет, Универзитет у Београду Члан: др Имре Лендак, ванредни професор, Факултет техничких наука, Универзитет у Новом Саду Члан: др Александар Селаков, доцент, Факултет техничких наука, Универзитет у Новом Саду Ментор: др Ервин Варга, ванредни професор, Факултет техничких наука, Универзитет у Новом Саду
Напомена:	

UNIVERSITY OF NOVI SAD
FACULTY OR CENTER

KEY WORD DOCUMENTATION[2]

Document type:	Doctoral dissertation
Author:	Davor Šutić
Supervisor (title, first name, last name, position, institution)	Ervin Varga, Phd, Associate Professor, Faculty of Technical Sciences, University of Novi Sad
Thesis title:	Selected algorithms in graph theory and linear algebra adapted for Big Data
Language of text (script):	Serbian language (cyrillic script)
Physical description:	Number of: Pages 100 Chapters 6 References 63 Tables 5 Illustrations 14 Graphs 0 Appendices 0
Scientific field:	Electrical and computer engineering
Scientific subfield (scientific discipline):	Applied software engineering
Subject, Key words:	Computer engineering, Computer science, distributed systems
Abstract in English language:	The overreaching topic of this dissertation is the adaptation of complex algorithms, which are extensively used in industrial applications, to the requirements dictated by the Big Data needs. Chosen algorithms are adapted to a distributed environment and made suitable for large amounts of data in such a way, that the increasing problem size can be compensated with scaling of the computing environment. Here is the emphasis particularly on horizontal scaling, i.e. the increase of the number of distributed computing units. The following problems are considered: 1. The power flow problem in a smart grid system 2. The contingency analysis problem in a smart grid system 3. The spectral sparsification problem of graphs in the broader sense The results indicate a reasonable execution time achieved by open-source software that is easily scalable and serve to direct further research in overcoming the current limitations. In a broader sense, although these problems seem very specific in their domain of application, the underlying core mechanics contain common mathematical challenges, applicable to a broad spectrum of problems.
Accepted on Scientific Board on:	30.06.2022.

[2] The author of doctoral dissertation has signed the following Statements:
5б – Statement on the authority,
5в – Statement that the printed and e-version of doctoral dissertation are identical and about personal data,
5г – Statement on copyright licenses.
The paper and e-versions of Statements are held at he faculty and are not included into the printed thesis.

Defended: (Filled by the faculty service)	
Thesis Defend Board: (title, first name, last name, position, institution)	President: Darko Čapko, PhD, Professor, Faculty of Technical Sciences, University of Novi Sad Member: Igor Tartalja, PhD, Associate Professor, School of Electrical Engineering, University of Belgrage Member: Imre Lendak, PhD, Associate Professor, Faculty of Technical Sciences, University of Novi Sad Member: Aleksandar Selakov, PhD, Assistant Professor, Faculty of Technical Sciences, University of Novi Sad Supervisor: Ervin Varga, PhD, Associate Professor, Faculty of Technical Sciences, University of Novi Sad
Note:	

Садржај

Листа слика

Листа табела

Листа појмова

Adjacency matrix	Матрица суседности
Amazon Elastic Compute Cloud (EC2)	Инстанце виртуалних машина на Амазон платформи
Amazon Web Services (AWS)	Платформа за рачунарство у облаку фирме Амазон
Apache Hadoop	Апачи Хадуп – идејна претеча Апачи Спарка
Apache Spark	Апачи Спарк - развојно окружење које нуди паралелну обраду велике количине података која је отпорно на испаде на кластеру машина
Big Data	Количина података која је превелика за обраду на појединачној машини
Bus	Сабирница – чвор у електроенергетској мрежи
Contingency analysis	Анализа испада снага у електроенергетском систему
Flat start	„Равни старт“ – почетни услови анализе токова снага, где су одговарајуће амплитуде напона и фазни помераји постављени на један и нула
Graph conductace	Проводност графа – величина повезана с конвергенцијом насумичног хода по графу ка униформној расподели
Graph volume	Запремина графа представља збир степени сваког чвора у датом подскупу чворова.
GraphX	Апачи Спарк библиотека за анализу графова
Island detection	Одређивање острва у графу или мрежи
Jacobian	Јакобијева матрица
Laplacian matrix	Лапласова матрица
Map/Reduce paradigm	Парадигма мапирања и свођења
MLlib	Апачи Спарк библиотека за машинско учење
Newton-Raphson method	Њутн-Рафсонова метода за итеративно решавање система једначина
Power flow analysis	Анализа токова снага у електроенергетском систему
Resilient Distributed	Основна апстракција с којом Апачи Спарк ради -

Dataset (RDD)	поуздани дистрибуирани скуп података (РДД)
Smart grid	Паметни електроенергетски систем
Sparsest cut problem	Проблем „најређег“ пресека
Spectral graph sparsification	Спектрална спарсификација графова
Spectral graph theory	Спектрална теорија графова
Vertex degree	Степен чвора или валенца - број грана које се стичу у датом чвору

Сажетак

Предмет ове дисертације је прилагођавање комплексних алгоритама, који се користе у индустријским наменама, захтевима које диктирају потребе велике количине података (енгл. Big Data). Другим речима, потребно је одређене алгоритме прилагодити дистрибуираном извршавању и великим улазним подацима на такав начин, да се пораст величине проблема може компензовати скалирањем, са акцентом на хоризонтално скалирање, тј. повећање броја дистрибуираних станица, рачунарске моћи система.

Конкретно, у фокусу су следећи проблеми:

1. Проблем тока снага у електроенергетском систему
2. Проблем анализе испада у електроенергетском систему
3. Проблем спектралне спарсификације графова у општем смислу

Прва два проблема потпадају у категорију анализе паметних електроенергетских мрежа. Системи који управљају и анализирају паметне електроенергетске системе, односно мреже које представљају, се ослањају на дистрибуираност као последицу строгих захтева за безбедност, складиштење података и поузданост који се намећу управљањем критичном инфраструктуром. Значајан изазов су рачунски захтевне енергетичарске функције које се рутински користе за процену стања мреже или симулацију потенцијалних стања мреже. Иако су алгоритми познати деценијама, њихова практична примењивост је ограничена високо специјализованим математичким програмским пакетима или наменским оптимизацијама које користе индустријски програми у циљу смањења проблема пре рачунања. Од посебног значаја су функције за анализу проблема токова снага и испада у систему. А циљ је њихова имплементација у Апачи Спарк инфраструктури у дистрибуираном окружењу, базираном на рачунарству у облаци. Избор је мотивисан природом алгоритама и њиховим статусом камена темељаца анализе електроенергетских система. Наиме, прорачун токова снага се ослања на обимне и учестале матричне операције, док је анализа испада у систему фокусирана на тополошку анализу и тиме директно повезана с облашћу графова. Апачи Спарк

пружа подршку за оба концепта кроз библиотеку за машинско учење и библиотеку за обраду графова. За решавање проблема тока снага се користи Њутн-Рафсонова (енгл. Newton-Raphson) метода. Анализа испада се врши на два, међусобно упоредива, начина, који се разликују по методи одређивања повезаности мреже. Први користи анализу повезаних компоненти графа мреже, док други врши бинарно степеновање матрице повезаности мреже.

Граф је једна од најбитнијих структура података у области дистрибуираног рачунарства и алгоритама. Многи комплексни проблеми се могу представити графовима и анализирати теоријом графова. Алгоритми засновани на графовима су основа скоро свих програмских решења везаних за мреже (социјалне, електроенергетске, телекомуникационе, итд.). Само постојање одређеног алгоритма није довољно. Практичне имплементације морају да испуне мноштво нефункционалних захтева, као што су перформансе. У том тренутку напредни приступи постају неопходно, посебно ако се ради о великим графовима. Само скалирање рачунарске инфраструктуре је често недовољно, па се јавља потреба да се улазни подаци смање. Један приступ је обрада дела оригиналног графа. Циљ је остварити приближан резултат уз одговарајуће процене грешака. Спектрална спарсификација графова је један такав поступак.

Трећи проблем се бави алатом који често недостаје у алгоритмима за обраду графова, а то је спектрална спарсификација графова. Комплексност анализе графа брзо расте с његовом величином, па би било користно смањити комплексност, а одржати математичке особине графа. Уклањање чворова обично није пожељно због семантичке важности. Срећом, графови у реалној употреби теже да имају далеко више грана од чворова, па тако „прорећивање“ графа, у смислу смањења броја грана, редукује комплексност проблема, а не утиче на семантику података.

Резултати извршавања указују на прихватљива времена извршавања које постиже програмска подршка отвореног кода, која се лако скалира и подстиче даља истраживања превазилажењем тренутних ограничења.

У ширем смислу, иако су проблеми наизглед везани искључиво за електроенергетске системе, у њиховој сржи се налазе општи математички проблеми који су примењиви у далеко ширем опсегу области.

Abstract

The overreaching topic of this dissertation is the adaptation of complex algorithms, which are extensively used in industrial applications, to the requirements dictated by the Big Data needs. In other words, chosen algorithms are adapted to a distributed environment and made suitable for large amounts of data in such a way, that the increasing problem size can be compensated with scaling of the computing environment. Here is the emphasis particularly on horizontal scaling, i.e. the increase of the number of distributed computing units.

The following problems are considered:

1. The power flow problem in a smart grid system
2. The contingency analysis problem in a smart grid system
3. The spectral sparsification problem of graphs in the broader sense

The first two problems are in the smart grid system analysis category. Systems that control and analyze Smart Grids, i.e. the electrical grids they represent, rely on their distributed nature in response to the strict security, storage and responsiveness requirements dictated for operating such critical infrastructure. A major challenge are compute-intensive power functions that are routinely used to assess the state of the grid or to simulate potential states. While the algorithms are known for decades, their practical application is limited either to highly specialized mathematical software or special optimizations that high-end industrial programs introduce to reduce the size of the computation. Particularly interesting are the Power Flow and Contingency Analysis functions and the goal of developing an efficient implementation using Apache Spark in a distributed cloud-based environment. The choice is motivated by the nature of the algorithms and their status as cornerstones of any grid analysis. Namely, Power Flow relies on extensive and iterative matrix operations, while Contingency Analysis is centered on topology analysis and thus is intuitively related to graphs. Both are concepts that Apache Spark natively supports through its MlLib and GraphX libraries. The Newton-Raphson method is used for solving the power flow problem. The contingency analysis is performed with two comparable approaches. The difference lies in the method of determining the connectedness of the

network. The first approach uses the connected components analysis of the network graph, while the second uses binary multiplication of the connectedness matrix.

A graph is one of the most versatile and important data structures in the realm of distributed systems and algorithms. Many complex problems can be expressed as graph problems that could be studied through graph theory. Graph based algorithms (like, power flow analysis for distribution networks) are the cornerstone in almost all software driven solutions pertaining to networks (social, power, telecommunication, etc.). The pure existence of an algorithm is not enough, though. Practical implementations must meet myriad of non-functional requirements, such as performance. This is where advanced approaches are needed, especially when the involved graph is huge. Solely scaling the computational infrastructure is usually unsatisfactory, so there is a demand to reduce the actual input. One viable tactic is to work on a sample of the original graph. The idea is to get an approximate result with proper error estimates. Spectral graph sparsification is one such approximation scheme.

The third problem addresses an algorithm that is frequently missing in graph analysis tools: the spectral graph sparsification. The complexity of graph analysis sharply rises with its size, so it would be beneficial to in some way reduce the complexity, while keeping the mathematical properties of the graph. The removal of vertices is usually not viable, because of the semantic importance. Fortunately though, real-world graphs tend to have significantly more edges than vertices, so a sparsification of the graph, in the sense of reducing the amount of edges, reduces the complexity of the problem without affecting the semantics of the data.

The results indicate a reasonable execution time achieved by open-source software that is easily scalable and serve to direct further research in overcoming the current limitations.

In a broader sense, although these problems seem very specific in their domain of application, the underlying core mechanics contain common mathematical challenges, applicable to a broad spectrum of problems.

Увод

Паметна мрежа је широки појам за електроенергетску инфраструктуру. Потреба за ширим појмом и тренутно важећом дефиницијом [1], потиче од појаве разних паметних уређаја и све већег присуства обновљивих енергетских извора. Новитет који уводе паметни уређаји, а по којем се разликују од традиционалних, је активно учешће у раду мреже. Пример је паметни мерач, чија је сврха да периодично комуницира с надлежном компанијом и размењује телеметрију. Ово допушта предузећу да надгледа потрошњу енергије и евентуалне кварове, како би детаљно подесили одговарајућу производњу енергије. Комплементарна ситуација се дешава у случају обновљивих енергетских извора, јер је производња у тим случајевима ретко конзистентна. На пример, снага ветра се мења током времена, а самим тим и енергија коју турбина претвара. Тако је опет случај да блиска комуникација с дистрибуцијом омогућава стабилан однос производње и потрошње енергије.

Један од основних проблема у раду паметне мреже је одређивање тренутног стања система. У енергетици је овај проблем познат као проблем токова снага. То је врло нелинеаран проблем, који захтева итеративан приступ како би се дошло до довољно тачног приближног решења. Процес је врло рачунарски захтеван, како по потребној процесној моћи тако и по захтеваној слободној меморији, јер величина система диктира комплексност и димензије матрица које су потребне у сваком кораку итерације.

Друга битна функционалност за паметне мреже је могућност симулације испада. Када у делу мреже дође до квара, потребно је проценити ново стање система. Како су паметне мреже углавном комплексни системи, утицај једног испада није тривијално предвидети. Последице могу бити благе, када је квар локализован, тешке, када квар доведе до преоптерећења неког другог дела мреже или подели мрежу (дође до настанка тзв. острва), или катастрофалне, када квар изазове каскадне отказе делова система. Имајући у виду да паметне

електроенергетске мреже обухватају критичну инфраструктуру, где тешки и катастрофални кварови могу да изазову опасност по људске животе и значајно оштете опрему, анализа испада је кључни фактор безбедног управљања мрежом. Важну улогу ту игра и анализа токова снага, јер је потребно поново израчунати ново стање мреже, након што се узму у обзир настали кварови. Посебно интересантан случај је настанак острва, тј. изолованих делова мреже. Реална електроенергетска мрежа је типично изузетно густо повезана, тако да идентификација могућих острва представља нетривијалан проблем. Анализа токова снага је осетљив прорачун, те ако се примени на подељену мрежу може довести до тога да решење или дивергира или да садржи нетачне вредности. Зато је битно при анализи квара прво идентификовати све постојеће сегменте мреже и применити прорачун токова снага на сваки од њих.

Алгоритми за наведене проблеме су предмет истраживања како науци тако и у индустрији и побољшања су углавном усмерена на оптимизацију математичких алата, као што су матрична алгебра и теорија графова. Ипак, очигледан је недостатак приступа који остварују побољшања дистрибуирањем задатка. Инфраструктуре у облаку постају приступачније и „природно“ окружење за паметне електроенергетске мреже, тако да додатне рачунарске могућности не треба сматрати додатним захтевом.

Овде је представљен систем за анализу паметних електроенергетских мрежа, који подржава прорачун токова снага и анализу кварова. Прорачун токова снага се изводи коришћењем Њутн-Рафсонове методе, док се анализа кварова изводи путем два приступа. Стање повезаности мреже се одређује анализом графа конструисаног од компоненти повезаности мреже и бинарним множењем Булових матрица. Нагласак имплементације је на могућности дистрибуције рачунски захтевних операција на неколико рачунара у облаку. Апачи Спарк је изабран као инфраструктурна платформа, а Амазонове услуге у облаку као одредиште експеримената.

Приказано је Апачи Спарков потенцијал да буде примењен на другу област научног рачунарства, поред његових основних области везаних та проучавање података и машинско учење. Резултујуће решење превазилази проминентне недостатке постојећих приступа. Екстензивност и скалабилност се

обезбеђују тиме што је решење развијано у програмском језику високог нивоа и широке намене (Јава) пратећи објектно-оријентисану парадигму, како би се обезбедила лака интеграција у постојеће продукте и крива учења разумног нагиба за широк спектар корисника, од студената до инжењера и истраживача. Решење је бесплатно и отвореног кода, а исто важи и за све треће компоненте које користи. Такође, парадигме и приступи које користи су примењиви на велики број проблема који допуњују и превазилазе оквире овде представљене анализе паметних електроенергетских система.

Овде је представљен алгоритам који, користећи Апачи Спаркову библиотеку за машинско учење, решава матричне једначине у дистрибуираном окружењу.

Експериментална провера је извршена на кластерима различитих типова и променљивог броја радних станица. Коришћени су подаци реалних електроенергетских мрежа као улаз. Идентификована су уска грла извршавања и дискутовани њихови узроци, те се на тај начин истичу постојећа ограничења и постављају циљеви за будућа истраживања за њихово превазилажење.

Многи тренутни алгоритми за анализу графова су рачунарски комплексни. Пракса захтева све веће количине података, што доступне алгоритме гура до граница примењивости, јер поступци постају прескупи или временски превише захтевни. Поред тога, чињеница је да се ретко могу наћи боље апстракције проблема, тако да је тешко избегавати анализу графова, доводи до тога да се значајна истраживања врше у смеру наласка одговарајућих апроксимација графова, које су мање по обиму и комплексности, али ипак чувају карактеристике које су релевантне за тренутну анализу.

Допринос овог рада је проширење постојеће програмске подршке отвореног кода за спектралну анализу графова [1] скалабилним решењем за спектралну спарсификацију графова. У сржи решења се налази варијанта алгоритма који су развили Шпилман и Тенг [2], а који је посебно прилагођен дистрибуираном извршавању. Фокус је на коришћењу кластера како би се постигла значајна спарсификација великих реалних графова у практично достижном времену извршавања. Секундарни доприноси укључују кратку

дискусију о фином подешавању параметара алгоритма и њиховом утицају на перформансе, као и анализу перформанси самог поступка.

Решење је примењено на два велика графа са милионима грана. Како би се показала скалабилност, коришћене су две конфигурације кластера са различитим нивоима паралелизма. Добијена мерења су оцењена на основу два критеријума: време извршавања (колико је времена требало за решење проблема) и степен спарсификације (број грана у резултујућем графу о односу на број грана у изворном). Имајући у виду да делови алгоритма, конкретно одабирање графа, зависе од насумичних процеса, може доћи до варијација у резултујућем скупу грана. Такође, због неконзистентних услова извршавања у кластерском окружењу, времена извршавања се очекивано разликују при различитим извршавањима с истим параметрима. Због тога су резултати приказани с назначеном стандардном грешком да би указали на одступања у времену и броју грана.

Резултати указују на то да је показано решење у стању да произведе резултат за прихватљиво време у односу на величину улазног скупа података, да може бити оптимизовано пажљивим подешавањем улазних података, као и да је смањење броја грана у спарсификованом графу око 70%.

Преглед тренутног стања у области

Апачи Спарк [3] је од свог увођења нашао широк спектар корисника, јер представља пакет за дистрибуирану обраду података. Примењује се у разним дисциплинама, од генске анализе [4], преко специјализованих математичких метода за обраду матрица [5], то одређивања геометријских шаблона у астрономским констелацијама [6]. Укратко, погодан је за примену у већини дисциплина у којима се проблем може свести на анализу велике количине података или машинско учење. Иако постоје покушаји примене Апачи Спарка на енергетско рачунарство у облаку [7], тренутно нема алата за дистрибуирану анализу паметних електроенергетских мрежа.

Проблем токова снага је формулисан пре неколико деценија [8], ипак, од како је развијена Њутн-Рафсонова метода за решавање [9], развијен је још само један приступ [10]. Унапређења су углавном била усмерена ка побољшању математичког апарата који користе методе за решавање, углавном фокусираног на алгебру матрица.

Приступ проблему одређивања острва у електроенергетској мрежи је представљен у [11], користи матрицу повезаности мреже у комбинацији с Буловом алгебром. Ипак, у дискусији рада J. Л. Маријо и други то решење називају „непотребно комплексним“ у поређењу с приступом који користи анализу графова. Одговор аутора истиче предности њиховог решења и тврди да у њиховом искуству предложени алгоритми за претрагу графова нису бржи. Ова размена је битна, јер дефинише главне правце за истраживање идентификације острва. Они су усмерени ка побољшању матричне анализе [12], [13], или ка напреднијој тополошкој анализи [14]. Такође представља и главну мотивацију за имплементацију и поређење оба приступа у овом решењу.

Тренутно су најбитнији алати отвореног кода за анализу електроенергетских мрежа базирани на MATLAB-у [15], [16]. Од алата писаних у Java-и, истичу се DCOPFJ [17] и InterPSS [18]. Ипак, оно што свим тим алатима недостаје је механизам за дистрибуирано решавање.

Појам спектралне спарсификације графова, који је релевантан за овај рад, су увели Шпилман и Тенг [2] и односи се на спектралну сличност графа и његових спарсификатора. Њихов главни резултат је доказ да за сваки граф постоји спектрални спарсификатор скоро-линеарне величине који се може рачунарски одредити у скоро-линеарном времену. Међутим, како тврде, експоненти логаритама као и константе у добијеној комплексности су превелике да би биле од практичног значаја, али је њихов циљ био да докажу да такви спарсификатори постоје, а не оптимизација процеса добијања истих. Тако је доказана временска комплексност $O(n \log^{c} n)$, где је n број чворова у графу а константа $c \approx 30$, па иако потпада у квази-линеарну категорију, број је изузетно велик. Фокус овог рада је управо та практична димензија, тј. решење које успешно спарсификује велике графове захтевајући релативно скромне ресурсе и прихватљиво време извршавања. С тим у вези, [2] је други рад у серији од три [19] [20], чији је циљ развој ефикасних метода за решавање линеарних система једначина у симетричним, слабо дијагоналним матрицама. С аспекта спектралне теорије графова, тај поступак је важан за проналажење сопствених вредности Лапласових матрица. У овом решењу се у циљу спарсификације користе алгоритми за партиционисање графова представљени у [19]. Одређивање сопствених вредности Лапласове матрице неког графа [20] је изван обима овог рада, али представља логички наставак развоје овде представљеног приступа.

Шпилман и Тенг [2], и Шпилман и Шривастава [21] су инспирисали даља истраживања у проналаску дистрибуираног приступа. Кутис и Зу [22] уводе теоријски алгоритам за спектралну спарсификацију. Њихов рад се фокусира на употребу тежинских спанера. Међутим, одређивање спанера је комплексан поступак и захтева значајне ресурсе. Зато овај рад надограђује оригинални алгоритам [2], који и додатно доноси и интуитивнији приступ користећи теорију скупова. Нажалост, Кутис и Зу [22] нису објавили резултате мерења, па је тешко упоредити овде представљено решење с њиховим. Сун и Занети [23] приступају на сличан начин проблему спарсификације користећи кластеровање уместо спектралних метода. Њихови експерименти су усредсређени на доказ исправне функционалности њиховог алгоритма. Скупови података који они користе су за неколико редова величина мањи од оних који су овде коришћени за верификацију, тако да поређење бесмислено. Међутим, Сун

и Занети у свом раду тврде да су спектралне методе комплексне па тиме и непогодне за дистрибуирано окружење. Приступ представљен овде доказује супротно.

Апачи Спарк библиотека за обраду графова [3] [24] [25] је проширена појмом дистрибуираног спектралног графа и основним операцијама за спектралну анализу [1]. Неке од њих, нпр. одређивање Лапласове матрице, су потребне за спектралну спарсификацију. Тако је операција спектралне спарсификације логичка надоградња постојеће библиотеке за спектралну анализу.

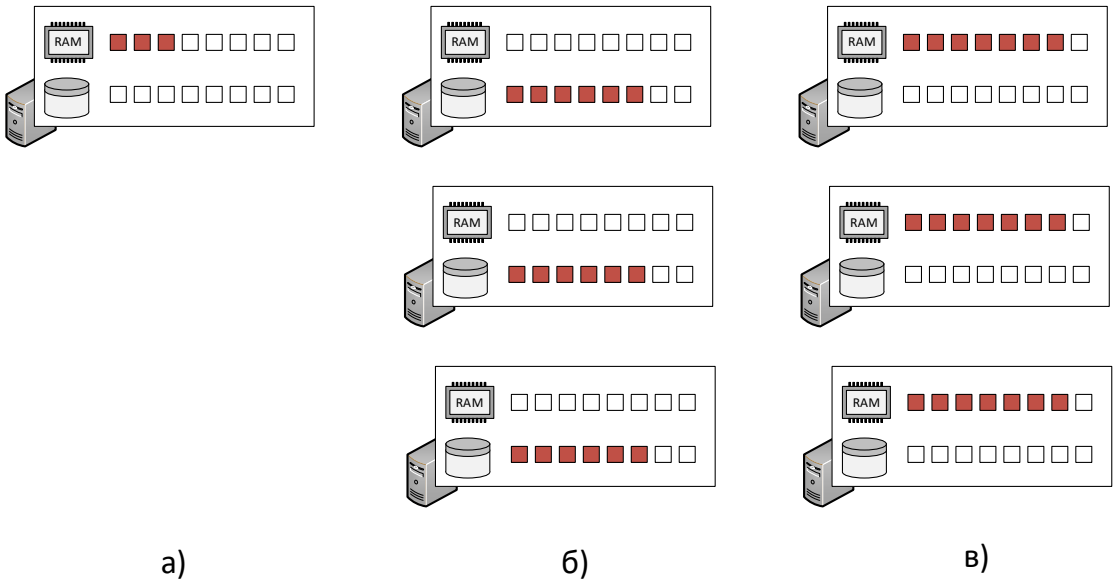
Спектрална спарсификација графова има важне примене. Смањењем броја грана у графу, а да се при томе не нарушавају његова својства, методе које су пре тога биле прескупе, постају прихватљиве и употребљиве. Посебно илустративан пример је рад који су представили Жао и остали [26]. Наиме, они предлажу нову методу, засновану на спектралној спарсификацији, за моделовање и симулацију великих електроенергетских мрежа. Они се осврћу на разне методе и закључују да ниједна није дорасла изазовима савремених електроенергетских мрежа, а да у исто време задржи и захтевану прецизност. Њихово решење користи спектралну спарсификацију да смањи граф мреже, ипак, оцењују да је тај приступ скуп за велике графове, па прибегавају партиционисању мреже, како би резултујући граф био погодан за спарсификацију.

Поставка проблема и коришћених технологија

Апачи Спарк као платформа

Апачи Спарк (енгл. Apache Spark) је сличан Апачи Хадупу (енгл. Apache Hadoop) по томе што податке чува дистрибуирано на кластеру сервера, односно чворова. Разлика је у томе што Апачи Спарк чува податке у радној меморији (RAM) док Апачи Хадуп чува податке на диску (било тврдом диску или ССД-у), као што је приказано на Слика 1.

Поред разлике у месту чувања података током обраде, корисничка спрега Апачи Спарка је и лакша за коришћење од Апачи Хадупа. Додатно, концизност Скала програмског језика (енгл. Scala) у коме је написан Апачи Спарк чини програме написане у њему краћим и лакшим за разумевање.



Слика 1 Велике количине података не стају на појединачну машину (а)). Апачи Спарк и Апачи Хадуп су технологије које дистрибуирају податке на кластер чворова. Апачи Спарк је бржи, јер дистрибуира податке користећи радну меморију машина у кластеру (в)), док Апачи Хадуп чува податке на дисковима (б)).

Дефиниција велике количине података (енлг. Big Data)

Идеја око обраде великих количина података је постала актуелна протеклих неколико година, како се потреба за тим јавила пратећи развој и доступност хардвера (процесорска моћ у виду бо) и напретка у вештачкој интелигенцији. Хардвер је постао моћнији и јефтинији у виду процесора који поседују више језгара, омогућавајући већи паралелизам у обради, и користе напредније технологије производње интегрисаних кола, између осталог и ситније производне процесе. Дискови су такође напредовали, можда најзначајније је да је цена по јединици меморије (најчешће $/GB) значајно опала, али се и технологија променила, па сад постоје насупрот ротационим дисковима и полупроводнички дискови који су далеко бржи. Модели који су релевантни за машинско учење и вештачку интелигенцију су значајно добили на димензијама, јер су алгоритми напредовали да су примењиви и на типове података који захтевају много више простора, нпр. слике, видео, звук, као и много примера за учење односно тренирање интелигенције. Тако да постоји потреба да се милијарде фотографија чува и обрађује у прихватљивом времену.

Појам велике количине података је временом добио мноштво дефиниција, али у сржи се налази кључни концепт који га дефинише: количина података која је превелика за појединачну машину. При томе треба истаћи да ограничење једне машине не важи за све димензије проблема. Дакле, могуће је да појединачна машина може да чува извесне податке, али да су они или њихова обрада сувише комплексни да би она могла и да их ефикасно обради.

Апачи Хадуп као претеча Апачи Спарка

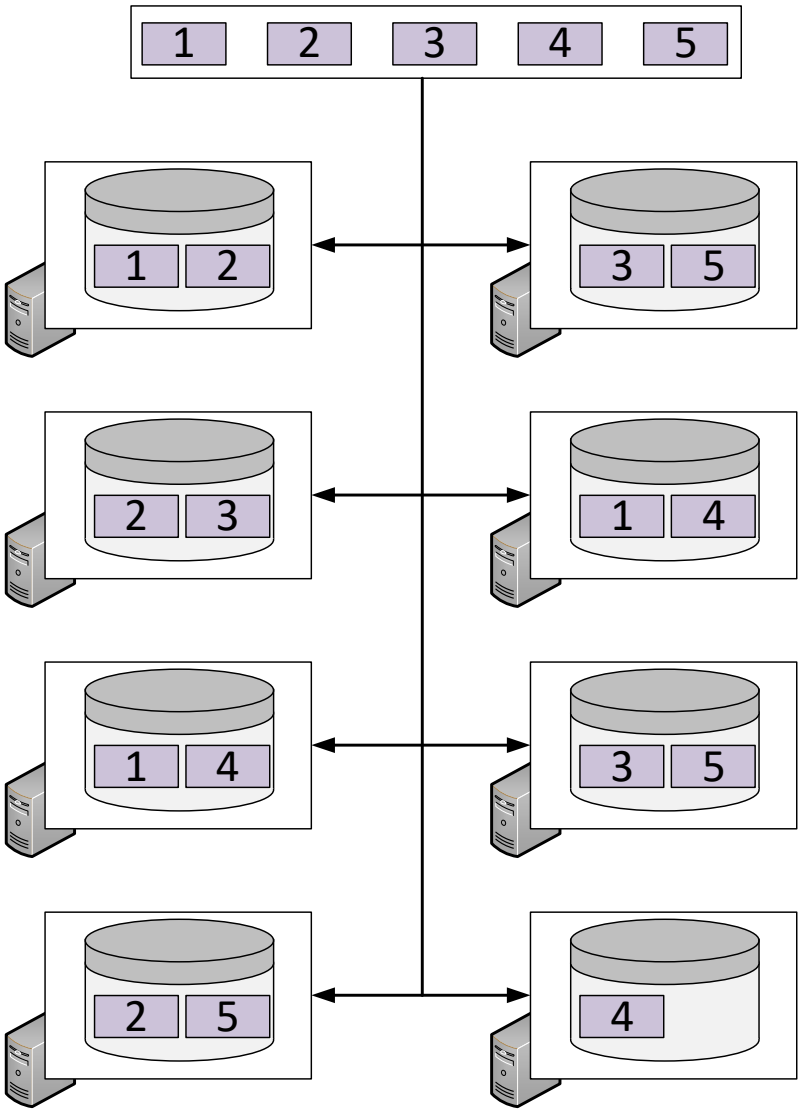
Дистрибуирани систем датотека [27] и приступ мапирања и свођења (енгл. Map/Reduce) [28] које је развила фирма Гугл (енгл. Google) су инспирисали развоја онога што данас називамо Апачи Хадуп [29]. Централни проблем за који је намењен Апачи Хадуп је процесуирање велике количине података.

Апачи Хадуп је развојно окружење које нуди паралелно процесуирање које је отпорно на испаде на кластеру машина. У ту сврху пружа две кључне компоненте:

- Апачи Хадупов дистрибуирани систем датотека (енгл. Hadoop Distributed File System - HDFS) за дистрибуирано складиштење података
- Парадигму мапирања и свођења (енгл. Map/Reduce) за дистрибуирану обраду података

Систем датотека пружа дистрибуирано складиштење отпорно на кварове. Претпоставка је да се за складиштење користи на стотине или хиљаде сервера. Имајући у виду да на толику количину машина кумулативна вероватноћа отказа постаје нетривијална, може се очекивати да ће до испада неке компоненте долазити релативно често. Како би се омогућио опоравак података, стратегија коју Апаче Хадуп имплементира је да сваку појединачну датотеку партиционише на мање блокове, чије су типичне величине између 64 MB и 128 MB. Блокови се распоређују на машине у кластеру на такав начин да се сваки блок датотеке реплицира на неколико чворова кластера (Слика 2). Подразумевана вредност је да постоје три реплике. У случају да један чвор откаже, чиме сви блокови који су се налазили на њему постају ефективно неупотребљиви, подаци нису изгубљени и преостали чворови могу транспарентно да обезбеде недостајуће блокове.

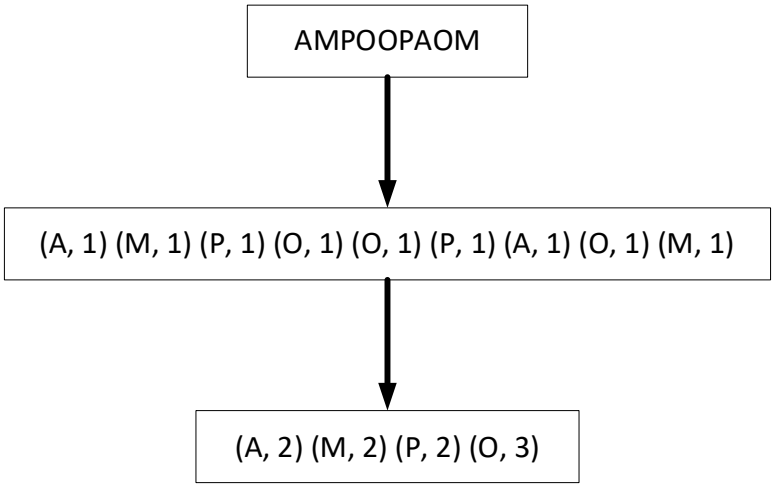
Апачи Хадупов систем датотека има и неколико архитектуралних ограничења. Није дизајниран за интерактивну употребу складиштених података. Нагласак је на великој пропусној моћи приступа подацима, а не на малом кашњењу при насумичном приступу. Апликације које га користе се типично ослањају на уређени проток података. Такође, систем је оптимизован за велике датотеке, типично неколико гигабајта или терабајта величине. Кохерентност података се обезбеђује тиме што је забрањена измена датотека. Дакле, датотека се једном уписује и могуће је вишеструко читање, али не и њена измена.



Слика 2 Блокови података распоређени с репликационим фактором 3 на Апачи Хадуп дистрибуираном систему датотека. На свакој машини у кластеру се налази део података, представљен блоковима 1-5. Блокови су распоређени тако, да су у сваком тренутку три копије доступне на кластеру. У случају отказа било које од машина, подаци су доступни и конзистентни.

Парадигма мапирања и свођења (Слика 3) је приступ који Апачи Хадуп користи за извршавање паралелне и дистрибуиране обраде. Тиме се кориснику омогућава да дефинише функције мапирања и свођења те да их примени на скуп података који се налази у Апачи Хадуповом дистрибуираном систему датотека. Функција мапирања се извршава на локалитету података, што значи да се преноси на одговарајуће чворове. Тиме се избегава премештање података преко мреже. Њена сврха је да постојеће податке неком обради претвори у други скуп података, при чему обавезно чува број изворних података, тзв. један-на-један функција. Операција сажимања се такође извршава на локалитету података, али је њена сврха агрегирање већег скупа у мањи, па се резултат враћа на неки

одређени чвор или опет дистрибуира. Паралелизам у обради се постиже приступом да „код прати податке“, па се операције извршавају на чворовима на којима су већ и подаци. Апачи Хадуп нуди и могућност поновног покретања операције, ако дође до отказа машине на којој се она извршава, што обезбеђује поузданост.



Слика 3 Парадигму мапирања и свођења користе и Апачи Хадуп и Апачи Спарк. Овде је приказана једна таква операција чији је циљ одређивања броја понављања карактера у улазној поруци. Операција мапирања трансформише сваки улазни податак у други (један-на-један мапирање). У овом конкретном случају, претвара улазни низ карактера у низ парова, где је први елемент знак који служи као кључ, а други бројач понављања иницијализован на 1. Операција свођења у овом случају сабира други елемент у пару по кључу и производи низ код којег је сваком знаку придружен његов број понављања. Овде је свођење операција типа више-на-више, али у општем случају може бити и више-на-један, дакле да излаз буде јединствена вредности.

У случају Апачи Хадупа, приступ мапирања и свођења дозвољава обраду низа парова кључ-вредност и њихово поновно уписивање у систем датотека. Па се овај приступ користи за решавање многих проблема низањем операција типа читање-обрада-упис. Ипак, испоставио се као примењив на једноставније задатке, али не и за итеративне алгоритме, као што је машинско учење. То је домен где Апачи Спарк уводи значајна побољшања.

Апачи Спарк

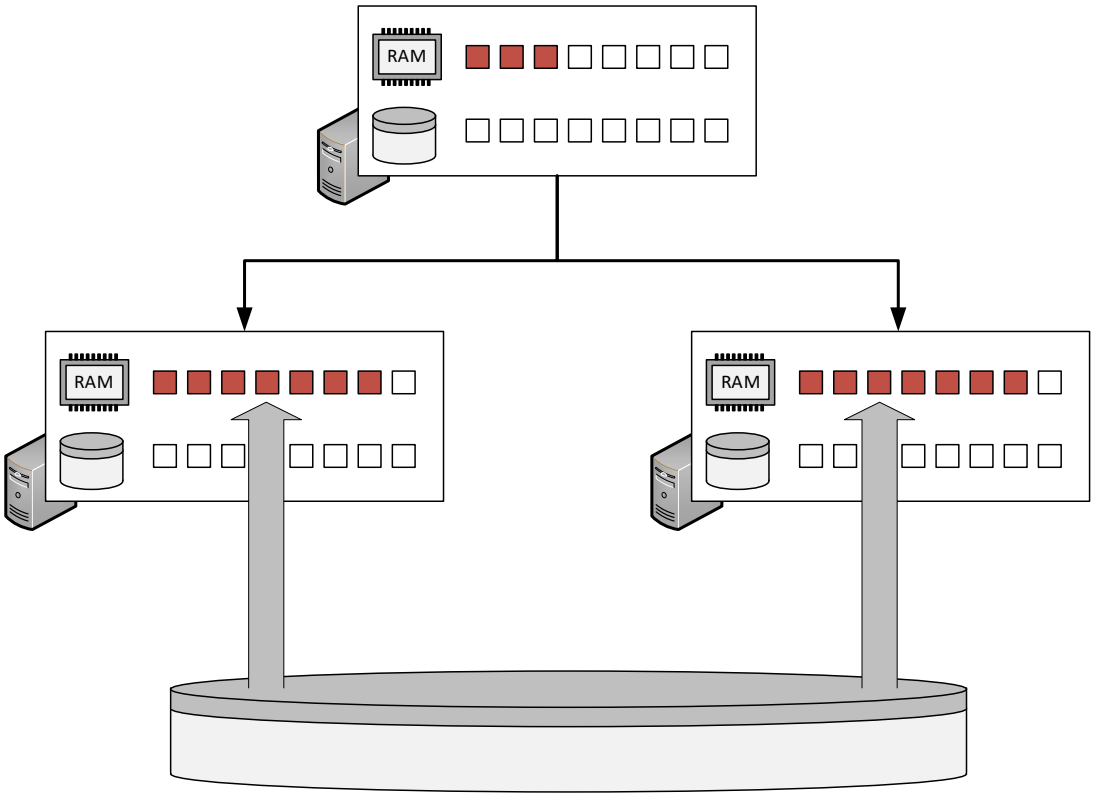
Апачи Хадуп се не сналази најбоље с интерактивним упитима и итеративним алгоритмима. Интерактивни упити подразумевају да се над истим

скупом података изврши више упита. Апачи Хадуп је у стању да врати резултат једног упита, а за сваки следећи мора да понови целокупну процедуру учитавања података с диска и њиховог поновне обраде. Тако да се неретко дешава да се понављају кораци већ урађене анализе као претече предстојеће обраде. Карактеристично за итеративне алгоритме је да извршавају неки скуп инструкција над датим скупом података док се не испуни критеријум заустављања. Имају широку примену, од машинског учења до обраде графова. Имплементација таквих алгоритама помоћу Апачи Хадупа типично захтева низ операција мапирања и свођења при чему се при свакој итерацији подаци изнова учитавају с диска. Такав прорачун може неприхватљиво дуго да траје, посебно ако се узму у обзир велики скупови података и поступци с хиљадама итерација.

Апачи Спарк се извршава на кластеру машина, слично као Апачи Хадуп. Основна апстракција с којом Апачи Спарк ради је тзв. поуздани дистрибуирани скуп података - РДД (енгл. Resilient Distributed Dataset - RDD) (Слика 4). РДД-еви су дистрибуиране структуре података, отпорне на испаде, које дозвољавају корисницима перзистенцију међуподатака у меморији, контролу партиционисања у циљу оптимизације расподеле података и манипулацију података помоћу скупа оператора.

Формално, РДД представља партиционисан скуп записа, над којим је дозвољено само читање, тј. непромењив је. Детерминистички се прави искључиво из две извора: подаци у стабилном складишту и други РДД. Операције које као резултат враћају РДД се називају трансформацијама, а поменута операција мапирања је један такав пример.

Насупрот трансформацијама, постоје и операције које се називају акцијама. Њихова карактеристика је да као резултат извршавања враћају вредност апликацији или уписују податке у складиште података. Пример акције је поменута операција свођења.



Слика 4 Апачи Спарк нуди РДД апстракцију, која се може посматрати као дистрибуирани низ у радној меморији. Управљачка машина чува Апачи Спарк програм и контролише рад радних машина. Радне машине учитавају велику количину података из неког дистрибуираног складишта и смештају их у своје радне меморије. Над тим скупом се извршавају инструкције које стрижу с управљачке машине.

Не постоји потреба да РДД-еви буду инстанцирани у сваком тренутку времена. Сваки РДД има довољно информација о томе како је изведен из других података, тј. његово „порекло“, да може у сваком тренутку да израчуна садржаје својих партиција почевши од извора података са стабилног складишта. Ово практично повлачи за собом да ниједан програм не може да дође у ситуацију да референцира РДД који не може да реконструише након настанка неког квара – РДД је структура отпорна на кварове.

Отпорност на кварове доноси једну битну импликацију. Речено је да трансформације производе нове РДД-еве и да на тај начин сваки од њих има јединствен и поновљив низ операција које су довеле до његовог настанка. То омогућава да се резултати трансформација не извршавају одмах при задавању, већ да се након задате акције изврши целокупно стабло инструкција, почевши од података у складишту или неког перзистираног РДД-а па до извршавање дате акције. Може се рећи да су различити РДД-еви чворови, трансформације гране,

а акције листови стабла извршавања. Простије речено, ниједна трансформација се сама по себи не извршава при задавању, него тек извршавање неке акције резултује и извршавањем свих трансформација над извесним скупом података које су јој претходиле. На овај начин се пружа прилика за оптимизацију. Апачи Спарк поседује посебну компоненту, организатор извршавања (енгл. Scheduler), која пре извршавања анализира стабло и покушава да оптимизује операције у њему.

Корисници могу да управљају двома аспектима РДД-а: перзистенцијом и партиционисањем. Тако могу да предвиде које РДД-ове ће поново користити и изабрати одговарајућу стратегију чувања тих података, нпр. чување у радној меморији. Такође, могу да одреде да се елементи РДД-а расподеле на машине на основу кључа у сваком запису. Ово је корисно за оптимизацију позиције, нпр. обезбедити да се два скупа податка који ће бити спојени у један налазе близу један другом.

Развој Апачи Спарк апликације подразумева писање програма који дефинише РДД-еве и позива операције над њима. Окружење се састоји од једне управљачке машине и више радних машина. Програм се извршава на управљачкој машини, која управља радним машинама да оне извршавају операције над РДД-евима у њиховој надлежности на одговарајући начин.

Постоје два типа зависности РДД-ева. Уске зависности код којих је карактеристични да је свака родитељска РДД партиција потребна највише једној РДД партицији детету и широке зависности где више РДД партиција деце може да зависи од ње. Уске зависности, као што је операција мапирања, дозвољавају извршавање на једном чвору кластера, а широке зависности, као што је операција спајања два РДД-а, зависе од свих родитељских партиција и захтевају прерасподелу података (енгл. shuffling). Већ поменути организатор извршавања покушава да групише што више трансформација уских зависности у један ступањ извршавања. Граница између ступњева извршавања је операција прерасподеле података. Када стигне ред на извршавање акције, организатор направи усмерени ациклични граф ступњева извршавања, који се онда извршавају од почетне тачке.

Апачи Спарк подржава две механизма комуникације вредности ка радним машинама који се могу користити при извршавању РДД операција. Програм на управљачкој машини може директно пренети (енгл. broadcast) вредност неке локалне променљиве свим радним машинама као непроменљиву копију. Постоји и операција прикупљања (енгл. collect) над неким РДД-ом. Она ће пренети целокупни РДД и сместити га на управљачку машину. Наравно, овде је неопходно обратити пажњу да је такав РДД довољно компактан да може да стане на једну машину. Ово су начини на које управљачка машина, односно програм на њој, може директно да утиче на извршавање и представљају комуникациони механизам. Поред тога је могуће и спојити два или више РДД-ова, тако што неко поље у њиховим записима служи као кључ. Спајање РДД-ова размењује информације директно између радних машина, без интервенције управљачке машине. Поступак спаја записе РДД-ева који се налазе на различитим машинама па их поново партиционише у нови РДД, све са циљем да се информације које су биле на различитим машинама нађу на истој.

Апачи Спарк већину својих операција извршава у радној меморији. То му даје додатне предности у односу на Апачи Хадуп, првенствено због брзине приступа, али и због начина приступања подацима. Радна меморија подржава и насумични приступ, што Апачи Хадупу није јача страна, као што је већ речено.

Оно што омогућава Апачи Спарку да се прилагоди интерактивним упитима и итеративном процесуирању је његова могућност да кешира РДД-еве у меморији. Тако се заобилази потреба да се изнова обрађује ланац РДД-ева сваки пута када се врати неки резултат. Ово такође имплицира да за ефективно извршавање Апачи Спарк програма, машине у кластеру треба да имају довољно радне меморије. Међутим, чак и да понестане радне меморије, Апачи Спарк ће почети аутоматски да користи и доступне дискове. То ће наравно резултовати у деградираним перформансама, али неће довести до прекида извршавања.

Кластер машина који извршава Апачи Спарк програме мора да има дистрибуиран простор предвиђен за трајно складиштење података. Неки примери су већ поменути Апачи Хадуп дистрибуирани систем датотека [29], Касандра (енгл. Cassandra) [30], Амазонов С3 [31].

Проблем токова снага

Анализа токова снага је фундаментални алат у електроенергетском инжењерингу који се користи за израчунавање протока електричне снаге кроз систем. Ово поглавље из електроенергетских импликација ствара математички модел, погодан за имплементацију решења.

Предмет проблема токова снага је балансирање потребног оптерећења и повезаних губитака с продукционим капацитетима генератора у електроенергетској мрежи. Електроенергетска мрежа се састоји од генератора, преносних линија, трансформатора, сабирница и потрошача. Па је по формалнијој дефиницији проблема токова снага циљ је израчунати комплексну вредност напона и снаге у свакој сабирници [32]. Ако је позната та информација, могуће је по потреби израчунати све друге параметре електроенергетске мреже. Ова тврдња произилази из чињенице да је познавање једне комплексне карактеристике свих чворова у систему, довољно за реконструкција целокупног режима рада система. Другим речима, амплитуда и фазни углови напона у свакој сабирници чине стационарни режим система.

Стање чвора је одређено са четири промењљиве: амплитуда напона (V), фазни померај напона (θ), активна (P) и реактивна (Q) снага, са још неколико мање битних параметара.

Стање система с N чворова је дефинисано са N једначина, од којих је i-та:

$$\widehat{S_i} = P_i - jQ_i = \widehat{V_i}\widehat{I_i^*} = \widehat{V_i}\sum_{j=1}^{N}\widehat{Y_{ij}^*}\widehat{V_j^*} \tag{1}$$

Где је $\widehat{S_i}$ комплексна снага, при чему је P_i активна снага, а Q_i реактивна, гране и, $\widehat{I_i}$ и $\widehat{V_i}$ су комплексне вредности напона и струје, респективно, а $\widehat{Y_{ij}}$ представља адмитансу између чворова i и j. Треба приметити да је једначина (1) изведена из

основне формуле за комплексну снагу и трансформисана, у складу с Омовим и Кирхофовим законима, тако да струја не фигурише.

Адмитанса се сматра константном у сваком прорачуну токова снага. Представља се квадратном матрицом величине н за систем од н чворова. Елементи ван главне дијагонале дефинишу нодалну адмитансу између чворова одређеним индексима колона и редова. Елементи на главној дијагонали представљају тзв. само-адмитансе. Ако чворови i и j нису повезани, елемент $\widehat{Y_{ij}} = \widehat{Y_{ji}} = 0$. Ово указује на то да су матрице адмитанси ретке, за већину реалних електроенергетских мрежа.

Раздвајање адмитансе на комплексне чиниоце, тј. кондуктансу (G) и сусцептансу (B), побољшава читљивост и има друге техничке предности, о којима се дискутује касније.

$$\widehat{Y_{ij}} = G_{ij} + jB_{ij} \tag{2}$$

Следеће релације важе између кондуктансе и сусцептансе, и резистансе (R) и реактансе (X):

$$G(i,j) = \frac{R_{ij}}{R_{ij}^2 + X_{ij}^2} \tag{3}$$

$$B(i,j) = -\frac{X_{ij}}{R_{ij}^2 + X_{ij}^2}. \tag{4}$$

Вредност једног елемента је одређена присуством, односно одсуством, трансформатора на било којем чвору, као и његовом позицијом, тј. да ли је на главној дијагонали или ван ње. На тај начин се дефинишу следеће једначине када се рачуна адмитанса између чворова i и j.

Ако нема трансформатора, елементи ван главне дијагонале су симетрични:

$$G_{ij} = G_{ji} = -G(i,j) \tag{5}$$

$$B_{ij} = B_{ji} = -B(i,j) \tag{6}$$

Елементи на главној дијагонали су представљени збиром свих грана које се стичу у задатом чвору и узимају у обзир линијске сусцептансе (B_{line}) релевантних грана.

$$G_{ii} = \sum_{\substack{k=0 \\ k \neq i}}^{N} G(i,k) \tag{7}$$

$$B_{ii} = \sum_{\substack{k=0 \\ k \neq i}}^{N} \left(B(i,k) + \frac{B_{line}}{2} \right). \tag{8}$$

Када је трансформатор присутан, елементи ван главне дијагонале нису симетрични. Утицај трансформатора је описан његовим односом извода (τ) и углом помераја фазе (θ_{shift}). Како карактеристике трансформатора нису симетричне, једначине подразумевају да је примар повезан на и чвор.

$$G_{ij} = -\frac{1}{\tau}\left(G(i,j)\cos\theta_{shift} - B(i,j)\sin\theta_{shift} \right) \tag{9}$$

$$G_{ji} = -\frac{1}{\tau}\left(G(i,j)\cos\theta_{shift} + B(i,j)\sin\theta_{shift} \right) \tag{10}$$

$$B_{ij} = -\frac{1}{\tau}\left(G(i,j)\sin\theta_{shift} + B(i,j)\cos\theta_{shift} \right) \tag{11}$$

$$B_{ji} = -\frac{1}{\tau}\left(B(i,j)\cos\theta_{shift} - G(i,j)\sin\theta_{shift} \right). \tag{12}$$

Исти принцип се примењује на елементе главне дијагонале, представљен једначинама (7) и (8), међутим због асиметрије, мењају се изрази сабирака који се односе не везу између чворова i и j где постоји трансформатор. Ти изрази следе.

$$G_{ii} = \frac{G(i,j)}{\tau^2} \tag{13}$$

$$G_{jj} = G(i,j) \tag{14}$$

$$B_{ii} = \frac{1}{\tau^2}\left(B(i,j) + \frac{B_{line}}{2} \right) \tag{15}$$

$$B_{jj} = B(i,j) + \frac{B_{line}}{2}. \tag{16}$$

Параметри R, X, B_{line}, τ и θ_{shift} су познати за сваку грану мреже и чине део улазних података за прорачун токова снага. Користећи наведене једначине, матрица адмитанси се може генерисати у сваком тренутку.

Чворови, тј. сабирнице, се деле на три типа:

1. Потрошачки чвор: Познат и као PQ сабирница. Овде су познате активне и реактивне снаге, док напони и фазни ставови нису. Потрошачи су типично везани за ове сабирнице.
2. Генераторски чвор: Познат и као PV сабирница. Овде су познати активна снага и амплитуда напона, док фазни став и реактивна снага нису. Генератори су типично везани за ове сабирнице.
3. Балансни чвор: Познат и као референтни чвор. Ово је наменски, „специјални“, генераторски чвор који се бира за сваки прорачун токова снага. Активна и реактивна снага су остављене неодређене, како би се балансирали евентуални губици у систему. Такође, служи као референтни угао за све друге помераје у систему. Фазни став је типично постављен на 0. Поред фазног става, позната је и амплитуда напона. Постојање овог типа чвора је више математичка потреба, него физичка представа реалности као у претходна два случаја.

Типови чворова су познати, или изабрани у случају референтног чвора, и играју значајну улогу у поставци једначина. Битно је и истаћи да сваки тип чвора дефинише две познате и две непознате променљиве.

Њутн-Рафсонов метод је добро познат приступ за решавање система нелинеарних једначина. Начин рада је да се нелинеарни проблем, као што је систем од N једначина проблема токова снага (1), апроксимира у линеарну матричну једначину и њу итеративним приступом реши.

Ово се постиже тако што се комплексна једначина (1) подели у две реалне:

$$P_i = V_i \sum_{j=1}^{N} V_j\left(G_{ij}\, cos\left(\theta_i - \theta_j\right) + B_{ij}\, sin\left(\theta_i - \theta_j\right)\right) \tag{17}$$

$$Q_i = V_i \sum_{j=1}^{N} V_j\left(G_{ij}\, sin\left(\theta_i - \theta_j\right) - B_{ij}\, cos\left(\theta_i - \theta_j\right)\right) \tag{18}$$

Сада се тзв. губици активне и реактивне снаге могу дефинисати на следећи начин:

$$\Delta P_i = P_i^t - P_i \tag{19}$$

$$\Delta Q_i = Q_i^t - Q_i, \tag{20}$$

где су P_i^t и Q_i^t укупна активна и реактивна снага и-тог чвора, тј. разлика произведене и потрошене активне и реактивне снаге на том чвору.

Ако се амплитуде напона и фазни помераји свих чворова у систему посматрају као вектори променљивих, (17) и (18) се могу искористити за формирање Јакобијеве матрице која је потребна за Њутн-Рафсонову методу. Како изрази с десне стране једначина (17) и (18) садрже две непознате (V и θ), Јакобијева матрица се конструише користећи парцијалне изводе првог реда за P_i и Q_i по непознатим променљивим. Тако да основи проблем за анализу токова снага система од N чворова је решавање следеће матричне једначине:

$$\begin{bmatrix} \Delta P^h{}_1 \\ \Delta Q^h{}_1 \\ \vdots \\ \Delta P^h{}_N \\ \Delta Q^h{}_N \end{bmatrix} = \begin{bmatrix} \frac{\partial P^h{}_1}{\partial \theta^h{}_1} & \frac{\partial P^h{}_1}{\partial V^h{}_1} & \cdots & \frac{\partial P^h{}_1}{\partial \theta^h{}_N} & \frac{\partial P^h{}_1}{\partial V^h{}_N} \\ \frac{\partial Q^h{}_1}{\partial \theta^h{}_1} & \frac{\partial Q^h{}_1}{\partial V^h{}_1} & \cdots & \frac{\partial Q^h{}_1}{\partial \theta^h{}_N} & \frac{\partial Q^h{}_1}{\partial V^h{}_N} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial P^h{}_N}{\partial \theta^h{}_1} & \frac{\partial P^h{}_N}{\partial V^h{}_1} & \cdots & \frac{\partial P^h{}_N}{\partial \theta^h{}_N} & \frac{\partial P^h{}_N}{\partial V^h{}_N} \\ \frac{\partial Q^h{}_N}{\partial \theta^h{}_1} & \frac{\partial Q^h{}_N}{\partial V^h{}_1} & \cdots & \frac{\partial Q^h{}_N}{\partial \theta^h{}_N} & \frac{\partial Q^h{}_N}{\partial V^h{}_N} \end{bmatrix} \begin{bmatrix} \Delta \theta^h{}_1 \\ \Delta V^h{}_1 \\ \vdots \\ \Delta \theta^h{}_N \\ \Delta V^h{}_N \end{bmatrix} \tag{21}$$

или, у компактнијем матричном облику:

$$\Delta \boldsymbol{S}^h = \boldsymbol{J}^h \cdot \Delta \boldsymbol{X}^h. \tag{22}$$

Показано је [33], да се парцијални изводи који чине елементе ван главне дијагонале Јакобијеве матрице ($\boldsymbol{J}^h$) могу одредити користећи следећи скуп једначина:

$$H_{ik}^h = \frac{\partial P_i^h}{\partial \theta_k^h} = V_i{}^h V_k{}^h\left(G_{ik}\, sin\left(\theta_i{}^h - \theta_k{}^h\right) - B_{ik}\, cos\left(\theta_i{}^h - \theta_k{}^h\right)\right) \tag{23}$$

$$N_{ik}^{h} = \frac{\partial P_i^h}{\partial V_k^h} = {V_i}^h {V_k}^h \left(G_{ik}\, cos\left({\theta_i}^h - {\theta_k}^h \right) + B_{ik}\, sin\left({\theta_i}^h - {\theta_k}^h \right) \right) \tag{24}$$

$$J_{ik}^{h} = \frac{\partial Q_i^h}{\partial \theta_k^h} = -{V_i}^h {V_k}^h \left(G_{ik}\, cos\left({\theta_i}^h - {\theta_k}^h \right) + B_{ik}\, sin\left({\theta_i}^h - {\theta_k}^h \right) \right) \tag{25}$$

$$L_{ik}^{h} = \frac{\partial Q_i^h}{\partial V_k^h} = {V_i}^h {V_k}^h \left(G_{ik}\, sin\left({\theta_i}^h - {\theta_k}^h \right) - B_{ik}\, cos\left({\theta_i}^h - {\theta_k}^h \right) \right) \tag{26}$$

док се елементи на главној дијагонали добијају на другачији начин:

$$H_{ii}^{h} = \frac{\partial P_i^h}{\partial \theta_i^h} = -B_{ii} {V^2}_i^h - Q_i \tag{27}$$

$$N_{ii}^{h} = \frac{\partial P_i^h}{\partial V_i^h} = G_{ii} {V^2}_i^h + P_i \tag{28}$$

$$J_{ii}^{h} = \frac{\partial Q_i^h}{\partial \theta_i^h} = -G_{ii} {V^2}_i^h + P_i \tag{29}$$

$$L_{ii}^{h} = \frac{\partial Q_i^h}{\partial V_i^h} = -B_{ii} {V^2}_i^h + Q_i. \tag{30}$$

Искази (23) до (26), а последично и (22), у тренутном облику претпостављају да сваки чвор у мрежи уводи две једначине, (17) и (18). Међутим, то не мора бити случај, зависно од типа сваког чвора. Имајући у виду да су промењљиве V и θ фиксиране у случају референтног чвора, тај чвор не уводи нове једначине у систем. По сличном принципу су за генераторске чворове (PV) познати активна снага и амплитуда напона, па се у систем уноси само једначина (17). Тако остаје потрошачки чвор (PQ) као једини који уводи једначине (17) и (18) у систем, које представљају две једначине с две непознате. Практично, то се на (22) одражава на следећи начин:

1. За сваки референтни чвор, нека он буде означен као и-ти чвор, ${\Delta P^h}_i$ и ${\Delta Q^h}_i$ се уклањају из $\Delta \boldsymbol{S}^h$, такође ${\Delta \theta^h}_i$ и ${\Delta V^h}_i$ се уклањају из $\Delta \boldsymbol{X}^h$, док се из Јакобијеве матрице $\boldsymbol{J}^h$ бришу и-ти и (i+1)-ти ред и колона, тј. редови и колоне који се односе на ${P^h}_i$, ${Q^h}_i$, ${\theta^h}_i$ или ${V^h}_i$.

2. За сваки генераторски чвор, нека он буде означен као и-ти чвор, $\Delta Q^h{}_i$ се уклања из $\Delta \boldsymbol{S}^h$, такође се $\Delta V^h{}_i$ уклања из $\Delta \boldsymbol{X}^h$, док се из Јакобијеве матрице $\boldsymbol{J}^h$ бришу (i+1)-ти ред и колона, тј. редови и колоне који се односе на $Q^h{}_i$ или $V^h{}_i$.

Тако је број једначина које описују систем од N чворова дат изразом:

$$n_{eq} = 2N - 2n_{slack} - n_{PV}. \tag{31}$$

Као што је већ напоменуто, Њутн-Рафсонова метода је итеративни поступак и сваком итерацијом производи бољу апроксимацију решења. Једначина (21) показује h-ту итерацију. Како су вектори V и θ једине непознате у систему, циљ сваке итерације је да нађе приближније вредности ових променљивих. Таква апроксимација служи као улазни податак за следећу итерацију, тј. да се израчуна $\Delta \boldsymbol{X}^h$. Резултујуће вредности ΔV и $\Delta \theta$ повезују текућу итерацију с будућом на следећи начин:

$$\Delta V^h = V^{h+1} - V^h \tag{32}$$

$$\Delta \theta^h = \theta^{h+1} - \theta^h \tag{33}$$

где V^{h+1} и θ^{h+1} представљају вредности следеће итерације. Решење се сматра довољно тачним, односно конвергира, ако испуњава критеријум конвергенције, који обично захтева да је разлика вредности добијених у текућој и следећој итерацији довољно мала:

$$|V^{h+1} - V^h| \leq \varepsilon_V \tag{33}$$

$$|\theta^{h+1} - \theta^h| \leq \varepsilon_\theta. \tag{34}$$

Имајући у виду да је систем једначина, дат изразима (17) и (18), нелинеаран и да је сврха Њутн-Рафсонове метода да апроксимира такав систем у линеаран систем како би га решила, постоје разне ситуације у којима Њутн-Рафсонова метода дивергира. То се може догодити ако проналажење решења траје предуго, или је немогуће, што је често случај када решења добијена из итерације осцилују на такав начин да критеријум конвергенције није никада испуњен. Друга могућност је да је решење израчунато, али просто погрешно, тј. да је поступак конвергирао ка погрешном корену. Таква и слична разматрања указују колико је битно пажљиво размотрити улове извршавања Њутн-

Рафсонове методе. Очекује се да прорачун релативно брзо конвергира, па постављање максималног броја итерација спречава бесконачне петље. Како би се обезбедила околина одговарајућег корена једначина (17) и (18), потребно је спречити да решења две узастопне итерације значајно одступају једно од другог. То се ради применом граничног услова на добијена решења итерације, који обезбеђује да апсолутна вредност сваког ΔV и $\Delta\theta$, добијених из тренутне итерације, није већа од предефинисаних вредности за амплитуду напона и фазни померај. Напослетку, избор почетних вредности је изузетно битан. Једначине (17) и (18) су периодичне, па теоријски могу имати бесконачан број решења. Ипак, само једно решење је исправно за сваку електроенергетску мрежу. Имајући то у виду, у случају да је познато решење неког претходног прорачуна, оно представља добру полазну тачку за сличне услове у мрежи. У случају да претходна решења нису доступна, као што је случај с првом анализом електроенергетске мреже, иницијална конфигурација тзв. равног старта је боља од насумичне. Она подразумева да су на почетку све амплитуде напона постављене на 1, а сви фазни помераји на 0.

Проблем анализа квара

Као и за проблем токова снага у претходном делу, овде се износе изазови везани за анализу квара у електроенергетској мрежи и свести их на математички модел, погодан за имплементацију.

Поступак анализе квара је важан симулациони алат, јер је његова сврха процена утицаја потенцијалних испада на стање електроенергетске мреже.

Када дође до испада, може доћи до следећих последица, у растућем поретку озбиљности:

1. испад се не „осећа“ у систему, тј. не доводи до настанка острва у мрежи нити ствара изузетне услове, као што је преоптерећење, ни на једном елементу мреже

2. испад доводи до распадања мреже на мање мреже, стварајући острва, али су она стабилна и самоодржива, па се не јављају изузетне околности
3. испад не доводи до настанка острва, али поставља известан број елемената мреже у стање аларма, квар је локализован и не долази до каскадног отказивања
4. испад резултује острвима која нису стабилна, нпр. у некима од њих нема генератора, па су и локални елементи одсечени, а откази опреме, ако постоје, су локализовани
5. испад доводи до каскадног отказивања мреже, узрокујући постепено да сваки елемент мреже откаже.

Анализа квара се изводи у два корака. Прво се одређује повезаност мреже. Друго, ако је констатован настанак острва, стање сваког од њих се оцењује прорачуном токова снага појединачног острва и проверавају се нерегуларности.

Анализа повезаности мреже одређује да ли су сви чворови међусобно повезани, макар и индиректно. Овај проблем се може свести на проблем компоненти повезаности, који је чест изазов у теорији графова. Наиме, ако се сабирнице посматрају као чворови, а гране мреже као гране графа, задатак је да се одреде сви подграфови који се састоје од међусобно повезаних чворова. У случају да је мрежа повезана, постоји тачно један такав подграф који је идентичан графу мреже. Ако постоје острва, овај метод ће произвести два или више подграфова од којих сваки одговара по једном острву у систему.

Други метод је анализа матрице [11], који се може сматрати и „класичним“ приступом. Идеја је да се генерише Булова матрица повезаности где су елементи на индексима i и j постављени на 1, ако су чворови i и j повезани, а у супротном случају су 0. Матрица ја симетрична, па се исте вредности налазе на индексима i и j, као и на j и i. Елементи главне дијагонале су увек постављени на 1. Ово је директна аналогија с претходно представљеној матрицом адмитанси, са изузетком да елементи имају вредности 0 и 1 уместо вредности адмитанси.

Бинарно множење матрице повезаности са самом собом даје матрицу повезаности другог степена. Ако је неки елемент с индексом i и j био 0 у матрици првог степена, али је променио вредност у 1 у матрици другог степена, то значи да су чворови i и j индиректно повезани преко једног чвора између. Због симетрије, исто важи и за чворове j и i. Даљим бинарним множењем с резултујућом матрицом се добијају матрице повезаности виших степена, од којих свака идентификује дубље повезаности између чворова. Последица тога је, да се у систему од N чворова, алгоритам зауставља након (N-2) итерације. Ако иницијална матрица која је бинарним множењем подигнута на (N-2) степен и даље садржи елементе који су 0, ти чворови имају међусобно одстојање веће од N. Према томе, тај скуп елемената указује на компоненте које нису повезане с главном мрежом. Потпуно повезана мрежа производи матрицу повезаности у којој су сви елементи постављени на 1.

Један критеријум заустављања је провера да ли (N-2) степен иницијалне матрице повезаности садржи елементе једнаке 0. Ипак, критеријум се може опустити. Мрежа је потпуно повезана, ако било који степен бинарног множења произведе матрицу која садржи искључиво вредности 1. Тада алгоритам може да се заустави, јер даље множење неће променити исход. Према томе је општији услов заустављања да два узастопна множења не мењају резултујућу матрицу. То имплицира да не може бити дубљих повезаности, јер нису откривене нове везе између чворова.

Острва се идентификују у финалној матрици анализом колона и редова за линеарном зависношћу. Сваки линеарно независан ред или колона представља једно острво. Чворови који га чине су одређени индексима свих елемената који су једнаки 1 у том реду, односно колони.

Потребно је анализирати стабилност тако добијене листе острва. Битно питање је да ли постоје острва без генераторских чворова. Такође је важно и комплементарно разматрање: да ли постоје острва у којима се искључиво налазе генераторски чворови, дакле без потрошача. У оба случаја прорачун токова снага за дато острво нема смисла и не може се спровести. За остале инстанце острва остаје још један корак пре приступања прорачуну токова снага: избор балансног чвора. Углавном се бира генераторски чвор с максималном активном

и реактивном снагом унутар острва. Прорачун токова снага се може извршити када су улазни параметри одређени. Ако поступак конвергира, на основу резултата се разматра стабилност, тако што се добијени резултати пореде с референтним вредностима за чворове, те се извлачи закључак по питању стабилности датог острва.

Реалне паметне електроенергетске мреже захтевају строга безбедносна ограничења. Углавном се примењује концепт тзв. „n-1“ безбедности, што значи да би мрежа морала бити у стању да издржи отказ једног елемента без последица по рад. То повлачи за собом да су мреже густо повезане с врло мало чворова који су само једном граном повезани с остатком мрежа. Симулација острва је стога тешка у реалним околностима, јер да би дошло до настанка острва, више испада мора да се догоди. У пракси се ретко разматрају случајеви с више испада, али су они свакако у опсегу рада овог алгоритма.

Кратак преглед спектралне теорије графова

Спектрална теорија графова је област која се бави проучавањем својстава Лапласове матрице и матрице суседности повезаних с датим графом. Посебан нагласак се ставља на везу сопствених вредности Лапласове матрице и повезаности графа, јер су те две ствари уско повезане. Примера ради, горња и доња граница најмање сопствене вредности Лапласове матрице која није нула пружају интуицију о томе колико је граф повезан. Док матрица повезаности графа указује на различите путање унутар графа.

Граф се дефинише као уређени пар скупова $G = (V, E)$. Елементи скупа V се називају чворовима, а елементи скупа E гранама. Њихова веза се може приказати на следећи начин, ако је $V = \{v_1, v_2, \dots, v_n\}$, постоји пар $\{v_i, v_j\} \in E$ ако и само ако постоји грана која спаја чворове v_i и v_j. Тако дефинисан граф се назива неусмереним графом. Код усмереног графа је редослед елемената у уређеном пару $\{v_i, v_j\}$ битан. Овде се разматрају неусмерени графови, осим ако је другачије назначено.

За два чвора v_i и v_j датог графа $G = (V, E)$ се каже да су суседни, ако постоји грана $\{v_i, v_j\} \in E$. Граф се назива потпуним, ако су сви чворови међусобно суседни. Потпуни граф са n чворова се означава са K^n, па је један од једноставнијих примера троугао K^3, тј. потпуни граф с три чвора.

Степен чвора се представља као $d(v)$ и одређује се као број суседа чвора v у графу G.

Постоје два типа матрица који се повезују с графом G: матрица суседности A_G и Лапласова матрица L_G.

Елементи матрице суседности A_G се дефинишу на следећи начин:

$$a_{ij} = \begin{cases} 1, \text{ако } \{i,j\} \in E \\ 0, \text{иначе} \end{cases}$$

Елементи Лапласове матрице L_G се дефинишу на следећи начин:

$$l_{ij} = \begin{cases} -1, \text{ако } \{i,j\} \in E \\ d(i), \text{ако } i = j \\ 0, \text{иначе} \end{cases}$$

Једна од најинтересантнијих својстава графа је његова повезаност, а Лапласова матрица представља згодан алат за његово испитивање и процену. Како је Лапласова матрица симетрична и све њене вредности су реални бројеви, следи да су и њене сопствене вредности реалне. Међутим, испоставља се и да су сопствене вредности додатно и ненегативне. То указује на то да је Лапласова матрица позитивно-полуодређена, односно, по дефиницији, ако је димензија $n \times n$, онда релација $x^T L_G x \geq 0$ важи за све $x \in R^n$. Посебно је битно нагласити да све сопствене вредности L_G постоје и да су позитивне или нула. Практично то значи да граф од n чворова има n сопствених вредности, који нису нужно различите. Обично се сопствене вредности означавају са $\lambda_1, \lambda_2, \dots, \lambda_n$, а пошто су реалне и позитивне или нула, следећа релација важи без губитка општости: $0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.

Већ је напоменуто да сопствене вредности Лапласове матрице пружају информације о повезаности графа. У циљу дефинисања повезаности, потребно је прво увести појам путање у графу. Путања је не-тривијални граф $P = (V, E)$, специфичног облика $V = \{v_1, v_2, \dots, v_n\}$ и $E = \{\{v_0, v_1\}, \{v_1, v_2\}, \dots, \{v_{n-1}, v_n\}\}$,

при чему су сви чворови v_i различити. Дефиниција путање одговара и интуитивном схватању. Указује на постојање јединственог редоследа грана од почетног до крајњег чвора. Граф G је повезан, ако постоји путања између свака два његова чвора. У овом погледу, путања је подграф графа G фокусиран на пут између изворног и одредишног чвора, па не мора да садржи све чворове графа G.

Може се доказати да је број сопствених вредности Лапласове матрице које су једнаке нула представља број повезаних компоненти у графу G. Повезана компонента у графу G је подграф $G' = (V', E')$, такав да $V' \subset V$ и $E' = \{\{v_i, v_j\} \in E | v_i, v_j \in V'\}$, при којем су свака два чвора $v_i, v_j \in V'$ повезана, док сваки пар чворова $\{v_i, v_k\}$ такав да $v_i \in V'$ и $v_k \in V \backslash V'$ није повезан. Као пример може послужити комплетан граф од n чворова, K^n. Његова Лапласова матрица има тачно једну сопствену вредност која једнака нули, док су преосталих $n-1$ једнаке n. То указује на једину повезану компоненту, која је уједно и цео граф.

Курант-Фишерова (Courant-Fischer) теорема пружа релације за одређивање појединих сопствених вредности матрице и гласи: Нека је A симетрична матрица димензија $n \times n$. Нека су $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ њене сопствене вредности, а $v_1 \leq v_2 \leq \cdots \leq v_n$ одговарајући сопствени вектори. Тада важе следеће релације:

$$\lambda_1 = \min_{x \neq 0} \frac{x^T A x}{x^T x}$$

$$\lambda_2 = \min_{x \neq 0 \wedge x \perp v_1} \frac{x^T A x}{x^T x}$$

$$\lambda_n = \max_{x \neq 0} \frac{x^T A x}{x^T x}$$

Теорема уводи нотацију која представља квадратну форму матрице и може се користити као оператор над Лапласовом матрицом. Помоћу дефиниције Лапласове матрице се добија релација

$$x^T L_g x = \sum_{(i,j) \in E} \left(x_i - x_j\right)^2$$

Као споредни резултат, имајући у виду да је квадратна форма збир квадрата, следи још једном да је Лапласова матрица позитивна полу-одређена. Међутим, битнија последица је да се сопствене вредности симетричне матрице могу представити кроз њену квадратну форму, па то важи и за Лапласову матрицу. Ако је x сопствени вектор с одговарајућом сопственом вредношћу λ, тада важи $x^T L_G x = \lambda$.

Проблем спектралне спарсификације графова

Као што је раније речено, најинтерсантнија чињеница везана за спектралну спарсификацију је да је могуће тако спарсификовати сваки граф. У ужем смислу, спектрална спарсификација указује да се граф $G = (V, E, w)$ може апроксимирати ретким подграфом који задржава исту Лапласову квадратну форму као и оригинални граф [34]. Лапласова квадратна форма је дефинисана са

$$x^T L_G x = \sum_{(u,v)\in E} w_{(u,v)}\big(x(u) - x(v)\big)^2, \tag{35}$$

где је x вектор реалних бројева од V елемената, а L_G је Лапласова матрица од G [1] [2]. Строго говорећи, Шпилман и Тенг [2] сматрају да је $\tilde{G}$ σ-спарсификација графа G, ако је следећа релација тачна за све x:

$$\frac{1}{\sigma} x^T L_{\tilde{G}} x \leq x^T L_G x \leq \sigma x^T L_{\tilde{G}} x. \tag{36}$$

Одредити $\tilde{G}$ за задати граф G је сврха овог и решења и његове основне методе - *sparsify*.

Постојећа библоптека отвореног кода за спектралну анализу графова [1] [35], која се овде проширује и користи, већ има следеће основне методе:

- *adjacencyMatrix* – Одређује матрицу суседности датог графа.
- *laplacianMatrix* – Одређује Лапласову матрицу датог графа.

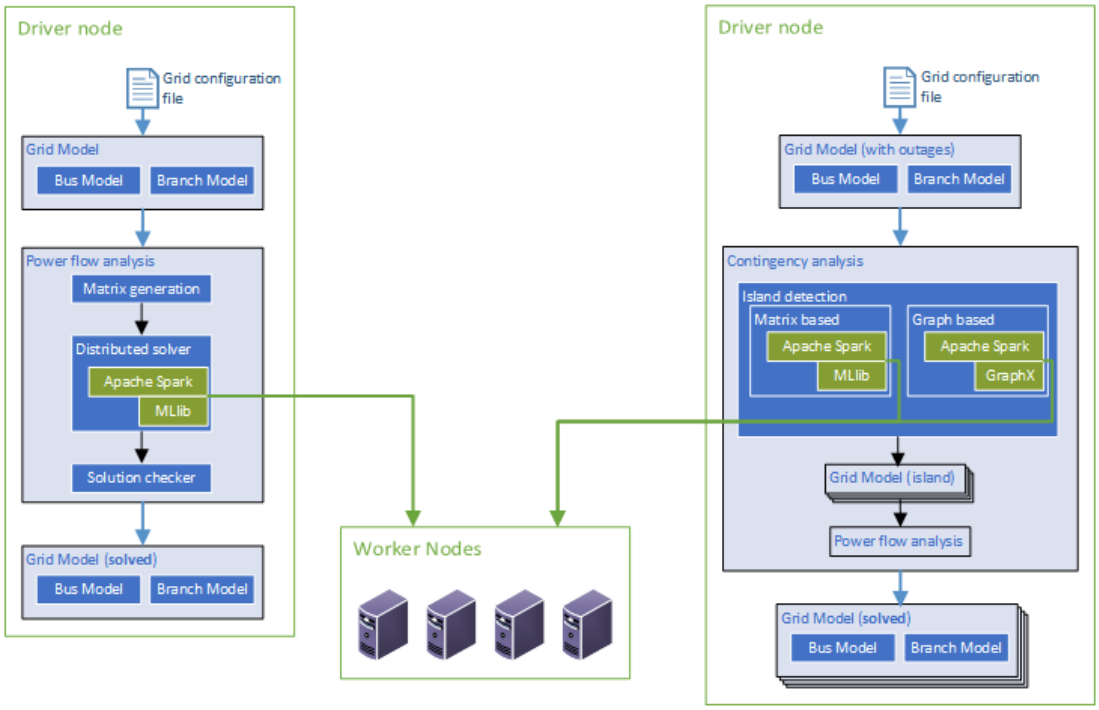
- *laplacianRayleighQuotient* – Рачуна Рајлијев количник Лапласове матрице графа за задати вектор.
- *laplacianQuadraticForm* – Рачуна квадратну форму Лапласове матрице графа за задати вектор.
- *spectralComparison* – Спектрано пореди два графа, што дозвољава да се успостави делимично уређена релација између графова.

Уводе се следеће методes: *volume*, *conductance*, *sum*, and *sparsify*. Почетна тачка разматрања је увек граф G, објекат Апачи Спарк библиотеке за обраду графова, и служи као улазни податак. Када је реч о гранама, чворовима, подграфовима, итд., увек је то у односу на граф G, осим ако је другачије експлицитно наведено. Основа решења је приступ проблему преко теорије скупова. Тако скупови чворова, грана, тежина, итд., односно уобичајени начини дефинисања графа, постају природно примењиви на Апачи Спарк парадигму [3]. Скалирање великог графа у дистрибуираном окружењу, као и многе рачунарски скупе операције, постају доступни.

Архитектура решења

Архитектура алата за анализу паметних мрежа

Слика 5 приказује преглед архитектуре решења. У овом поглављу се анализира сваки од њених компоненти.



Слика 5 Преглед архитектуре. Показује који делови анализе токова снага (лево) и анализе квара (десно) се изводи на управљачкој машини, а који се дистрибуирају на радне машине.

Модел мреже

Улазни параметри се прослеђују кроз датотеке које садрже модел мреже. То су структуиране текстуалне датотеке које дефинишу све познате параметре мреже. Формат је сличан IEEE CDF и PTI форматима и инспирисан MatLab структурама које се користе у алату MatPower [15].

Датотека модела мреже се састоји од шест сегмената раздвојених тачка-запетом (“;”).

1. Димензија – овај цео број означава број чворова у мрежи.
2. MVA база – дефинише корекције амплитуда у систему, узимајући у обзир њихове мерне јединице.
3. Број грана – представља број грана у графу мреже, које поред броја чворова чини основне информације за димензионисање модела.
4. Подаци о сабирницама – свака линија дефинише један чвор у мрежи. Очекивано је се број чворова поклапа са претходно дефинисаним параметром димензије. Сваки ред текста садржи запетом одвојене вредности које су карактеристичне за дату сабирницу, заједно с јединственим идентификатором чвора.
5. Подаци о генераторима – комплементарни подаци сабирница примењени на генераторске чворове. Сваки ред текста садржи запетом одвојене вредности које дефинишу један генератор у мрежи. Ови подаци се мапирају на генераторске топиве чворова коришћењем јединственог идентификатора чвора.
6. Подаци о гранама – сваки ред текста дефинише једну везу између два чвора у мрежи. Очекује се да се број таквих редова поклапа с поменутим бројем грана. Запетом подељене вредности дефинишу проводна својства сваке електроенергетске везе, која су од суштинског значаја за формирање матрице адмитанси. Грана између два чвора је одређена јединственим идентификаторима оба чвора (извор и одредиште).

Уз помоћ дате датотеке се генерише објекат модела мреже. Модел мреже се састоји од модела сабирница, који садржи информације о чворовима, и модела електроенергетских веза, који представља физичке гране између чворова.

Модел сабирница је скуп елемената који дефинишу сабирницу уз помоћ следећих информација:

1. ID – јединствени идентификатор чвора
2. Type – тип сабирнице, односно чвора (PQ, PV или балансни)
3. P_d – потрошња активне снаге
4. Q_d – потрошња реактивне снаге
5. G_{shunt} – одводна кондуктанса
6. B_{shunt} – одводна сусцептанса
7. P_g – активна снага која се генерише у датом чвору
8. Q_g – реактивна снага која се генерише у датом чвору
9. V – амплитуда напона
10. θ – фазни угао (померај)

Сегменти о генераторима и спојницама у датотеци модела мреже заједно дефинишу скуп чворова.

Почетне вредности за Њутн-Рафсонов метод могу бити постављене на конфигурацију „равног старта“, где су одговарајуће амплитуде напона и фазни помераји постављени на један и нула, респективно. Након сваке итерације анализе токова снага, амплитуда напона и фазни померај се освежавају текућим апроксимираним вредностима. Освежавање зависи од типа чвора: вредности референтних чворова се никада не освежавају, на генераторском чвору се само освежава вредност амплитуде напона, док се на потрошачким чворовима освежавају вредности и амплитуде напона и фазног помераја. Тако да се по закључку анализе коначно стање система налази очувано у моделу спојница.

Модел грана је скуп елемената који повезују чворове. Сваки елемент је дефинисан следећим параметрима:

1. fromBus – јединствени идентификатор изворног чвора
2. toBus – јединствени идентификатор одредишног чвора

3. R – отпорност гране
4. X – реактанса гране
5. B_{line} – паразитска сусцептанса гране
6. τ – амплитуда односа извода трансформатора, у случају да је повезан трансформатор с граном
7. θ_{shift} – угао фазног помераја, у случају да је повезан трансформатор с граном
8. branchStatus – индикатор који одређује да ли је грана активна или не

Користећи дате информације о свакој грани, горе наведене једначине се могу користити за генерисање матрице адмитанси. Како се подаци чувају у моделу мреже, испади се могу додавати и одузимати (помоћу branchStatus параметра) и матрица адмитанси се може поново генерисати како би осликавала нове услове.

Решење се ослања на чињеницу да је матрица адмитанси ретка матрица, користећи одговарајућу структуру података како би се оптимизовале операције и штедела доступна меморија. Такође, пошто је матрица адмитанси комплексна матрица, подељена је на две реалне матрице (2).

Када се улазни параметри једном обраде и успешно претворе у модел мреже, могуће је произвољан број пута анализирати различите сценарије испада и стабилности токова снага.

Решење проблема токова снага

Математичка формулација проблема токова снага је изложена у одељку Проблем токова снага. Слика 6 приказује алгоритам који је имплементирај у решењу. Начелна идеја је да се брзи и локализовани задаци (као што су иницијализација улазних параметара и провера добијеног решења) извршавају на управљачкој машини, док се рачунски захтевне операције (као што је решавање матричне једначине) дистрибуирају на радне машине.

Algorithm 1: Power Flow

```
1:  function PowerFlowCalculation
2:      gridModel ← grid model
3:      currentIteration ← 0
4:
5:      initialize gridModel
6:
7:      while currentIteration < MaxIterations do
8:          Populate the Jacobian matrix
9:          Populate the ΔS vector
10:         Solve the matrix equation ΔS = J · ΔX
11:
12:         if ΔX solution acceptable then
13:             Update gridModel with current solution
14:             return gridModel
15:
16:         Update gridModel with current solution
17:         currentIteration ← currentIteration + 1
18:
19:     return null
```

Слика 6 Алгоритам прорачуна токова снага

Улазни параметар је модел мреже. Њега је прво потребно иницијализовати, тј. поставити полазне претпоставке и генерисати матрицу адмитанси.

Итеративно се покушава доћи до задовољавајућег решења све док се предлог решења не прихвати или се не достигне максималан дозвољени број итерација.

Користећи једначине (19) и (20), $\boldsymbol{\Delta S}$ вектор се рачуна у зависности од типа чвора, тј. не узимају се у обзир елементи активне снаге за референтни чвор ни елементи реактивне снаге за потрошачке чворове.

Уз помоћ израза (23) до (26) и (27) до (30) се генерише Јакобијева матрица. Јакобијева матрица је квадратна матрица величине једнакој броју једначина које су потребне за представљање мреже у прорачуну токова снаге. Као што је већ речено, то не значи да је број једначина једнак броју чворова у систему, већ је умањен на основу броја различитих типова чворова, а представљен је изразом (31). Овај број је својство модела мреже и унапред познат, тако да је димензија Јакобијеве матрице дефинисана на почетку сваке

итерације. Ово је важна информација, јер је наивни приступ, који би подразумевао рачунање комплетне Јакобијеве матрице и касније је реконструисао, рачунски скуп. Прво, подразумева узалудно рачунање читавих редова и колона који ће бити у потпуности занемарене и одбачене. Примера ради, једначине (18) одређују уклањање свих редова и колона који садрже референце на Q_i или V_i (где і означава генераторски чвор) за све генераторске чворове. Друго, чак и са потпуном Јакобијевом матрицом, тј. укључујући сувишне елементе, и даље би постојала потреба за реструктуирањем матрице. Јава виртуелна машина (JVM) не подржава промену величине низова алоцираних у меморији, тако да би приступ подразумевао копирање одговарајућих елемената из једне матрице у другу, ефективно одржавајући две матрице у меморији и извршавајући додатну итерацију по свим елементима веће матрице.

Како би се оптимизовало рачунање, уводи се табела која мапира комбинације типова чворова, зависно од “from” и “to” крајева гране, на одговарајућу Јакобијеву торку (Табела 1). Јакобијева торка дефинише парцијалне изводе повезаних чворова који су релевантни за прорачун токова снага. Матрица се попуњава ред по ред, а одлуке се доносе за сваки елемент да ли треба да се попуни и којом вредношћу. Сваки чвор и његов однос са свим осталим чворовима се разматра и одговарајући парцијални изводи се рачунају за сваки пар повезаних чворова. Чињеница да су H_{ij} и N_{ij} у тренутном реду, док су J_{ij} и L_{ij} технички у реду испод, захтева извесно предзнање, јер чворови могу да заузимају нула, један или два реда у зависности од њиховог типа, па се користи Табела 1 да би се одредило да ли Јакобијева матрица садржи редове за тренутни чвор. Тако је само један пролаз кроз све чворове потребан да би се генерисала Јакобијева матрица, без потребе за сувишном алокацијом меморије.

i чвор	j чвор	Јакобијева торка
Slack	Slack	$[-, -, -, -]$
Slack	PQ	$[-, -, -, -]$
Slack	PV	$[-, -, -, -]$
PQ	Slack	$[-, -, -, -]$
PQ	PQ	$[H_{ij}, N_{ij}, J_{ij}, L_{ij}]$
PQ	PV	$[H_{ij}, -, J_{ij}, -]$
PV	Slack	$[-, -, -, -]$
PV	PQ	$[H_{ij}, N_{ij}, -, -]$
PV	PV	$[H_{ij}, -, -, -]$

Табела 1 Јакобијеве торке

Вектори $\boldsymbol{\Delta S}$ и $\boldsymbol{\Delta X}$ прате исте принципе по питању искључивања непожељних елемената као Јакобијева матрица. Такође, како би једначина (22) имала смисла, оба вектора морају да имају исти број колона као и матрица, тако да су димензије Јакобијеве матрице $n_{eq} \times n_{eq}$, а $1 \times n_{eq}$ за векторе $\boldsymbol{\Delta S}$ и $\boldsymbol{\Delta X}$, при чему је n_{eq} дефинисано изразом (31).

Решавање једначине (22) је тежак проблем. Поред тога је тај поступак потребно поновити неколико пута, за сваку итерацију која је потребна да би се стигло до решења.

У овом решењу је развијен алгоритам који користећи Апачи Спарк решава матричне једначине општег типа ($\boldsymbol{b = Ax}$), па самим тим и једначину, на дистрибуиран начин. Посматрано у грубим цртама (Слика 7), улазни параметри се припремају на управљачкој машини, а онда дистрибуирају као РДД-ови [3] (енгл. Resilient Distributed Datasets) на радне машине на решавање. Резултат

операције се враћа на управљачку машину због провере и припреме следеће итерације.

Algorithm 2: Distributed solving of matrix equation $b = A \cdot x$

```
1: procedure SOLVE(Matrix A, Vector x, Vector b)
2:     rowsRDD ← convert A into RDD of IndexedRows
3:     indexedRowMatrix ← convert rowsRDD into IndexedRowMatrix
4:
5:     sdd ← compute Singular Value Decomposition of indexedRowMatrix
6:     U ← get the U value from sdd, as IndexedRowMatrix
7:     S ← get the S value from sdd, as Vector
8:     V ← get the V value from sdd, as Matrix
9:
10:    Utrans ← transpose the value U, as IndexedRowMatrix
11:    UtransB ← Utrans multiplied with Vector b, as IndexedRowMatrix
12:    UtransBSinv ← UtransB multiplied with the inversed Vector S, as RDD of Tuples
13:
14:    UtransBSinvVector ← UtransBSinv collected to local Vector
15:
16:    x ← UtransBSinvVector multiplied with V
```

Слика 7 Алгоритам за дистрибуирано решавање матричних једначина

Први корак је паралелизација улазне матрице у РДД. РДД-ови су посебне апстракције скупова објеката које представљају основне објекте за све операције у Апачи Спарк окружењу. Они се партиционишу и дистрибуирају по доступним радним машинама и отпорни су на грешке приликом извршавања операције и испаде радних машина. Поставља се и питање основног типа РДД-а. Наиме, сваки РДД има тачно један тип података који је дефинисан скупом података, коришћеним за стварање тог РДД-а. Овај метод, тј. паралелизација колекције, је један од два начина да се направи РДД. Други је учитавање спољног скупа података. Ово питање је повезано с оптималним типом дистрибуиране матрице за следеће кораке алгоритма. Апачи Спаркова библиотека за машинско учење (енгл. MLlib) подржава следеће дистрибуиране матрице [36]:

1. RowMatrix – Матрица редова је основни тип матрице. Представља матрицу организовану по редовима, без индекса редова. То значи

да су редови распоређени по партицијама без неког значајног поретка, док су операције над матрицом свакако детерминистичке. С практичне стране су операције приступа индивидуалном реду као и здруживања скупе.

2. IndexedRowMatrix – Индексирана матрица редова је слична, тачније представља подтип, матрици редова по питању организације и доступних операција. Разлика је у томе што су редови јединствено означени, чиме се превазилазе недостаци матрице редова у смислу индексирања редова и операција здруживања.

3. CoordinateMatrix – Матрица координата представља скуп торки координата-вредност, при чему је координата дефинисана индексима реда и колоне. Општа препорука је да се овај тип користи за велике и ретке матрице. Такође, карактеристично је да има мање доступних операција у односу на матрицу редова и индексирану матрицу редова.

4. BlockMatrix – Матрица блокова је слична матрици координата у погледу локалне организације. Попут ње се састоји од скупа торки координата-вредност, с разликом да је вредност под-матрица оригиналне матрице, а координата је индекс блока. Величина блока је фиксирана. Битна предност овог типа је подршка за множење две матрице блокова, јер претходне матрице нуде само множење с локалним типовима (вектори или матрице).

Индексирана матрица редова је посебно интересантна за дати проблем, јер нуди два приступа факторизације матрица: QR-декомпозиција и декомпозиција матрице на сингуларне вредности (енгл. Singular Value decomposition - SVD). Поред тога и доноси додатну предност над матрицом редова у виду индексирања редова. Једини недостатак је да, судећи по изворном коду Апачи Спарка, многе операције индексирану матрицу редова прво претварају у матрицу редова и тек онда изврше захтевану операцију над тим типом с опционалним накнадним реиндексирањем, што са собом носи извесно нарушавање перформанси. Апачи Спарк документација сугерише да претварање

једног дистрибуираног типа у други може бити скупи с аспекта перформанси. Ипак, као што ће ускоро бити показано, практично постоје случајеви када претварање има боље перформансе од алтернативне комплексније имплементације. Типично је ово случај када је прерасподела података неизбежна, било да је изазвана претварањем типа или последицом имплементације.

Избор факторизације матрице је битан за проналажење инверзне матрице од A како би се решила матрична једначина

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}. \qquad (\mathbf{35})$$

Овде је изабрана декомпозиција матрице на сингуларне вредности (SVD). QR-декомпозиција је нумерички стабилна и чест избор за решавање система линеарних једначина, али захтева корак замене уназад. Декомпозиција матрице на сингуларне вредности се углавном користи за идентификацију и манупулацију сопствених вредности неке матрице. Међутим, има извесна својства која су корисна при декомпозицији квадратних матрица, какав је и овде случај. Наиме, дозвољава да се релативно брзо проблем сведе на серију множења једнодимензионалних вектора, што је брже од множења матрица. Нажалост је операција декомпозиције матрице на сингуларне вредности у Апачи Спарку првенствено намењена да проналази релативно мали број најзначајнијих сингуларних вредности, тако да има фиксирану максималну величину матрице за коју је могуће одредити све сингуларне вредности.

Иако је декомпозиција матрице на сингуларне вредности примењива на све матрице, овде су квадратне од посебног интереса, па је и дискусија сведена на њих.

Декомпозиција матрице на сингуларне вредности факторизује дату матрицу у производ три матрице

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{T} \qquad (36)$$

где је $\boldsymbol{U}$ ортогонална квадратна матрица, $\boldsymbol{S}$ дијагонална квадратна матрица, чије су вредности на главној дијагонали позитивни реални бројеви, а $\boldsymbol{V}^{T}$ је конјуговано транспонована матрица од ортогоналне матрице $\boldsymbol{V}$ па самим тим је и сама ортогонална. Све матрице у производу су исте димензије као $\boldsymbol{A}$. Једно

својство ортогоналних матрица, као и њихових унарних комплексних парњака, је да се конјугованим транспоновањем добија инверзна матрица. Према томе, након извршене декомпозиције матрице на сингуларне вредности, постаје тривијално добити инверзну матрицу и решавање једначине (36) постаје:

$$\boldsymbol{x} = \boldsymbol{V}\boldsymbol{S}^{-1}\boldsymbol{U}^{T}\boldsymbol{b}. \qquad (\mathbf{37})$$

Множење матрица је асоцијативан поступак. Иако редослед множења није битан с математичког становишта, у пракси оптималан редослед може да побољша перформансе. Циљ је да се матрице што пре сведу на векторе. Имплементације ове факторизације у Апачи Спарк окружењу враћа мешавину локалних и дистрибуираних структура података. Тако је $\boldsymbol{U}$ типа индексирана матрица редова па тиме и дистрибуирана, док је $\boldsymbol{V}$ локална матрица и враћена на управљачку машину. На радним машинама се извршава што је могуће већи број операција, а резултат се само једном купи и враћа управљачкој машини на даљу обраду, како би се комуникација између машина свела на минимум. Редослед операција множења је представљен заградама:

$$\boldsymbol{x} = \boldsymbol{V}\big(\boldsymbol{S}^{-1}(\boldsymbol{U}^{T}\boldsymbol{b})\big). \qquad (38)$$

Множење индексиране матрице редова $\boldsymbol{U}$ вектором **b** смањује матрицу на величину вектора, а при томе задржава тип индексиране матрице редова. Одређивање инверзне матрице од $\boldsymbol{S}$ подразумева рачунање реципрочних вредности њених елемената на главној дијагонали. Имплементација Апачи Спарка већ враћа $\boldsymbol{S}$ као вектор који садржи дијагоналне вредности уместо целе матрице, која би иначе била попуњена нулама, што олакшава поступак. Множење индексиране матрице редова из првог корака вектором $\boldsymbol{S}$ подразумева да се свака њена вредност подели одговарајућом вредношћу из $\boldsymbol{S}$. Ово се ефективно постиже функцијом мапирања (енгл. map), која примењује задату операцију на све елементе у РДД-у и при томе направи нови РДД. Резултујући РДД се прикупља и преноси на управљачку машину где се множи са $\boldsymbol{V}$ како би се добио резултат **x**.

Транспоновање дистрибуиране матрице, што је у случају матрице $\boldsymbol{U}$ предуслов за даљи поступак множења, представља изазов само по себи. Основни губитак перформанси је у прерасподели релевантних РДД-ова по

радним машинама. Ово је аутоматски процес којим управља Апачи Спарк окружење, где се скупови података преуређују у логички значајније партиције. Иако поступак не манипулише садржајем, тј. вредностима, РДД-ова, оптерећује мрежну инфраструктуру при дистрибуцији података на друге радне машине. Поред тога, значајни ресурси се користе за интерно вођење евиденције и контролне процедуре које дефинишу и стабилизују ново стање система. Губитак перформанси је посебно уочљив при операцијама с опширним скуповима података и великим бројем радних машина које обично имају релативно већу рачунарску моћ од пропусног опсега мреже. При имплементацији решења је примећено да је вештачко преуређивање индексиране матрице редова серијом операција мапирања, придруживања и придруживања по кључу (енгл. joinByKey) подложно грешкама и не резултују значајним побољшањем перформанси. Претварање индексиране матрице редова у матрицу координата, која доноси операцију транспоновања, се показало као перформантније са мањим оптерећењем комуникације између радних машина.

Након решавања израза (22), $\boldsymbol{\Delta X}$ се проверава критеријумима конвергенције, дефинисаним изразима (33) и (34). Вредности ε_θ и ε_V се типично дефинишу за свако решавање проблема токова снага. Постављање сувише строгих граница може довести до тога да Њутн-Рафсонова метода дивергира или да захтева сувише итерација да дође до решења. Док поставка сувише лабавих критеријума може утицати на прецизност решења. Баланс се обично емпиријски одређује. Овде је примећено да се може доћи до квалитетних резултата с критеријумима у опсегу 10^{-3} – 10^{-6}.

У случају да текућа итерација конвергира, улазни модел мреже садржи најсвежије измене, тј. решено стање система. Та инстанца модела мреже може бити даље коришћена за анализу стања система или даље симулације.

У случају да Њутн-Рафсонова метода не успе да одреди решење након датог броја итерација, поступак се сматра неуспелим. Максималан број итерација се такође емпиријски одређује. Углавном се очекује да је потребан релативно мали број итерација. Најгори сценарио је „равни старт“, јер је свако претходно решење истог система ближе тренутном, па тиме захтева и мање итерација. За „равни старт“, утврђено је да је максималан број итерација

достигнут током извршавања експеримената 6, тако да је 10 итерација постављено као разумна горња границе за дозвољени број итерација.

Решење проблема анализе квара

У поглављу Проблем анализа квара су изложене математичке основе за решавање анализе кварова у систему. Циљ је да се што више дистрибуирају рачунарски захтевне операција. У ту сврху су овде имплементирана два приступа. Први користи Апачи Спарк библиотеку за обраду графова (енгл. GraphX) [25] за анализу мреже путем анализе графова, а други користи бинарно множење матрица како би се одредила острва у систему.

Модел мреже служи као улазни податак. Потребно је утврдити почетно стање система пре симулације било каквих испада. То се ради извршавањем анализе токова снага на непромењеном моделу мреже. Полазно стање служи као референтна тачка за све измене које уводе кварови.

Поступци се разликују по процедури анализе повезаности, чији је излаз скуп потенцијалних острва. Оба поступка имају исту корисничку спрегу, тј. узимајући моделе чворова и грана из модела мреже, производе скуп индивидуалних острва. Острва су дефинисана скупом јединствених идентификатора чворова који се у њима налазе.

Приступ преко графова је једноставнији, па ће он бити и прво обрађен пре наставка о алгоритму бинарног множења матрица.

Библиотека за обраду графова (енгл. GraphX) је проширење Апачи Спарка, инспирисано значајем графова за савремене аналитичке проблеме. Наменска окружења за обраду графова, као на пример Прегел (енгл. Pregel) [37], су некада показивали драстично боље перформансе од окружења за дистрибуиране токове података, као што су MapReduce [38] или Апачи Спарк, када се у њима покушала емулација операција над графовима. Док истовремено, та окружења за обраду графова нису пружала исте могућности за анализу и обраду података нити толеранцију на грешке као за то намењени системи. Овакво стање је изискивало додатни развој међусистемске подршке. Апачи

Спарк библиотека за обраду графова је направљена да попуни овај празан простор, пружајући начин да се прошире операције над токовима података, толеранцију на грешке и дистрибуирани приступ са обрадом графова. Тако је Апачи Спарк проширен концептом који дозвољава дефиницију дистрибуиране граф структуре помоћу два РДД објекта, један за чворове, а други за гране. Оваква организација је корисна, јер структура графа може да се користи и као „прави“ граф, користећи операције и концепте својствене графовима, али и као скупови грана и чворова, на које су применљиве операције за паралелну обраду података својствене Апачи Спарк окружењу, као што су мапирање и здруживање.

Природно је представити мрежу електроенергетског система графом. Скуп чворова се прави од свих сабирница у моделу мреже. Гране се слично добијају из модела грана, који такође садржи и јединствене идентификаторе изворишних и одредишних чворова. Потребно је истаћи да класе чвора и гране могу да садрже и додатне карактеристика, у облику кориснички дефинисаног објекта или примитивног типа, али то за анализу повезаности није потребно. Објекат графа се креира помоћу ова два РДД објекта. Он садржи имплементацију операције за одређивање компонената повезаности, која враћа други објекат графа са сличном структуром као и полазни граф. Свако идентификовано острво се налази у РДД-у чворова и означено је најнижим јединственим идентификатором чвора у компоненти и листом чворова који јој припадају. Практично, трансформација РДД-а чворова у листу острва и њено враћање на управљачку машину решава проблем.

Анализа графова је не-итеративан и ефикасан приступ у случајевима када постоји окружење за обраду графова које се лако интегрише у систем, као што је случај са Апачи Спарком и библиотеком за обраду графова. Метод бинарног множења матрица је и даље фактички стандардан приступ за многе индустријске система, иако је током година унапређиван. Дистрибуиране операције над матрицама, укључујући множење, су у домену Апачи Спаркове библиотеке за машинско учење (енгл. MLlib) [36]. Разматрање устаљених пракси и изазов развоја дистрибуираног бинарног множења матрица су разлози који су инспирисали развој алтернативног метода за идентификацију острва.

Дистрибуирано бинарно множење матрица има три фазе:

1. Генерисање иницијалне матрице повезаности на основу стања система
2. Проналажење стабилне матрице острва
3. Идентификација острва на основу добијене матрице острва

Матрица повезаности се генерише на основу модела чворова. То је квадратна матрица величине једнаке броју сабирница у систему и садржи искључиво нуле и јединице као могуће вредности. Елемент с индексима i и j је једнак један, ако постоји грана која спаја чворове i и j, или симетрично чворове j и i. У супротном, вредност тог елемента је нула. Додатно, сви елементи главне дијагонале такође имају вредност један. Матрица повезаности је дистрибуирана матрица. За њену репрезентацију је погодна матрица координата, јер њен матрични елемент садржи координате и вредност, што дозвољава да се елементи индивидуално постављају. Другим речима, везана је за апстракцију елемента као основног чиниоца, а не на пример реда, као што је то случај с матрицом редова. Поред тога, пошто матрица повезаности представља физичке везе између чворова система, што је чини изузетно ретком, посебно за велике система, па је потребно релативно мало елемената за њено представљање. Ипак, пошто је матрица блокова једини тип који подржава множење с другом дистрибуираном матрицом, генерисана матрица координата се претвара у матрицу блокова на крају ове фазе. Тим претварањем се не губе значајно перформансе, јер је алтернатива да се директно прави матрица блокова. Креирање и организација блокова је значајно комплексније од креирања индивидуалних елемената матрице, пре свега што се ради о реткој матрици, те би се перформансе изгубиле при иницијализацији.

Следећа фаза одређује матрицу острва. Матрица блокова из претходне фазе се овде множи сама са собом. Максималан број множења зависи од броја чворова у систему и једнак је $\lceil log_2(n_{buses} - 2) \rceil$. Логаритамска тенденција је због тога што се у свакој итерацији текућа матрица множи са собом, уместо да се изнова множи са матрицом повезаности. Циљ је подизање матрице повезаности на $(n_{buses} - 2)$-ти степен, те је ово ефикаснији начин. Након сваког множења се све вредности које су испале веће од нула постављају на један.

Поступак се зауставља када се достигне максималан број итерација или све вредности у матрици постану једнаке један. Реална повезана мрежа резултује матрицом која након само неколико итерација достигне то стање. Разлог је повезан с $(n-1)$ сигурносним захтевом. Наиме, циљ је да се оствари високо повезана мрежа, како би кварови имали мању шансу да је раздвоје. Чворови су повезани већим бројем рута, што дозвољава да се достигне сваки чвор полазећи од било ког другог у релативно мало корака. Са становишта алгоритма је битно потврдити да не постоје два чвора чија је раздаљина већа од n_{buses}. У пракси су ретки дуги низови међусобно повезаних чворова, па је и раздаљина између свака два чвора мала у поређењу с n_{buses}. На крају ове фазе се употребљена матрица блокова претвара у матрицу координата, како би се олакшао приступ у следећој фази.

Последња фаза идентификује острва у добијеној матрици острва. Матрица координата се обрађује ред по ред. Њени елементи се филтрирају тако да прва координата одговара текућем реду. Резултујући скуп се сакупља и шаље на управљачку машину. Тамо се елементи чија је вредност нула стављају у листу текућег острва, а елементи чија је вредност један се означавају као посећени. Итеративно се сви елементи смештају у листе које одговарају острвима, док сви чворови нису означени као посећени. Како би резултат био конзистентан с резултатом добијеним анализом графа, потпуно повезана мрежа резултује јединственим скупом који садржи све чворова.

Независно од изабраног приступа, добијена листа острва се разматра. За сваку се креира нови модел мреже који одговара подмрежи тог острва. Само острва који имају барем један генераторски чвор се даље разматрају. Ако подмрежа не садржи претходно изабрани референтни чвор, бира се нови од доступних генераторских чворова у том острву. Иначе, потрошачки чворови остају непромењени. Сваки тако генерисани модел мреже се подвргава анализи токова снага, како би се оценило стање сваке подмреже. Може се догодити да неколико наизглед валидних подмрежа не конвергирају. Ово је прихватљив резултат, јер указује да дата подмрежа није довољно стабилна, тј. не постоји решење које ће задовољити израз (22). Постоји неколико разлога за такав исход. Један пример би био да генератори унутар мреже острва нису у стању да произведу довољно снаге потребне за снабдевање свих потрошача. Други

могући сценарио је да прорачун токова снага дође до резултата, али да су добијени параметри изван безбедносних норми за опрему на терену.

Поменута дискусија се односи и на испаде који уопште не разбијају мрежу, тј. не узрокују настанак острва. Сваки квар се сматра безбедним, дакле не нарушава стабилност мреже, ако резултује стабилном мрежом.

Примена на анализу паметних мрежа

Надзор стања мреже је битан, јер нерегуларности могу довести до отказа критичне инфраструктуре. Паметни уређаји повећавају обим мреже и уводе додатну несигурност у стање мреже. Традиционални уређаји који активно учествују су углавном статични с становишта доступности, нпр. индустријски трансформатор, или релативно мали (нпр. мало домаћинство у поређењу с електроенергетском мрежом града). Паметни уређају су често непредвидиви у погледу активности и далеко бројнији од традиционалних уређаја.

Прорачун токова снага пружа увид у стање мреже у датом тренутку. Ефикасност је ту битна, јер доступност анализе може помоћи да се контролише или спречи штета услед непредвиђеног квара. Иако се последице изненадних испада релативно споро пропагирају кроз мрежу, прорачун стања и даље не сме бити превише спор или скуп. Овде циљ обезбедити могућност честе анализе целокупне мреже без потребе за индивидуалном конфигурацијом или додатним ресурсима. Ако се овај процес аутоматизује, примера ради да се анализа токова снага извршава на сваких 30 минута, оператери могу благовремено добити обавештење о свакој нерегуларности.

Анализа кварова се користи за процену стања мреже након симулираних испада. Квар се уноси локалном изменом параметара модела мреже и може сезати од простог прекомерног или недовољног оптерећења елемената до комплетног искључивања целих секција. Корисно је знати утицај испада пре него што се он догоди, јер тада план опоравка може унапред да се одреди и да се поново провери користећи анализу квара за процену стања мреже након митигације.

Ефикасна имплементација анализе квара значи да администратори мреже могу систематично да врше „шта ако“ анализе на сваком сегменту мреже, предузимају превентивне мере и симулирају разне митигационе сценарије.

Архитектура решења спектралне спарсификације графова

Интуиција иза спектралне сличности графова

Граф од n чворова се може посматрати као симетрична квадратна матрица (g_{ij}) која садржи ненегативне реалне вредности, $g_{ij} = g_{ji} > 0$ за неки одређени пар $(i, j) \in \{1, \dots, n\}$. Граф који та матрица индукује одговара претпоставци да грана између чворова i и j постоји ако и само ако је одговарајућа вредност $g_{ij} = g_{ji}$ строго позитивна. У математички ширем смислу се на овакав начин моделује интеракција парова, у овом случају чворова повезаних граном, па вредност $g_{ij} = g_{ji}$ означава интензитет интеракције. Овде се ограничавамо на неусмерене графове, те то својство захтева симетричност матрице, тј. $g_{ij} = g_{ji}$. У општем случају, матрица није симетрична, па се усмерена грана од чвора i ка чвору j представља вредношћу g_{ij} која се разликује од вредности g_{ji}.

Могуће је граф G посматрати и као дефиницију посебне врсте функције. Ако се сваком чвору i додели реална вредност x_i, повратна вредност те функције је $\frac{1}{2}\sum_{i,j=1}^{n} g_{ij}(x_i - x_j)^2$. Испоставља се да се мноштво информација о структури графа G налази у могућим вредностима низа $\boldsymbol{x} = \{x_1, \dots, x_n\}$. Примера ради, посебан случај када су вредности x_i искључиво ограничене на 0 или 1, добија се структура тзв. пресека графа (енгл. cut), која за сваки подскуп чворова S представља укупну тежину грана које спајају скуп S с његовим комплементом. Другим речима, представља величину границе скупа S с остаком графа G.

Појам $(1+\epsilon)$-спектралне сличности два графа $G=(g_{ij})$ и $H=(h_{ij})$, истог броја чворова, подразумева да је за малу позитивну вредност ϵ однос вредности $\frac{1}{2}\sum_{i,j=1}^{n} g_{ij}(x_i-x_j)^2$ и $\frac{1}{2}\sum_{i,j=1}^{n} h_{ij}(x_i-x_j)^2$ између 1 и $(1+\epsilon)$ за сваки низ $\boldsymbol{x}=\{x_1,\dots,x_n\}$.

Кључни резултат који су Шпилман и његови сарадници доказали је да за сваки граф G постоји $(1+\epsilon)$-спектрално сличан граф H са не више од $4n^2$ грана. Тако матрица графа H репродукује вредности $\frac{1}{2}\sum_{i,j=1}^{n} g_{ij}(x_i-x_j)^2$ с високом прецизношћу, а да при томе има мањи број грана. Алгоритам који за улазни граф G враћа спарсификовани граф H се извршава у детерминистичком полиномијалном времену [39].

Матрица суседности – операција *adjacencyMatrix*

Матрица суседности је један начин логичког приказивања графа. Њени елементи указују на то да ли су два чвора графа повезана граном или не. Назив је и добијен због таквог представљања карактеристике суседства двају чворова.

Математички дефинисано, матрица суседности је квадратна матрица димензије једнаке броју чворова графа из којег се изводи. Њени елементи могу да имају вредности искључиво 0 или 1, с тим да су на главној дијагонали увек и искључиво вредности 0. У случају неусмереног графа, она је симетрична матрица. Елемент (i, j) ван главне дијагонале је једнак 1, ако и само ако постоји грана која спаја чворове і и j. Одатле и следи и разлика у симетрији за усмерене графове: ако је грана која спаја усмерена од і ка j, тада је (i, j) = 1, али не постоји веза у супротном смеру, па је (j, i) = 0.

Као алат анализе својстава графа, матрица суседности и њене варијације су врло корисне. Један такав пример је већ представљен у одељку (...) који се бави идентификацијом потенцијалних острва у мрежи представљеној графом, ту се и користи наменска бинарна матрица суседности. Показано је како суседство два чвора, које је тривијално описано релацијом „да ли постоји грана која их спаја“, може концептуално да се подигне на више нивое. Односно, да се коришћењем степеноване матрице суседности могу увидети и непосредне повезаности чворова. Матрица суседности првог степена показује све чворове

који су непосредно и директно повезани гранама. Матрица суседности другог степена уводи нове елементе који указују на повезаност чворова који претходно нису били повезани, тј. настанак нових грана. Међутим, ти нови елементи у матрици указују на посредну повезаност, односно повезаност два чвора преко трећег између њих. Како се степеновање повећава, тако нови се и број „корака“ између два ново повезана чвора повећава. Одатле следи и закључак за идентификацију острва: након последњег степеновања, свака непостојећа грана, тј. вредност 0 у матрици суседности, указује на постојање острва, јер је у супротном посредно растојања та два чвора у графу веће од укупног броја чворова, што је наравно немогуће.

Проучавање односа и својстава графа и сопствених вредности и сопствених вектора његове матрица суседности је од централног значаја и донекле и основ спектралне анализе графова.

У овом решењу је матрица суседности представљена као матрица координата Апачи Спаркове библиотеке за машинско учење, квадратних димензија које одговарају броју чворова у графу. То је подобан избор, јер су сви елементи постављени на вредност 0, сем оних који означавају повезаност. У општем случају се не може тврдити да је матрица суседности ретка матрица, јер то зависи од густине повезаности, односно броја грана. Ипак, коришћењем матрице координата се гарантује минималан број операција потребан за њено попуњавање, а и прати се парадигма да свака грана производи два елемента, те да логички нема смисла везивати се за колоне односно редове.

Како се у Апачи Спарк библиотеци за анализу графова грана представља изворишним и одредишним чворовима, а ово решење је фокусиране на неусмерене графове, потребно је очувати симетричност матрице суседности и обезбедити два елемента матрице координата за сваку грану графа. Скуп грана се због тога мапира на два скупа елемената матрице координата, где сваки елемент има симетричне координате у другом скупу, тј (изворишни чвор, одредишни чвор) и (одредишни чвор, изворишни чвор), и исту вредност 1 у оба.

У Апачи Спарк терминологији, од РДД-а грана се мапирањем добијају два РДД-а елемената матрице координата. А сама матрица координата која

представља матрицу суседности се добија спајањем та два РДД-а елемената и поставком броја редова и колона на број чворова у графу.

Лапласова матрица – операција *laplacianMatrix*

Лапласова матрица је још један вид представљања графова, а такође је и једна варијације матрице суседности. Иако је позната и под другим именима, као што су Кирхофова матрица и матрица адмитанси, име потиче од концепта дискретног Лапласовог оператора. Повезана је с многим корисним особинама графова.

Једна од њих је број обухватајућих стабала датог графа. Као што и име сугерише, обухватајуће стабло неког графа је стабло које садржи све чворове полазног графа. Стабло у теорији графова представља посебан тип графа где су свака два чвора повезана највише једном граном, другим речима стабло је повезани неусмерени ациклични граф. Други начин да се посматра дефиниција обухватајучег стабла је да је то минимални скуп грана који повезује све чворове. Једна очигледна последица овога је да обухватајућа стабла могу да имају искључиво повезани графови. Али исто тако постаје и јасније зашто поједини графови могу да имају већи број обухватајућих стабала, јер је нередак случај да постоји више начина да се уклоне цикличности из датог графа. Обухватајућа стабла имају своју примену као међукорак у алгоритмима за проналажење путање која спаја две задате тачке, нпр. Дијсктрин алгоритам.

У теорији графова, пресек представља поделу скупа чворова у два дисјунктна скупа. Пресек је минималан, ако величина или тежина пресека није већа од било којег другог пресека датог графа. Минимални пресек се може наћи у полиномијалном времену користећи Едмондс–Карпов алгоритам [40] [41] поред сличних метода. Слично томе, пресек је максималан, ако величина или тежина пресека није мања од било којег другог пресека датог графа. Не постоје методе за налажење максималног пресека у полиномијалном времену.

Проблем „најређег“ пресека (енгл. sparsest cut problem) подразумева поделу скупа чворова на два дела такву да се минимизира однос броја грана које „прелазе пресек“, односно које спајају та два новонастала скупа, и броја чворова

у мањој партицији. У практичном смислу је често циљ да се пронађу решења која што приближније испуњавају два услова: да су ретке, тј. да релативно мали број грана спаја два скупа, и балансиране, тј. да пресек приближно дели скуп чворова на пола. Овај проблем доказано припада класи НП-тешких проблема.

Најређи пресек графа се може приближно одредити користећи другу најмању сопствену вредност Лапласове матрице тог графа. Такође, спектрална декомпозиција Лапласове матрице има примену у одабраним алгоритмима из области машинског учења.

Важан концепт при конструисању Лапласове матрице је степен чвора. Степен чвора или валенца представља број грана које се стичу у датом чвору. Битан исказ је да је збир степена свих чворова у графу једнак двоструком броју грана, тј. за $G = (V, E)$ важи $\sum_{v \in V} \deg(v) = 2|E|$. Матрица степена чворова (D) је дијагонална квадратна матрица величине броја чворова у датом графу на чијој се главној дијагонали налазе степени сваког индивидуалног чвора, дефинисаног редом и колоном којима припада.

Лапласова матрица L се дефинише као $L = D - A$, где је D матрица степена чворова, а A матрица суседности датог графа. Дакле, на главној дијагонали Лапласове матрице се налазе степени сваког чвора, а елемент (i, j) ван главне дијагонале је једнак -1, ако и само ако постоји грана која спаја чворове i и j. Очигледно је како Лапласова матрица представља својеврсну „унапређену верзију“ матрице суседности, пружајући исте информације као и она уз додатне податке о степенима чворова. Поред тога, чињеница да је Лапласова матрица симетрична и дијагонално доминантна, тј. сума свих елемената изузев дијагоналног у сваком појединачном реду је мања или једнака од вредности дијагоналног елемента, повлачи да су све сопствене вредности Лапласове матрице веће или једнаке од 0. Оваква конструкција повлачи да је сума сваког реда и колоне једнака 0, јер је број грана који се стичу у датом чвору једнак степену чвора, а постоји и исти број вредности -1 који дефинишу сваку од тих веза.

У овом решењу је Лапласова матрица представљена као матрица координата Апачи Спаркове библиотеке за машинско учење, квадратних димензија које одговарају броју чворова у графу. То је подобан избор, јер су сви

елементи постављени на вредност 0, сем оних који означавају повезаност. У општем случају се не може тврдити да је матрица суседности ретка матрица, јер то зависи од густине повезаности, односно броја грана. Ипак, коришћењем матрице координата се гарантује минималан број операција потребан за њено попуњавање, а и прати се парадигма да свака грана производи два елемента, те да логички нема смисла везивати се за колоне односно редове.

Како се у Апачи Спарк библиотеци за анализу графова грана представља изворишним и одредишним чворовима, а ово решење је фокусирано на неусмерене графове, потребно је очувати симетричност Лапласове матрице и обезбедити два елемента матрице координата за сваку грану графа. Скуп грана се због тога мапира на два скупа елемената матрице координата, где сваки елемент има симетричне координате у другом скупу, тј (изворишни чвор, одредишни чвор) и (одредишни чвор, изворишни чвор), и исту вредност -1 у оба.

Поред тога је потребно и одредити елементе на главној дијагонали. Апачи Спарк библиотека за анализу графова пружа интересантан концепт агрегирања порука. Наиме, тај оператор примењује користи две кориснички дефинисане функције и примењује их на све елементе графа. Прва је слање поруке која пружа информације о грани, одредишном и изворишном чвору и даје могућност слања поруке одредишном и изворишном чвору. Може се рећи да је то аналогно функцији мапирања у парадигми мапирања и свођења. Друга је спајање порука, прихвата две поруке намењене истом чвору и спаја их у једну. По претходној аналогији, ово одговара кораку спајања у парадигми мапирања и свођења. Користећи овај оператор, могуће је са сваке гране послати њеном одредишном и изворишном чвору поруку с вредношћу 1, а онда их свести у јединствен број на сваком чвору који је и степен тог чвора. Како се као резултат те операције добија РДД који мапира јединствени идентификатор чвора на његов степен, лако је претворити тај РДД у РДД елемената матрице координата, који онда садржи дијагоналне елементе Лапласове матрице.

Након што су добијени РДД-ови грана на еквивалентан начин као у случају матрице суседности, они се спајају с РДД-ом дијагоналних елемената. А сама матрица координата која представља Лапласову матрицу се добија

спајањем та три РДД-а елемената и поставком броја редова и колона на број чворова у графу.

Запремина графа – операција *volume*

Запремина представља збир степени сваког чвора у датом подскупу чворова. Степен чвора је једнак броју грана које се стичу у том чвору. Како се овде разматрају неусмерени графови, није битно да ли су гране долазеће или одлазеће. Запремина је константна, ако је подскуп чворова заправо цео скуп чворова графа G, и једнака двоструком броју грана графа G. Ако се ради о подскупу у ужем смислу, прво се одреде степени свих чворова у графу G и креира скуп торки које упарују степен неког чвора с његовим јединственим идентификационим бројем. Резултујући скуп се пореди с идентификационим бројевима чворова у траженом подскупу, те као резултат остају само они чворови који се налазе у подскупу намапирани на своје степене. Збир степени чворова из тог скупа је тражена запремина за задати подскуп чворова.

Проводност графа – операција *conductance*

Проводност графа, позната и као Чигерова константа, је формално повезана с конвергенцијом насумичног хода по графу ка униформној расподели. Име потиче од сличности с значајем насумичних ходова у електричним мрежама. Овде се проводност користи примарно за партиционисање графа у смислу процене квалитета локалног кластера. За кластер се сматра да има висок квалитет, ако је релативно густо повезан унутар себе, а релативно ретко с остатком графа. Другим речима, проводност графа је однос броја грана које повезују кластер с остатком графа и броја грана унутар кластера [19].

У овом решењу се проводност рачуна у односу на дати скуп чворова, тј. кластер, који чине подграф, који у општем случају може бити и цео граф G. Прво се одређује број грана које излазе из кластера, тј. оне гране чији је један

чвор унутар кластера, а други изван. Следећи корак је одређивање запремина чворова у кластеру и преосталих чворова у графу G. Битно је нагласити да се запремине подграфова рачунају користећи „оригиналне“ степене чворова у изворном графу G. Проводност се добија односом броја грана унутар кластера и мање од две добијене запремине.

Збир графова – операција *sum*

Концепт сабирања два графа можда делује неинтуитивно. Ипак, као што ће касније бити показано, важан корак у спарсификацији графова је тзв. партиционисање у одређене подграфове, чији збир даје јединствен спарсификовани граф.

У спектралном смислу, збир два графа резултује графом чија је Лапласова матрица једнака збиру Лапласових матрица сабирака. То практично значи да је тежина сваке гране у резултујућем графу једнака збиру одговарајућих грана у графовима сабирцима. У случају да су скупови чворова сабирака дисјунктни, збир графова је проста унија.

Збирни граф је дефинисан својим скуповима чворова и грана. Скуп чворова је унија скупова чворова графова сабирака. У случају да се једна грана налази у скуповима грана оба сабирка, одговарајућа тежина гране у збирном графу ће бити збир њихових тежина. Ако је грана присутна у само једном графу сабирку, исти случај се примењује, с тим да се тежина непостојеће гране сматра да је нула. Проблем који се јавља је униформна идентификација грана. Грана је у општем случају дефинисана својим изворним чвором, одредишним чвором и тежином. Имајући у виду да је овде реч о неусмереним графовима, не постоји разлика између грана код којих су (идентични) чворови заменили места, тј. за потребе сабирања, такве гране се сматрају једнаким и као такве треба да буду сабране. Ипак, не постоји правило које гарантује да чворови у гранама не могу да замене места. Како би се избегао овакав случај, оба скупа грана се манипулишу, тако да је гарантован растући поредак идентификационих бројева чворова у торци која дефинише грану, тј. $(a, b) => (b, a), \text{ако } a > b$. На овај

начин се обезбеђује јединствен кључ, којим се одговарајуће гране, које у општем случају имају различите тежине, могу повезати с чворовима које повезују. Резултат је скуп који јединствено мапира чворове гране на тежине које одговарајућа грана има у скуповима грана оба графа сабирка. Операција спољног спајања обезбеђује да, у случају да грана није присутна у једном графу сабирку, ће се она ипак наћи у спојеном скупу с специјалном назнаком која означава „непостојећу“ тежину. Резултујући скуп грана се формира условним сабирањем тежина и задржавањем идентификационих бројева чворова. Тако је задовољен други захтев за сабирање два графа.

Спарсификација графа – операција *sparsify*

Ова сложена метода се налази у сржи процеса спарсификације. Састоји се од два засебна корака, који су повезани заједничким подешавањем тежина. Прво се граф партиционише и врши се одабирање скупова грана резултујућих подграфова. Онда се добијени графови сажимају у јединствен спарсификован граф.

Како би подржавао графови с произвољним тежинама, поступак спарсификације је ограничен на графове чије тежине имају реалне вредности између нула и један. Ово ограничење се практично може лако превазићи тиме што се све тежине скалирају, тј. свака подели највећом тежином у графу, пре спарсификације и још једном у супротном смеру након, тј. множењем истом вредношћу. То је још један задатак који се лако испуњава коришћењем Апачи Спарк парадигме, чак и за велике графове. Међутим, проблем овде није могућност скалирања, већ чињеница да тежине заиста могу бити насумичне, тј. велики број цифара иза децималне запете. Мале разлике, које су „довољно далеко од децималне запете“, вероватно не утичу на резултат, али повећавају комплексност прорачуна и могу довести до немогућности конструкције спарсификованог графа. Зато се тежине скалирају на извесни број бита након децималне запете. Зависно од тог броја бита, исти број подграфова је креиран зависно од бинарне репрезентације индивидуалних (скалираних) тежина грана. Над сваким од тих подграфова се извршавају следеће операције.

Пре уласка у детаље партиционисања и одабирања графова, постоји проблем који треба поменути. Апачи Спарк библиотека за обраду графова нуди ефикасан метод за одређивање компоненти повезаности неког графа, која се користи у поступку декомпозиције. Међутим, постоји пријављен и тренутно још нерешен проблем [42] који доводи до заглављивања методе у бесконачној петљи. Током експерименталне евалуације је повремено долазило до испољавања тог понашања при обради великих графова. Имајући у виду да је проблем како се чини повезан с оптерећењем целог окружења и другим факторима, најједноставнији начин да се превазиђе је покретање прорачуна наново.

Циљ при партиционисању је изолација дела графа с датом циљном проводношћу [43]. Ако је добијена партиција велика, тј. има велику проводност, та партиција и преостали комплемент графа се даље партиционишу док се не достигне жељена проводност. Ако партиција испуњава циљ, њен комплемент се даље партиционише, док и он не достигне тражену проводност. Резултат партиционисања је скуп подграфова. Над сваком од њих се врши одабирање, што представља насумични процес, при чему скуп чворова подграфа остаје непромењен док се гране скалирају на основу расподеле вероватноће. Формално [43], овај корак прави $(1+\epsilon)$-апроксимацију подграфа, при чему је ϵ рационалан параметар. Овако добијени скуп подграфова се на крају процеса сабира у јединствен граф. Такви графови се већ могу сматрати у извесној мери спарсификованим.

У овом тренутку је иницијални граф, након скалирања тежина грана, декомпонован на подграфове. Сваки од њих је индивидуално партиционисан и подвргнут одабирању у спарсификовану верзију себе. Даља спарсификација грана подграфова се одвија идентификацијом оних који најмање доприносе укупној проводности подграфа. Напослетку се спарсификовани граф, настао као збир тако добијених подграфова, враћа као резултат операције.

Укратко, алгоритам итеративно разбија граф у мање, већином преклапајуће, графове и покушава да смањи број грана у сваком кораку. Сваки од корака спарсификације има једну сличност. Сваки од тренутно релевантних подграфова се дели у скуп његових подграфова, њихове гране се обрађују на

неки начин, и бивају сабрани у јединствен збирни граф. Ништа се не губи током поновљеног разбијања, чак иако се подграфови углавном преклапају, јер сабирање графова чува чворове и само манипулише тежинама грана које постоје у било ком подграфу.

Експериментални резултати и дискусија

Окружење за алат анализе паметних мрежа

Окружење

За рачунарску инфраструктуру на којој ће да се извршавају експерименти је изабрана платформа за рачунарство у облаку фирме Амазон (енгл. Amazon cloud computing platform, Amazon Web Services (AWS)). Тестирање је извршено на различитим конфигурацијама Амазонових ЕЦ2 инстанци (енгл. Amazon Elastic Compute Cloud (EC2)). Резултати мерења су ограничени на типове виртуелних машина оптимизованих за рачунање („c3“ типови) и графичко процесуирање опште намене („p2“ типови), а чије карактеристике су приказане наТабела 2. Инстанце „c3“ користе Intel Xeon E5-2680 v2 (Ivy Bridge) процесоре, док „p2“ инстанце имају графичке процесоре NVIDIA K80, од којих сваки има 2496 процесорских језгара за паралелну употребу, 12Gb графичке меморије и процесоре типа Intel Xeon E5-2686v4 (Broadwell).

Instance Model	CPU Cores	GPU Cores	GPU RAM	RAM
c3.4xlarge	16	n/a	n/a	30 GB
c3.8xlarge	32	n/a	n/a	60 GB
p2.8xlarge	32	8	96 GB	488 GB
p2.16xlarge	64	16	192 GB	732 GB

Табела 2 ЕЦ2 Инстанце

Постоје решења која омогућавају извршавање Апачи Спарк задатака на графичким процесорима (енгл. CUDA GPUs), као што је SparkGPU [44], али се

они ослањају на проширивање Апачи Спарк језгра подршком за графичке процесоре, па се тако практично добија нека нова наменска Апачи Спарк верзија. Датабрикс (енгл. Databricks), као компанија која комерцијализује Апачи Спарк, нуди готову верзију Апачи Спарка за употребу са графичким процесорима, која не захтева измене кода или поновно превођење апликације која би је користила. У овом решењу је употребљена та верзија за експерименте. Велики недостатак је да у тренутку извођења тестова, Датабриксов Апачи Спарк са подршком за графичке процесоре служи само као распоређивач за библиотеке које већ подржавају извршавање на графичким процесорима (нпр. TensorFrames). Ово значи да Апачи Спарк не добија на перформансама када се извршава на кластеру графичких процесора. Чак супротно од тога, губи на перформансама, јер се користи само једна нит извршиоца на свакој радној машини [45]. Тако да ефективно, не само да се не користе ресурси доступних графичких процесора, већ се и процесорски ресурси ограничавају на једно језгро по машини.

Експерименти подразумевају две конфигурације за сваки тип инстанци, како би се показала скалабилност решења. Једна конфигурација има пет радних машина, а друга десет, док обе имају по једну управљачку машину. Како није одобрено повећање горње границе броја инстанци с графичким процесорима од стране фирме Амазон, експерименти су ограничени на конфигурација од две и пет радних машина за инстанце типа p2.8xlarge, а на само две радне машине инстанце типа p2.16xlarge. Међутим, имајући у виду претходну дискусију о ограничењима графичких процесора, скалирање експеримената на овим типовима инстанци је од изузетно скромног значаја.

Експерименти су извршавани уз помоћ Датабрикс интернет окружења, у случају графичких процесора, а Амазоново EMP (енгл. Amazon Elastic MapReduce (EMR)) окружење, за случај процесора. Оба нуде апстракцију изнад „чистих“ ЕЦ2 инстанци, која омогућава аутоматско постављање Апачи Спарк и повезаних сервиса (нпр. Ganglia или S3) на виртуалне машине при њиховом покретању унутар кластера. Од корисника се онда само очекује да одреди конфигурацију кластера, подигне апликацију, конфигурише параметре Апачи Спарка, подеси потребне зависности и изврши апликацију. Тако се избегавају

учестали ручни кораци у виду инсталације Апачи Спарка и повезаних система, а који би били неопходни при сваком стартовању „чистих“ ЕЦ2 инстанци.

Исправна конфигурација Апачи Спарка није тривијалан поступак [46], јер је у понуди око 180 параметара [47], тако да фино подешавање извршавања неке апликације се углавном своди на иницијалну процену, на основу доступне документације, и накнадно експериментисање док се перформансе не поправе. На пример, подешавање партиционисања РДД-ова је представљало изазов, а има значајан утицај на укупне перформансе. Ако РДД има превише партиција, систем губи перформансе на додатне операције управљања које нису директно везане за сам задатак, док се у случају с премало партиција у систему могу појавити неактивне радне машине и повећати мрежна комуникација, што опет нарушава перформансе. У идеалном случају, радне машине у потпуности користе сва доступна процесорска језгра током целокупног извршавања задатка. Током имплементације је утврђено да број партиција треба да буде функција доступних процесорских језгара, а не количине података која се обрађује. Време извршавања се смањило шест пута, када је број партиција постављен на петоструку вредност броја доступних процесорских језгара.

Улазне извршне датотеке и излазне протоколарне датотеке се смештају на Амазонов С3 (енгл. Simple Storage Service (S3)), у случају извршавања у EMP окружењу. Корисничка спрега коју нуди Датабрикс има сопствене контроле које апстрахују Амазонове сервисе испод, што резултује у фокусиранијем искуству за корисника, иако на крају пружа исте информације.

Случајеви тестирања

За евалуацију решења су коришћене мреже које обухватају ситуације експерименталног и наменског порекла, као и реалне случајеве коришћења. Користан извор одабраних примера мреже је репозиторијум истих који долази уз програмски пакет MatPower [15]. Експерименти су извршени на 21 мрежи, с бројевима чворова у опсегу од 4 до 9241, добијених из различитих извора [32], [48], [49], [50], [51], [52], [53]. Овде су резултати приказани за мреже наведене, заједно са њиховим пореклом и значајем, у Табела 3. Разлози се воде презентационим вредностима. Избор неколико репрезентативних узорака

демонстрира могућности и аналитичку вредност решења без губитка општости и читљивости.

Број Сабирница	Порекло
30	Део електричне мреже САД, децембар 1961. [50]
300	Развијено од тима за тестне системе, 1993. [50]
1354	Део европске високонапонске преносне мреже [53]
2383	Део пољске мреже током услова високог оптерећења зиме 1999-2000 [52]
3384	Део пољске мреже током услова високог оптерећења зиме 2007-2008 [52]
9241	Део европске високонапонске преносне мреже [52]

Табела 3 Примери мрежа

Резултати експеримената и дискусија

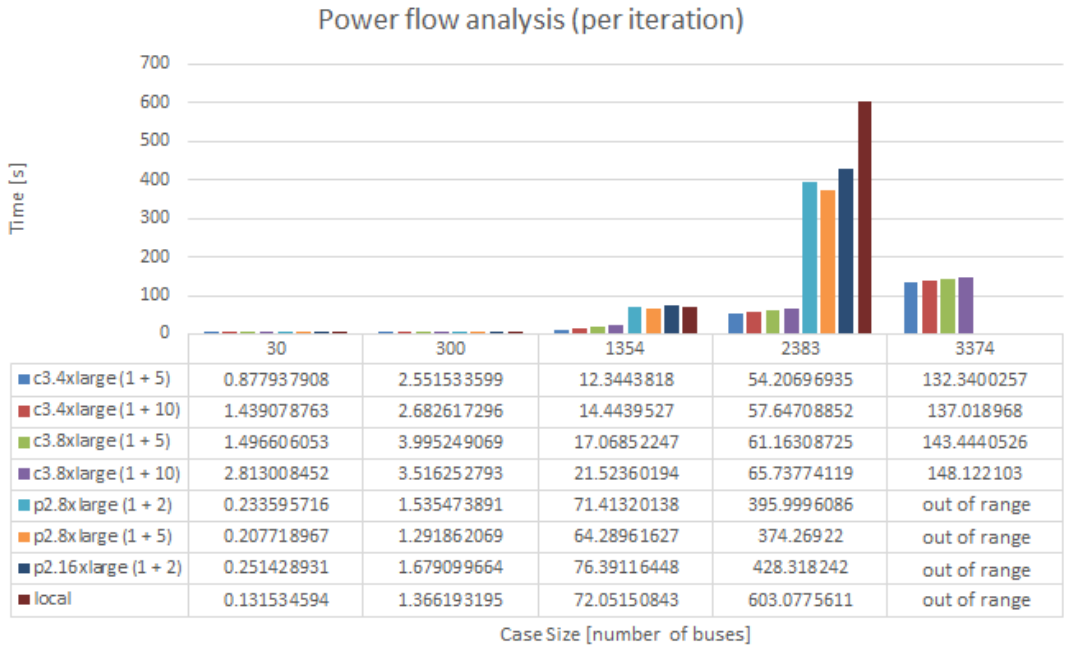
Циљ је оценити и упоредити перформансе дистрибуираног извршавања у различитим конфигурацијама, као и на локалном систему, како би се добиле референтне вредности. Локални систем се састоји од једне машине са 12 Gb меморије и четворојезгарним процесором типа Intel i5 2500k (Sandy Bridge) на 3.3 GHz. У ту сврху је Апачи Спарк конфигурисан у моду рада локалног управљача (енгл.“local master”). То значи да машина на којој се извршава апликација служи истовремено као управљачка машина и известан број радних. Број радника је дефинисан један по језгру процесора и подешен да користи сва доступна језгра. Под овим околностима су извршавани сви експерименти.

Резултати извршавања прорачуна токова снага су приказани на Слика 8. Времена извршавања су скалирана тако да приказују трајање једне итерације

Њутн-Рафсонове методе, јер су потребни различити бројеви итерација за прорачуне појединих мрежа (Табела 4). Утврђено је да у просеку мање од 2% времена прорачун извршава на управљачкој машини припремајући текућу итерацију, нпр. генерисање ΔS или Јакобијеве матрице, провере валидности тренутног решења.

Број сабирница	30	300	1354	2383	3374
Број итерација	3	5	5	6	6

Табела 4 Број итерација по прорачуну токова снага



	30	300	1354	2383	3374
c3.4xlarge (1 + 5)	0.877937908	2.551533599	12.3443818	54.20696935	132.3400257
c3.4xlarge (1 + 10)	1.439078763	2.682617296	14.4439527	57.64708852	137.018968
c3.8xlarge (1 + 5)	1.496606053	3.995249069	17.06852247	61.16308725	143.4440526
c3.8xlarge (1 + 10)	2.813008452	3.516252793	21.52360194	65.73774119	148.122103
p2.8xlarge (1 + 2)	0.233595716	1.535473891	71.41320138	395.9996086	out of range
p2.8xlarge (1 + 5)	0.207718967	1.291862069	64.28961627	374.26922	out of range
p2.16xlarge (1 + 2)	0.251428931	1.679099664	76.39116448	428.318242	out of range
local	0.131534594	1.366193195	72.05150843	603.0775611	out of range

Слика 8 Резултати перформансног тестирања прорачуна токова снага, скалирани на време потребно за једну итерацију

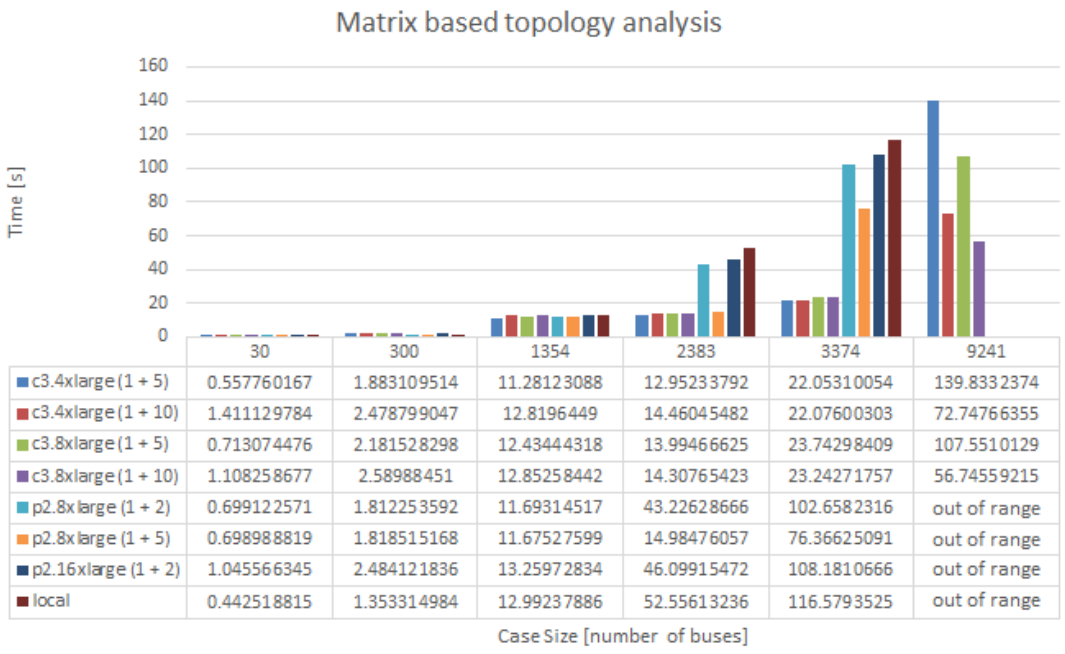
Измерена времена указују ја релативну сличност дистрибуираног и локалног извршавања за мање мреже, односно мањи број чворова. Понекад се чак дешава да локално извршавање буде брже од дистрибуираног. То је очекивано понашање, јер су скупови података довољно мали да додатне

операције потребне за њихово партиционисање, слање на радне станице и поновно сакупљање, премашују комплексност самог прорачуна. Такође, то показује да су мање мреже обрадиве на појединачним машинама, што је и разлог зашто комерцијална решења за анализу електроенергетских мрежа теже да разбију мрежу на мање делове пре прорачуна токова снага. Међутим, разлике брзо расту с порастом величине мреже. За мрежу од 3374 чворова већ није изводљиво исцртати време локалног извршавања, јер је било реда величине неколико сати. У таквим случајевима предности дистрибуираног приступа јасно долазе до изражаја. Нажалост, могућност Апачи Спарка да изврши декомпозицију матрице на сингуларне вредности је ограничена на матрице од максимално 17515 колона. У изворном коду за матрицу редова, ово је објашњено коментаром да веће матрице премашују могућности пакета Бриз. Тако да се испоставља да је ово ограничење пакета Бриз (енгл. Breeze) [54], што је програмска подршка за нумеричку обраду, односно библиотека коју Апачи Спарк користи. Тако да није било могуће извршавати прорачун токова снага за мрежу од 9241 чворова и веће системе. Овим се такође објашњава и благи губитак перформанси при скалирању и надоградњи „c3“ инстанци. Циљ је да се максимално искористи доступна процесорска снага, колико год је то могуће. Прорачун на примеру мреже од 3374 чвора је користио око 80% процесорске могућности у просеку при извршавању на кластеру од пет радних машина типа c3.4xlarge. Исти прорачун је на кластеру од десет радних машина типа c3.8xlarge захтевао само око 30% процесорске снаге у просеку. Када се уклони ограничење прорачуна декомпозиције матрице на сингуларне вредности, анализа перформанси може бити у потпуности извршена на већим мрежама које би у потпуности користиле доступне ресурсе кластера.

Експерименти с инстанцама опремљеним за графичко процесуирање си извршавани користећи Датабрикс подршку за Апачи Спарк на графичким процесорима. Због недостатка адекватне подршке у Апачи Спарк решењу, они су показали сличне резултате као и локална извршавања. Мање побољшање у перформансама је због напредније конфигурације процесора (обичног, не графичког) и меморије.

График који приказуке резултате перформансног тестирања тополошке анализе матричним приступом је приказан на Слика 9. У начелу су иста

разматрања примењива као и за експерименте за прорачун токова снага. Случај мреже од 9241 чвора је од посебног интереса, јер није могао бити анализиран у претходној дискусији. Уочљиво је побољшање перформанси при повећању броја радних машина или употребе бољих EC2 инстанци. Значај скалирања је такође евидентан, десет радних машина типа c3.4xlarge показује за око 30% боље перформансе од кластера од пет инстанци типа c3.8xlarge. У екстремном случају, кластер од десет радних машина типа c3.8xlarge је скоро три пута бржи од конфигурације од пет радних машина типа c3.4xlarge. Ако се додатно узме у обзир да мање мреже показују сличне разлике у перформансама међу конфигурацијама инстанци типа c3 као експерименти чији су предмет прорачун токова снага, и да оба поступка користе Апачи Спарк библиотеку за машинско учење (енгл. MLlib), резултати матричне тополошке анализе подржавају тврдњу да би веће мреже у потпуности користиле доступне ресурсе додељеног кластера.

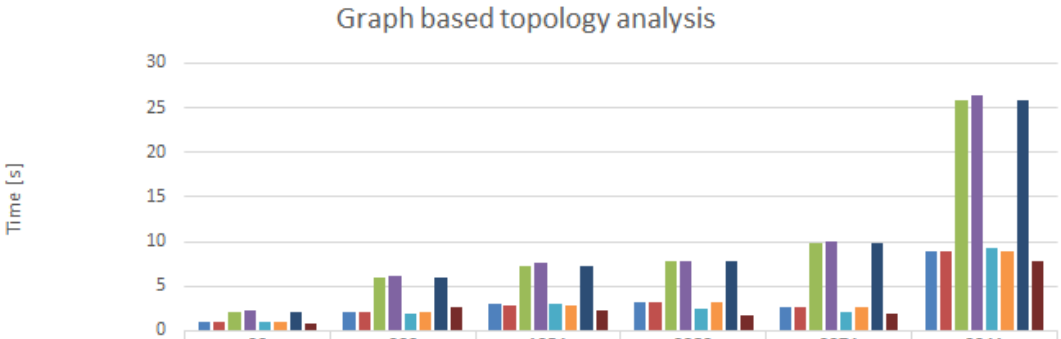


	30	300	1354	2383	3374	9241
c3.4xlarge (1 + 5)	0.557760167	1.883109514	11.28123088	12.95233792	22.05310054	139.8332374
c3.4xlarge (1 + 10)	1.411129784	2.478799047	12.8196449	14.46045482	22.07600303	72.74766355
c3.8xlarge (1 + 5)	0.713074476	2.181528298	12.43444318	13.99466625	23.74298409	107.5510129
c3.8xlarge (1 + 10)	1.108258677	2.58988451	12.85258442	14.30765423	23.24271757	56.74559215
p2.8xlarge (1 + 2)	0.699122571	1.812253592	11.69314517	43.22628666	102.6582316	out of range
p2.8xlarge (1 + 5)	0.698988819	1.818515168	11.67527599	14.98476057	76.36625091	out of range
p2.16xlarge (1 + 2)	1.045566345	2.484121836	13.25972834	46.09915472	108.1810666	out of range
local	0.442518815	1.353314984	12.99237886	52.55613236	116.5793525	out of range

Слика 9 Резултати перформансног тестирања тополошке анализе матричним приступом

Најбоље је поредити резултате тополошке анализе добијене преко анализе графова с њеном матричном алтернативом. С графика Слика 10 је очигледно да показује боље перформансе од матричне тополошке анализе.

Битан закључак је да нема практичне потребе да се дистрибуира процес за мале графове. Графови су оцењени као мали, јер је Апачи Спарк подршка за анализу графова (енгл. GraphX) тестирана на графовима који имају милионе чворова [25], [24]. Рачунарска моћ кластера је великим делом неискоришћена и под утицајем извршавања додатних операција.



	30	300	1354	2383	3374	9241
c3.4xlarge (1 + 5)	0.975195602	2.017794842	2.952260189	3.219670738	2.740817398	8.955701344
c3.4xlarge (1 + 10)	1.016142621	2.016799081	2.917491082	3.161749899	2.679823808	8.995971311
c3.8xlarge (1 + 5)	2.128718829	5.913769344	7.321784448	7.863164328	9.854708256	25.78978928
c3.8xlarge (1 + 10)	2.264113157	6.136128295	7.590892804	7.832341303	10.0517955	26.39472823
p2.8xlarge (1 + 2)	0.945881548	1.941247341	2.933100235	2.426570201	2.158021202	9.291418389
p2.8xlarge (1 + 5)	1.016142621	2.016799081	2.917491082	3.161749899	2.679823808	8.995971311
p2.16xlarge (1 + 2)	2.128718829	5.913769344	7.321784448	7.863164328	9.854708256	25.78978928
local	0.898913354	2.644115507	2.308397711	1.749879953	1.862544629	7.834551844

Слика 10 Резултати перформансног тестирања тополошке анализе приступом анализе графова

Окружење за спектралну спарсификацију

Окружење

Експериментално окружење се састоји од две конфигурације Амазонове подршке за рачунарство у облаку. Конкретно, тзв. *ElasticMapReduce (EMR)* пружа подршка за инстанце виртуелних машина на којима је Апачи Спарк с пратећим програмима већ инсталиран и спреман за конфигурацију и коришћење. Конфигурације користе инстанце намењене интензивном рачунању и састоје се од једне управљачке машине и више радних машина. Како би се

оценила скалабилност решења, прва конфигурација садржи пет радних машина, док друга има десет.

Случајеви тестирања

За експерименталну евалуацију су изабрана два реална скупа података. Подаци су јавно доступни [55] и представљају графове с тежинама. Први скуп података ("bio-mouse-gene") представља регулативну мрежу мишијих гена добијен анализом генских профила. Састоји се од 45101 чворова и 14506196 грана. Други скуп података ("bio-human-gene2") је сличан и представља генску регулативну мрежу људских гена. Састоји се од 14340 чворова и 9041364 грана. Ови графови су изабрани, јер имају семантичко значење, тј. нису вештачки креирани, и сличност, тј. оба представљају мреже гена. Поред тога, ови графови имају релативно велики број грана, различите густине, расподеле тежина, и топографије.

Битно је нагласити да мреже гена немају никакав специјалан значај за овде представљено решење или његове перформансе. Примера ради, електроенергетске мреже и паметне мреже су такође одлични улазни подаци, нарочито због значаја тежина (које су обично адмитансе) у таквим графовима. Међутим, такве мреже које су јавно доступне, нпр. [56] са око 75000 грана или [57] која има око 145000 грана, су прилично мале за демонстрационе сврхе. Овде је нагласак на могућности решења да оперише у складу с очекивањима која намеће Апачи Спарк, тј. великим скуповима података [58].

Резултати и дискусија

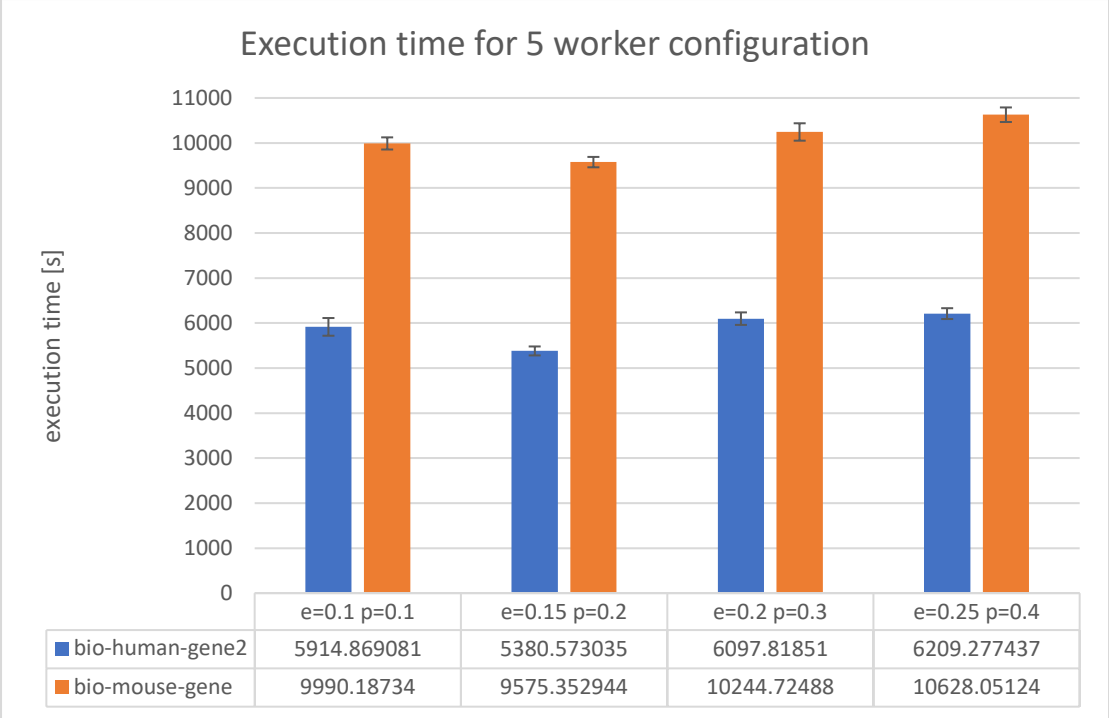
Метода *sparsify* прихвата два улазна параметра, ϵ и p. Како не постоји упутство за избор вредности у циљу производње оптималних резултата, неколико вредности параметара је изабрано, како би се решење било тестирано под различитим условима.

ϵ	0.1	0.15	0.2	0.25
p	0.1	0.2	0.3	0.4

Табела 5 Улазни параметри методе sparsify

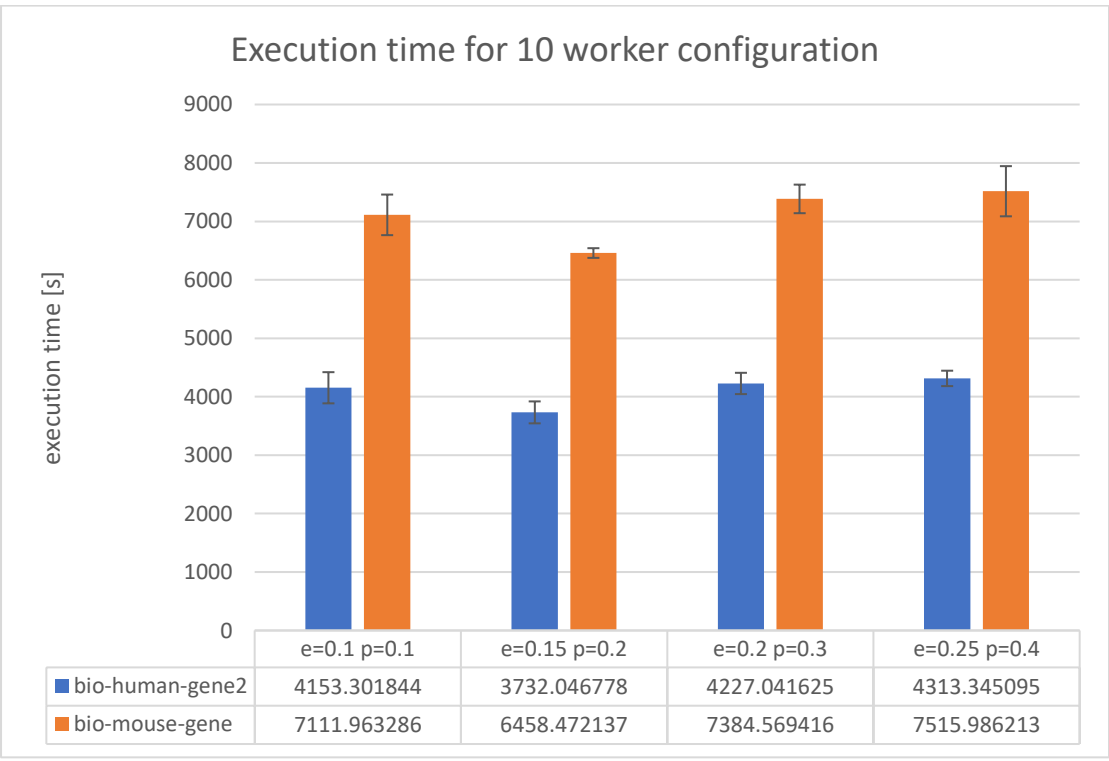
Експериментални поступак подразумева парсирање података у објекат графа и спарсификацију истог. Анализирана су два излазна параметра: време потребно за завршетак поступка спарсификацију (не рачунајући време припреме) и степен спарсификације, тј. колико грана је преостало након поступка. Експерименти су вршени у два окружења, као што је претходно поменуто, са пет и десет радних машина. На свакој су све наведене вредности улазних параметара приказане у Табела 5 истестиране и итериране известан број пута, како би се добила средња вредност за обе излазне величине.

Мерења времена извршавања су показани на Слика 11 и Слика 12. Уочљиво је побољшање перформанси од око 30% при повећању броја радних машина с пет на десет. Имајући у виду да је конфигурација Апачи Спарк окружења и кластера наменски одржавана што униформнијом током експеримената, има још простора за фино подешавање, које би могло донети боље резултате. Констатована девијација се може објаснити двома доминантним факторима. Насумична природа процеса одабирања и дистрибуирано окружење у облаку, које не може да гарантује исте услове при сваком извршавању. Интересантно је приметити да су добици у перформансама добијени изменом вредности параметара конзистентни у различитим итерација, што указује на то да избор вредности параметара пружа још једну могућност финог подешавања, независно од насумичног процеса.



	e=0.1 p=0.1	e=0.15 p=0.2	e=0.2 p=0.3	e=0.25 p=0.4
bio-human-gene2	5914.869081	5380.573035	6097.81851	6209.277437
bio-mouse-gene	9990.18734	9575.352944	10244.72488	10628.05124

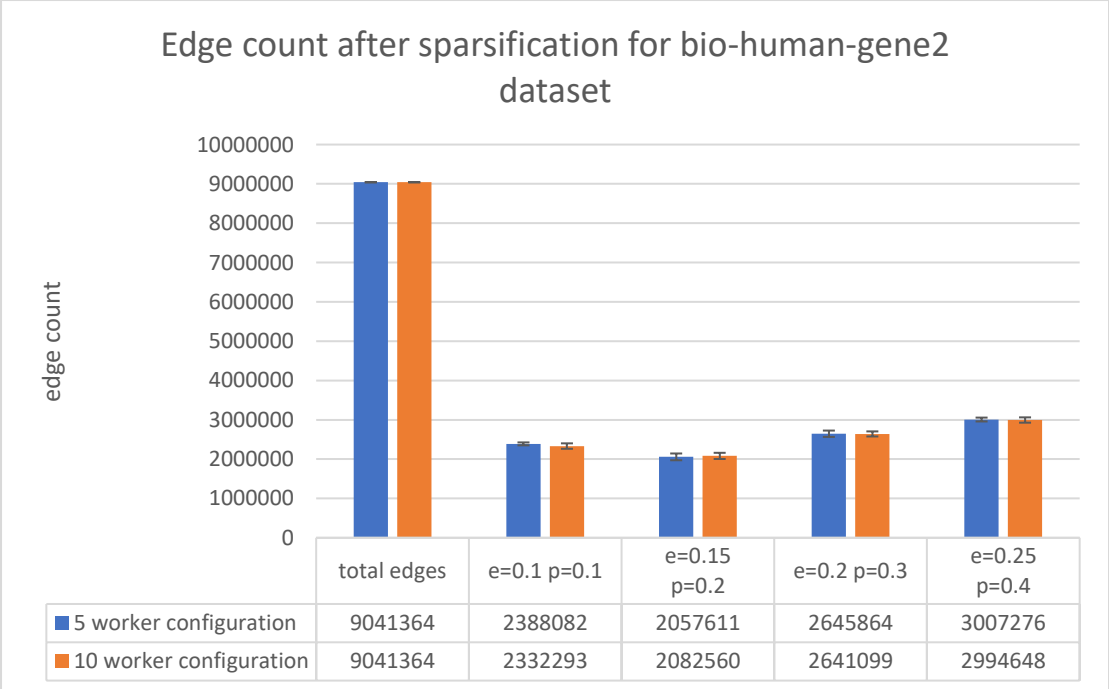
Слика 11 Времена која су била потребна за спарсификацију оба скупа података у окружењу које се састоји од једне управљачке машине и пет радних.



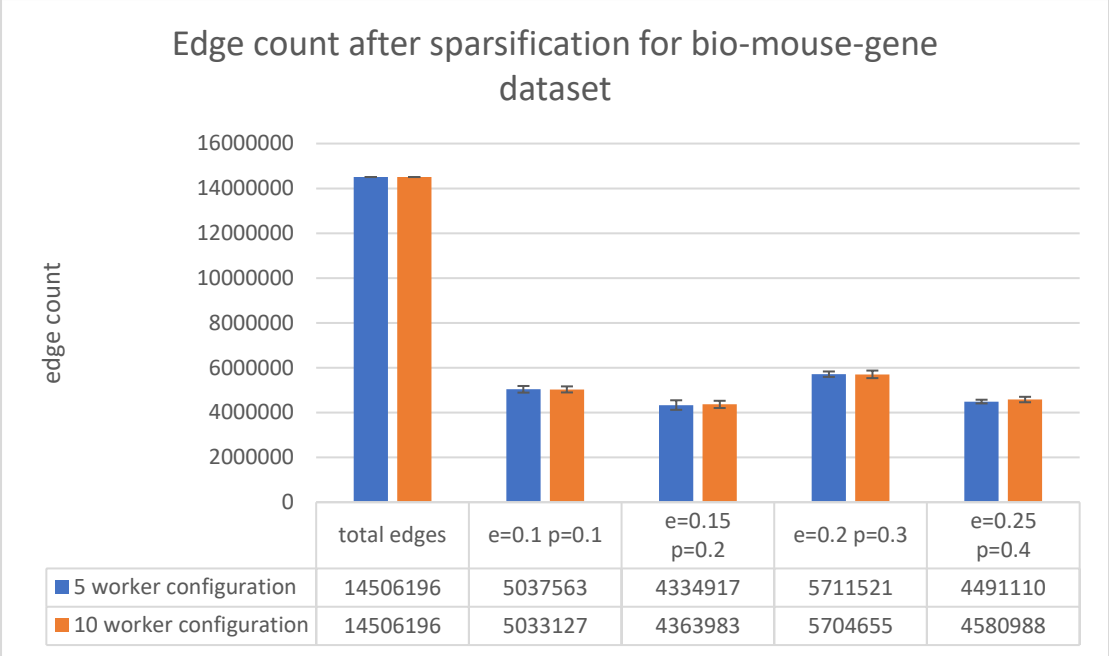
	e=0.1 p=0.1	e=0.15 p=0.2	e=0.2 p=0.3	e=0.25 p=0.4
bio-human-gene2	4153.301844	3732.046778	4227.041625	4313.345095
bio-mouse-gene	7111.963286	6458.472137	7384.569416	7515.986213

Слика 12 Времена која су била потребна за спарсификацију оба скупа података у окружењу које се састоји од једне управљачке машине и десет радних.

Слика 13 и Слика 14 показују достигнути степен спарсификације. Графици пореде број грана добијених након процеса спарсификације с истим варијацијама параметара и окружења као и у претходном случају. Истиче се ред величине спарсификације. У неким случајевима, добијени скуп грана је мањи од 30% од полазног. То значи да је могуће конструисати граф, спектрално сличан оригиналном, са само трећином грана. Опет је уочљива блага девијација у добијеном броју грана. Што значи да се може очекивати другачији број грана при вишеструком извршавању под истим условима. Иако делује изненађујуће, то је још једна последица насумичног процеса. Такође, битно је нагласити да експерименти показују да је та варијација занемарљива у поређењу с бројем грана као и да је резултат процеса спарсификације апроксимација почетног графа. Избор вредности параметара овде има већи утицај него у претходној анализи времена извршавања. Уочљив је неки вид тренда, различит за оба скупа података, ипак, указује на то да постоји минимална конфигурација која би произвела максималну спарсификацију.



Слика 13 Степен спарсификације, тј. преостали број грана након процедуре, за скуп података bio-human-gene2 и за наведене вредности параметара.



	total edges	e=0.1 p=0.1	e=0.15 p=0.2	e=0.2 p=0.3	e=0.25 p=0.4
5 worker configuration	14506196	5037563	4334917	5711521	4491110
10 worker configuration	14506196	5033127	4363983	5704655	4580988

Слика 14 Степен спарсификације, тј. преостали број грана након процедуре, за скуп података bio-mouse-gene и за наведене вредности параметара.

Закључак

Добијени резултати показују да је коришћењем Апачи Спарка за извршавање комплексних математичких операција за анализу стања и симулацију кварова у паметним електроенергетским системима у дистрибуираном окружењу могућа анализа великих мрежа за кратко време. Иако се Апачи Спарк доминантно користи за задатке машинског учења и анализу великих количина података (енгл. big data), овде је приказана његова примењивост на неки други скуп проблема.

Констатовано је неколико недостатака које је потребно разрешити. Најбитнији међу њима је ограничење операције декомпозиције матрице на сингуларне вредности које дефинише максималну димензију матрице, а тиме и величину мреже. Други је недостатак оптимизације Апачи Спарка за извршавање задатка на графичким процесорима.

Спарсификација графова је поуздана процедура за систематско добијање апроксимативног резултата које је довољно добар за подршку оперативним активностима. Овде су демонстриране могућности дистрибуираног рачунарства у комбинацији с моћном методом спектралне анализе графова. Таква комбинације може да унапреди операције и помери границе могућег за аутоматизацију паметних електроенергетских мрежа, телекомуникационих мрежа, предузећа који се баве анализом друштвеним мрежа, итд. Све што је овде приказано је имплементирано ослањајући се на Апачи Спарк. Такође, код је отворен и јавно доступан, што омогућује проширења и репродукцију овде експеримената.

Представљени програмски пакет може бити коришћен у затеченом стању, међутим, спектрална анализа графова често захтева дубље разумевање њених математичких основа. На пример, спектрално кластеровање нуди многе изборе, али њихова интерпретација није тривијална без опширног разумевања

позадине. Ово је тачно у већем или мањем облику и за метода машинског учења.

Даља истраживања је могуће усмерити ка уоченим недостацима имплементације Апачи Спарка. Ово решење тренутно нуди прорачун токова снага и анализу кварова, али постоје многе друге функције за анализу паметних мрежа, нпр. прорачун оптималног тока снага, које би биле унапређене дистрибуираним решењем. Показано је да Апачи Спарк подршка за анализу графова нуди добре перформансе за ову намену. Ипак, његов потенцијал није у потпуности искоришћен у приказаном случају тополошке анализе. Моћни систем размене порука који поседује Апачи Спарк може бити искоришћен у анализи паметних мрежа. Тако би интересантан правац истраживања био доказ да се проблем токова снага може решити аналицом графова уместо уобичајеним путем матричне алгебре.

Постоје многе могућности за побољшања и примене спектралне спарсификације графова. Значајно проширење постојећег програмског пакета може бити развој операција за обраду неодређених графова [59]; графова где је присуство или одсуство грана дефинисано вероватноћом (нпр. Телекомуникационе мреже [60]). Пробабалистички приступ исходима је прихватљиву у случајевима када се доносе закључци на основу непотпуних чињеница, као што је случај у већини комплексних домена.

Литература

[1] D. Šutić и E. Varga, „Spectral Graph Analysis with Apache Spark,“ *Proceedings of the 2018 International Conference on Mathematics and Statistics. ACM,* 2018.

[2] D. A. Spielman и S.-H. Teng, „Spectral sparsification of graphs,“ *SIAM Journal on Computing 40.4,* pp. 981-1025, 2011.

[3] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker и I. Stoica, „Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing,“ *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association,* 2012.

[4] X. Li, G. Tan, C. Zhang, X. Li, Z. Zhang и N. Sun, „Accelerating large-scale genomic analysis with Spark,“ *Bioinformatics and Biomedicine (BIBM), 2016 IEEE International Conference on. IEEE,* 2016.

[5] H. Ji, S. H. Weinberg, M. Li, J. Wang и Y. Li, „An Apache Spark implementation of block power method for computing dominant eigenvalues and eigenvectors of large-scale matrices,“ *Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom), 2016 IEEE International Conferences on. IEEE,* 2016.

[6] F. Porto, A. Khatibi, J. R. Nobre, E. Ogasawara, P. Valduriez и D. Shasha, „Constellation Queries over Big Data,“ *arXiv preprint arXiv:1703.02638,* 2017.

[7] W. Guilan, Z. Guoliang, Z. Hongshan и L. Hongyang, „Real-time big data technologies of energy internet platform,“ *Power System Technology (POWERCON), 2016 IEEE International Conference on. IEEE,* 2016.

[8] J. E. Van Ness, „Iteration methods for digital load flow studies,“ *Transactions of the American Institute of Electrical Engineers. Part III: Power Apparatus and Systems 78.3,* pp. 583-586, 1959.

[9] W. F. Tinney и C. E. Hart, „Power flow solution by Newton's method,“ *IEEE Transactions on Power Apparatus and systems 11,* pp. 1449-1460, 1967.

[10] A. Trias, „The holomorphic embedding load flow method,“ *Power and Energy Society General Meeting, 2012 IEEE. IEEE,* 2012.

[11] F. Goderya, A. A. Metwally и O. Mansour, „Fast detection and identification of islands in power networks,“ *IEEE transactions on power apparatus and systems 1,* pp. 217-221, 1980.

[12] B. Stott, O. Alsac и A. J. Monticelli, „Security analysis and optimization,“ *Proceedings of the IEEE 75.12,* pp. 1623-1644, 1987.

[13] M. Montagna и G. P. Granelli, „Detection of Jacobian singularity and network islanding in power flow computations,“ *IEE Proceedings-Generation, Transmission and Distribution 142.6,* pp. 589-594, 1995.

[14] T. Guler и G. Gross, „Detection of island formation and identification of causal factors under multiple line outages,“ *IEEE Transactions on Power Systems 22.2,* pp. 505-513, 2007.

[15] R. D. Zimmerman, C. E. Murillo-Sanchez и R. J. Thomas, „MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education,“ *IEEE Transactions on power systems 26.1,* pp. 12-19, 2011.

[16] J. Beerten и R. Belmans, „Development of an open source power flow software for high voltage direct current grids and hybrid AC/DC systems: MATACDC,“ *IET Generation, Transmission & Distribution 9.10,* pp. 966-974, 2015.

[17] H. Li, J. Sun и L. Tesfatsion, „Dynamic LMP response under alternative price-cap and price-sensitive demand scenarios,“ *Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE,* 2008.

[18] M. Zhou и S. Zhou, „Internet, open-source and power system simulation,“ *2007 IEEE Power Engineering Society General Meeting,* pp. 1-5, 2007.

[19] D. A. Spielman и S.-H. Teng, „A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning,“ *arXiv preprint arXiv:0809.3232,* 2008.

[20] D. A. Spielman и S.-H. Teng, „Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems,“ *SIAM Journal on Matrix Analysis and Applications,* pp. 835-885, 2014.

[21] D. A. Spielman и N. Srivastava, „Graph sparsification by effective resistances,“ *SIAM Journal on Computing 40.6,* pp. 1913-1926, 2011.

[22] I. Koutis и S. C. Xu, „Simple parallel and distributed algorithms for spectral graph sparsification,“ *ACM Transactions on Parallel Computing (TOPC),* 2016.

[23] H. Sun и L. Zanetti, „Distributed graph clustering and sparsification,“ *ACM Transactions on Parallel Computing (TOPC) 6.3,* 2019.

[24] J. E. Gonzales, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin и I. Stoica, „Graphx: Graph processing in a distributed dataflow framework,“ *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14),* 2014.

[25] R. S. Xin, J. Gonzales, M. J. Franklin и I. Stoica, „Graphx: A resilient distributed graph system on spark,“ *First International Workshop on Graph Data Management Experiences and Systems. ACM,* 2013.

[26] X. Zhao, Z. Feng и C. Zhuo, „An efficient spectral graph sparsification approach to scalable reduction of large flip-chip power grids,“ *Proceedings of the 2014 IEEE/ACM*

International Conference on Computer-Aided Design. IEEE Press, 2014.

[27] S. Ghemawat, H. Gobioff и S.-T. Leung, „The Google file system,“ *Proceedings of the nineteenth ACM symposium on Operating systems principles,* 2003.

[28] J. Dean и S. Ghemawat, „MapReduce: simplified data processing on large clusters.,“ *Communications of the ACM 51.1,* pp. 107-113, 2008.

[29] K. Shvachko, H. Kuang, S. Radia и R. Chansler, „The hadoop distributed file system,“ *2010 IEEE 26th symposium on mass storage systems and technologies (MSST),* pp. 1-10, 2010.

[30] A. Lakshman и P. Malik, „Cassandra: a decentralized structured storage system,“ *ACM SIGOPS Operating Systems Review 44, no. 2,* pp. 35-40, 2010.

[31] V. Persico, A. Montieri и A. Pescape, „On the network performance of amazon S3 cloud-storage service,“ *2016 5th IEEE International Conference on Cloud Networking (Cloudnet),* pp. 113-118, 2016.

[32] J. J. Grainger и W. D. Stevenson, Power system analysis, McGraw-Hill, 1994.

[33] I. A. Hiskens, Power Flow Analysis, Department of Electrical and Computer Engineering, University of Wisconsin-Madison, 2003.

[34] J. Batson, D. A. Spielman, N. Srivastava и S.-H. Teng, „Spectral sparsification of graphs: theory and algorithms,“ *Communications of the ACM 56.8,* pp. 87-94, 2013.

[35] D. Šutić, „SpectralGraphAnalysisTool Repository,“ 2018. [На мрежи]. Available: https://bitbucket.org/suticd/spectralgraphanalysistool/src/master/. [Последњи приступ 2021].

[36] R. Bosagh Zadeh, X. Meng, A. Ulanov, B. Yavuz, L. Pu, S. Venkataraman, E. Sparks, A. Staple и M. Zaharia, „Matrix computations and optimization in apache spark,“ *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM,* 2016.

[37] G. Malewisz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser и G. Czajkowski, „Pregel: a system for large-scale graph processing,“ *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM,* 2010.

[38] J. Dittrich и J.-A. Quiané-Ruiz, „Efficient big data processing in Hadoop MapReduce,“ *Proceedings of the VLDB Endowment 5.12,* pp. 2014-2015, 2012.

[39] A. Naor, „Every graph is essentially sparse,“ *Communications of the ACM 56, no. 8,* pp. 86-86, 2013.

[40] E. A. Dinic, „Algorithm for solution of a problem of maximum flow in a network with power estimation,“ *soviet math. doll. 11,* т. English translation by RF. Rinehart (1970), pp. 1277-1280, 1970.

[41] J. Edmonds и R. M. Karp, „Theoretical improvements in algorithmic efficiency for

network flow problems,“ *Journal of the ACM (JACM) 19.2,* pp. 248-264, 1972.

[42] J. Zhang, „GraphX Connected Components fail with large number of iterations,“ JIRA, 2015. [На мрежи]. Available: https://issues.apache.org/jira/browse/SPARK-10335. [Последњи приступ 2018].

[43] D. A. Spielman и S.-H. Teng, „A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning,“ *SIAM Journal on Computing,* pp. 1-26, 2013.

[44] Apache Spark, „IBMSparkGPU/SparkGPU,“ GitHub, 2016. [На мрежи]. Available: https://github.com/IBMSparkGPU/SparkGPU. [Последњи приступ 2018].

[45] Databricks, „GPU-enabled clusters,“ Databricks, 2018. [На мрежи]. Available: https://docs.databricks.com/user-guide/clusters/gpu.html. [Последњи приступ 2018].

[46] E. Jonas, Q. Pu, S. Venkataraman, I. Stoica и B. Recht, „Occupy the Cloud: Distributed Computing for the 99%,“ *arXiv preprint arXiv:1702.04024,* 2017.

[47] G. Wang, J. Xu и B. He, „A Novel Method for Tuning Configuration Parameters of Spark Based on Machine Learning,“ *High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016 IEEE 18th International Conference on. IEEE,* 2016.

[48] F. Li и R. Bo, „Small test systems for power system economic studies,“ *Power and Energy Society General Meeting, IEEE,* 2010.

[49] A. J. Wood и B. Wollenberg, „Power generation operation and control—2nd edition,“ *Fuel and Energy Abstracts. Vol. 37. No. 3. Elsevier,* 1996.

[50] University of Washington, College of Engineering, „Power Systems Test Case Archive,“ University of Washington, 2006. [На мрежи]. Available: https://labs.ece.uw.edu/pstca/. [Последњи приступ 2020].

[51] H. C. Grigg, P. Wong, P. Albrecht, R. Allan, M. Bhavaraju, R. Billinton, Q. Chen, C. Fong, S. Haddad, S. Kuruganty, W. Li, R. Mukerji, D. Patton, N. Rau, D. Reppen, A. Schneider, M. Shahidehpour и C. Singh, „The IEEE reliability test system-1996. A report prepared by the reliability test system task force of the application of probability methods subcommittee,“ *IEEE Transactions on power systems 14.3,* pp. 1010-1020, 1999.

[52] 7th Framework Program of the European Union, „Pan European Grid Advanced Simulation and State Estimation,“ 2018. [На мрежи]. Available: http://www.fp7-pegase.com/. [Последњи приступ 2018].

[53] C. Josz, S. Fliscounakis, J. Maeght и P. Panciatici, „AC power flow data in MATPOWER and QCQP format: iTesla, RTE snapshots, and PEGASE,“ *arXiv preprint arXiv:1603.01533,* 2016.

[54] V. Jancauskas, Scientific Computing with Scala, Packt Publishing Ltd, 2016.

[55] R. Rossi и N. Ahmed, „The network data repository with interactive graph analytics and visualization,“ *Twenty-Ninth AAAI Conference on Artificial Intelligence,* 2015.

[56] F. D. Freitas, J. Rommes и N. Martins, „Gramian-based reduction method applied to large sparse power system descriptor models,“ *IEEE Transactions on Power Systems 23.3,* pp. 1258-1270, 2008.

[57] J. Rommes, N. Martins и F. D. Freitas, „Computing rightmost eigenvalues for small-signal stability assessment of large-scale power systems,“ *IEEE transactions on power systems 25.2,* pp. 929-938, 2009.

[58] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzales, S. Shenker и I. Stoica, „Apache spark: a unified engine for big data processing,“ *Communications of the ACM 59.11,* pp. 56-65, 2016.

[59] P. Parchas, N. Papailiou, D. Papadias и F. Bonchi, „Uncertain graph sparsification,“ *IEEE Transactions on Knowledge and Data Engineering 30.12,* pp. 2435-2449, 2018.

[60] A. Kaveh, M. Magnani и C. Rohner, „Comparing node degrees in probabilistic networks,“ *Journal of Complex Networks,* 2019.

[61] M. Lee, O. Aslam, B. Foster, D. Kathan, J. Kwok, L. Medearis, R. Palmer, P. Sporborg и M. TIta, „Assessment of demand response and advanced metering,“ *Federal Energy Regulatory Commission, Tech. Rep,* 2013.

[62] L. Chendan, S. K. Chaudhary, M. Savaghebi, J. C. Vacquez и J. M. Guerrero, „Power flow analysis for low-voltage AC and DC microgrids considering droop control and virtual impedance,“ *IEEE Transactions on Smart Grid,* 2016.

[63] Y. P. Hong и C.-T. Pan, „Rank-revealing QR factorizations and the singular value decomposition,“ *Mathematics of Computation 58.197,* pp. 213-232, 1992.

План третмана података

Назив пројекта/истраживања
Одабрани алгоритми теорије графова и линеарне алгебре прилагођени великим количинама података
Назив институције/институција у оквиру којих се спроводи истраживање
Универзитет у Новом Саду, Факултет техничких наука, Департман за енергетику, електронику и телекомуникације
Назив програма у оквиру ког се реализује истраживање
Докторске академске студије - Енергетика, електроника и телекомуникације
1. Опис података

1.1 Врста студије

Укратко описати тип студије у оквиру које се подаци прикупљају
Циљ истраживања је развој напредних алгоритама примењивих на велике количине података из области теорије графова и линеарне алгебре. Експериментална верификација добијених поступака се верификује на одабраним јавно доступним скуповима података који служе као улаз. Резултати експеримената, односно излаз алгоритама, су перформансе извршавања, т.ј. време извршавања као и постигнути резултати израчунавања.

1.2 Врсте података
а) квантитативни
б) квалитативни

1.3. Начин прикупљања података
а) анкете, упитници, тестови
б) клиничке процене, медицински записи, електронски здравствени записи
в) генотипови: навести врсту _____
г) административни подаци: навести врсту _____
д) узорци ткива: навести врсту_____
ђ) снимци, фотографије: навести врсту_____
е) текст, навести врсту _____
ж) мапа, навести врсту _____
з) остало: јавно доступни подаци који описују реалне електроенергетске мреже и скупове података који се могу представити графовима из домена генетског истраживања

1.3 Формат података, употребљене скале, количина података

1.3.1 Употребљени софтвер и формат датотеке:
а) Excel фајл, датотека _____
b) SPSS фајл, датотека _____
c) PDF фајл, датотека _____
d) Текст фајл, датотека : .csv
e) JPG фајл, датотека _____
f) Остало, датотека _____

1.3.2. Број записа (код квантитативних података)

а) број варијабли _____
б) број мерења (испитаника, процена, снимака и сл.) _____

1.3.3. Поновљена мерења
а) да
б) не

Уколико је одговор да, одговорити на следећа питања:
а) временски размак између поновљених мера је у просеку 1-30 минута, у зависности од претходних итерација извршавања
б) варијабле које се више пута мере односе се на време извршавања и добијење резултате
в) нове верзије фајлова који садрже поновљена мерења су именоване као _____

Напомене: _____

Да ли формати и софтвер омогућавају дељење и дугорочну валидност података?
а) Да
б) Не
Ако је одговор не, образложити _____
_____

2. Прикупљање података

2.1 Методологија за прикупљање/генерисање података

2.1.1. У оквиру ког истраживачког нацрта су подаци прикупљени?
а) експеримент, навести тип _____
б) корелационо истраживање, навести тип _____
ц) анализа текста, навести тип _____
д) остало, навести шта – Подаци за евалуацију су јавно доступни. Коришћени су модели електроенергетских мрежа које обухватају ситуације експерименталног и наменског порекла, као и реалне случајеве коришћења. Поред тога, коришћена су још два скупа података који представљају људску и мишију регулативну мрежу гена.

2.1.2 Навести врсте мерних инструмената или стандарде података специфичних за одређену научну дисциплину (ако постоје).
_____

_____

2.2 Квалитет података и стандарди

2.2.1. Третман недостајућих података
а) Да ли матрица садржи недостајуће податке? Да Не

Ако је одговор да, одговорити на следећа питања:
а) Колики је број недостајућих података? _____
б) Да ли се кориснику матрице препоручује замена недостајућих података? Да Не
в) Ако је одговор да, навести сугестије за третман замене недостајућих података
_____

2.2.2. На који начин је контролисан квалитет података? Описати

_____

_____

2.2.3. На који начин је извршена контрола уноса података у матрицу?

_____

_____

3. Третман података и пратећа документација

3.1. Третман и чување података

3.1.1. Подаци ће бити депоновани у _____ репозиторијум.
3.1.2. URL адреса :

1. https://labs.ece.uw.edu/pstca
2. *https://networkrepository.com/*

3.1.3. DOI _____

3.1.4. Да ли ће подаци бити у отвореном приступу?
*а) **Да***
б) Да, али после ембарга који ће трајати до _____
в) Не

Ако је одговор не, навести разлог _____

3.1.5. Подаци неће бити депоновани у репозиторијум, али ће бити чувани.
Образложење

_____

_____

3.2 Метаподаци и документација података
3.2.1. Који стандард за метаподатке ће бити примењен? _____

3.2.1. Навести метаподатке на основу којих су подаци депоновани у репозиторијум.

_____

_____

Ако је потребно, навести методе које се користе за преузимање података, аналитичке и процедуралне информације, њихово кодирање, детаљне описе варијабли, записа итд.

_____

_____

_____

_____

3.3 Стратегија и стандарди за чување података
3.3.1. До ког периода ће подаци бити чувани у репозиторијуму? _____
3.3.2. Да ли ће подаци бити депоновани под шифром? Да Не
3.3.3. Да ли ће шифра бити доступна одређеном кругу истраживача? Да Не
3.3.4. Да ли се подаци морају уклонити из отвореног приступа после извесног времена?
Да Не
Образложити

_____

_____

4. Безбедност података и заштита поверљивих информација

Овај одељак МОРА бити попуњен ако ваши подаци укључују личне податке који се односе на учеснике у истраживању. За друга истраживања треба такође размотрити заштиту и сигурност података.
4.1 Формални стандарди за сигурност информација/података
Истраживачи који спроводе испитивања с људима морају да се придржавају Закона о заштити података о личности *(https://www.paragraf.rs/propisi/zakon_o_zastiti_podataka_o_licnosti.html)* и одговарајућег институционалног кодекса о академском интегритету.

4.1.2. Да ли је истраживање одобрено од стране етичке комисије? Да Не
Ако је одговор Да, навести датум и назив етичке комисије која је одобрила истраживање

_____

4.1.2. Да ли подаци укључују личне податке учесника у истраживању? Да **Не**
Ако је одговор да, наведите на који начин сте осигурали поверљивост и сигурност информација везаних за испитанике:
а) Подаци нису у отвореном приступу
б) Подаци су анонимизирани
ц) Остало, навести шта

_____
_____

5. Доступност података

5.1. Подаци ће бити
а) јавно доступни
б) доступни само уском кругу истраживача у одређеној научној области
ц) затворени

Ако су подаци доступни само уском кругу истраживача, навести под којим условима могу да их користе:

_____

_____

Ако су подаци доступни само уском кругу истраживача, навести на који начин могу приступити подацима:

_____

_____

5.4. Навести лиценцу под којом ће прикупљени подаци бити архивирани.

_____

6. Улоге и одговорност

6.1. Навести име и презиме и мејл адресу власника (аутора) података

_____

6.2. Навести име и презиме и мејл адресу особе која одржава матрицу с подацима

_____

6.3. Навести име и презиме и мејл адресу особе која омогућује приступ подацима другим истраживачима

_____