УНИВЕРЗИТЕТ У БЕОГРАДУ

ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА

Невена З. Миленковић

РАЗВОЈ АЛГОРИТАМА МАШИНСКОГ УЧЕЊА ЗА ПЕРИОДИЧНА УНУТАР-СЕКВЕНЦИЈАЛНА ПРАВИЛА И ДИНАМИЧКУ КОРИСНОСТ СТАВКИ

докторска дисертација

Београд, 2026

UNIVERSITY OF BELGRADE

FACULTY OF ORGANISATIONAL SCIENCES

Nevena Z. Milenković

DEVELOPMENT OF MACHINE LEARNING ALGORITHMS FOR PERIODIC INTRA-SEQUENTIAL RULES AND DYNAMIC ITEM UTILITY

Doctoral Dissertation

Belgrade, 2026

Ментор:

Др Борис Делибашић,
ред. проф. Факултета организационих наука у Београду

Чланови комисије:
Проф. др Милија Сукновић,
редовни професор, Факултет организационих наука, Универзитет у Београду
Проф. др Милош Јовановић,
ванредни професор, Факултет организационих наука, Универзитет у Београду
Проф. др Душан Старчевић,
професор емеритус, Факултет организационих наука, Универзитет у Београду
Проф. др Мирослав Марић,
редовни професор, Математички факултет, Универзитет у Београду

Датум одбране: _____

РАЗВОЈ АЛГОРИТАМА МАШИНСКОГ УЧЕЊА ЗА ПЕРИОДИЧНА УНУТАР-СЕКВЕНИЈАЛНА ПРАВИЛА И ДИНАМИЧКУ КОРИСНОСТ СТАВКИ

Сажетак

Ова дисертација представља синтезу научних доприноса у две значајне области откривања образаца: у секвенцијалним и трансакционим подацима, са посебним фокусом на унапређење метода за проналажење секвенцијалних правила и скупова ставки високе корисности.

У оквиру анализе секвенцијалних података, уводи се нови задатак откривања периодичних унутар-секвенцијалних правила заједничких за више секвенци. Поред увођења ограничења периодичности у домен секвенцијалних правила, формализована је и обрада унутар-секвенцијалних правила, чиме је омогућено откривање узрочно-временских односа који се периодично понављају унутар и између секвенци. Предложен је алгоритам PISRuleGrowth, заснован на приступу раста образаца и ефикасној структури мапа, који постиже боље или упоредиве резултате у погледу времена извршавања и потрошње меморије у односу на постојеће методе.

Истраживање додатно превазилази она ограничења традиционалних метода за откривање скупова ставки високе корисности која нису пројектована за адекватну обраду података са динамичким променама профитабилности. Дефинисан је концепт мешовите корисности и развијен алгоритам MEHAUPM за откривање скупова ставки високо-просечне корисности у окружењима са променљивим позитивним и негативним вредностима корисности. Резултати показују супериорност у перформансама, тачности и квалитету резултата, уз већу просечну и укупну корисност генерисаних скупова ставки.

Рад доприноси развоју алгоритама прилагођених динамичким и реалним пословним подацима, са значајном применом у анализи продаје и подршци одлучивању.

Кључне речи: унутар-секвенцијална правила, периодична секвенцијална правила, делимично уређена секвенцијална правила, откривање правила у секвенцама, откривање образаца у секвенцама, мешовита корисност, скупови ставки високо-просечне корисности, скупови ставки са динамичком променом корисности, откривање образаца у трансакцијама, откривање образаца у подацима

Научна област: Рачунарске науке – Машинско учење

Ужа научна област: Откривање образаца у подацима

DEVELOPMENT OF MACHINE LEARNING ALGORITHMS FOR PERIODIC INTRA-SEQUENTIAL RULES AND DYNAMIC ITEM UTILITY

Abstract

This dissertation represents a synthesis of scientific contributions in two important areas of pattern discovery: sequential and transactional data analysis, with a particular focus on improving methods for mining sequential rules and high-utility itemsets.

Within the analysis of sequential data, a new task of mining periodic intra-sequential rules common to multiple sequences is introduced. In addition to incorporating the periodicity constraint into the domain of sequential rules, the processing of intra-sequential rules is formally defined, enabling the discovery of causal–temporal relationships that recur periodically within and across sequences. The proposed PISRuleGrowth algorithm, based on a pattern-growth approach and an efficient map data structure, achieves superior or comparable performance in terms of execution time and memory consumption when compared to existing methods.

The research further overcomes the limitations of traditional high-utility itemset mining methods that are not designed to adequately handle data with dynamically changing profitability. The concept of mixed utility is formally defined, and the MEHAUPM algorithm is developed for mining high-average utility itemsets in environments with fluctuating positive and negative utility values. The results demonstrate superiority in performance, accuracy, and quality of outcomes, achieving higher average and total utility of the generated itemsets.

Overall, this work contributes to the development of algorithms tailored to dynamic and real-world business data, with significant applications in sales analysis and decision support.

Keywords: intra-sequential rules, periodic sequential rules, partially ordered sequential rules, sequential rule mining, sequence mining, mixed utility, high-average utility itemsets, dynamic changing utility itemsets, transaction mining, data mining

Scientific Field: Computer Science – Machine Learning

Narrower Scientific Field: Data mining

Садржај

Садржај 6
1. Увод 8
1.1. Предмет и циљ истраживања 10
1.2. Полазне хипотезе 12
1.3. Научне методе истраживања 12
1.4. Структура и организација рада 13
2. Теоријски оквир и сродне методе откривања образаца 13
2.1. Теоријски оквир области откривања секвенцијалних правила и периодичних образаца 14
2.2. Теоријски оквир области откривања скупова ставки високе корисности 15
3. Основни појмови и формулисање проблема истраживања 18
3.1. Дефинисање проблема периодичних унутар-секвенцијалних правила 18
3.2. Дефинисање проблема скупова ставки високо-просечне корисности са мешовитим вредностима корисности 22
4. Развој и имплементација предложених алгоритама 24
4.1. PISRuleGrowth: архитектура и логика алгоритма 24
4.1.1. Скенирање базе података 26
4.1.2. Главна процедура 28
4.1.3. ExpandLeft 29
4.1.4. ExpandRight 32
4.1.5. Метода get_Single_Item_Rule_Occur 32
4.1.6. Метода is_PIS_Rule 33
4.1.7. Метода getNextIC 34
4.1.8. Метода countIC 34
4.1.9. Метода getRuleOccur 35
Procedure 4: getRuleOccur procedure's pseudocode 35
4.2. MEHAUPM: Модификације и обрада мешовитих вредности 36
5. Експериментална валидација и анализа резултата 42
5.1. Експериментална евалуација PISRuleGrowth алгоритма 42
5.1.1. Поређење перформанси алгоритма PISRuleGrowth са другим алгоритмима за генерисање секвенцијалних правила при истим вредностима параметара 44

5.1.2. Поређење перформанси PISRuleGrowth и других алгоритама за генерисање секвенцијалних правила при генерисању истог броја правила 45
5.1.3. Поређење перформанси PISRuleGrowth алгоритма и његове неоптимизоване верзије при варирању параметра minSup 47
5.1.4. Поређење перформанси PISRuleGrowth алгоритма и његове неоптимизоване верзије при варирању параметра minConf 49
5.1.5. Поређење перформанси PISRuleGrowth алгоритма и његове неоптимизоване верзије при варирању параметра maxStd 51
5.1.6. Поређење перформанси PISRuleGrowth алгоритма и његове неоптимизоване верзије при варирању параметра minRa 52
5.1.7. Поређење перформанси PISRuleGrowth алгоритма и његове неоптимизоване верзије при генерисању истог броја правила 54
5.2. Експериментална евалуација MEHAUPM алгоритма 55
5.2.1. Поређење перформанси алгоритма MEHAUPM и других алгоритама високе корисности који обрађују негативне вредности корисности при истим вредностима параметара корисности 57
5.2.2. Предности алгоритма MEHAUPM у односу на друге алгоритме високе корисности који обрађују негативне вредности корисности при истим вредностима параметара корисности 60
5.2.3. Предности алгоритма MEHAUPM у односу на стандардне алгоритме за откривање скупова ставки високо-просечне корисности 67
6. Закључак 73
Референце 76
Пратеће изјаве 81

1. Увод

Савремени дигитални системи, апликације и уређаји производе огромне количине хетерогених, динамичких и комплексних података. Овај експоненцијални раст поставља високе захтеве пред системе за њихову анализу, захтевајући континуирани развој напредних техника откривања образаца. Међу тим техникама посебно значајно место заузимају две истраживачке области: откривање скупова ставки високе корисности (High-Utility Itemset Mining - HUIM) и откривање секвенцијалних образаца (Sequential Pattern Mining - SPM) (Gan et al., 2018a; Agrawal & Srikant, 1995). Оба домена имају заједничку мотивацију, односно идентификацију најзначајнијих образаца у базама података, али полазе од различитих концептуалних оквира и решавају различите проблеме. Њихов историјски развој показује сличан ток, јер су се основни алгоритми временом суочили са ограничењима у погледу ефикасности, изражајности и применљивости у реалним сценаријима. Такви изазови подстакли су развој специјализованих техника и концепата чији је главни циљ унапређење тачности, репрезентативности и практичне вредности добијених образаца.

Откривање честих скупова ставки (Frequent Itemset Mining - FIM) представља једну од најранијих и највише истражених тема у области откривања образаца у подацима. Циљ овог приступа је идентификовање скупова ставки које се често јављају заједно у трансакционим базама података (Han et al., 2007). Ипак, овај приступ је ограничен чињеницом да учесталост појављивања скупа ставки не мора одражавати његову стварну вредност за корисника или пословни систем. Када се у разматрање укључе количине или профит по артиклу, традиционалне технике губе значај у примени.

Управо ова ограничења довела су до развоја откривања скупова ставки високе корисности (High-utility Itemset Mining - HUIM) (Gan et al., 2018a). HUIM омогућава анализу трансакција узимајући у обзир не само присуство ставки, већ и њихову корисност, која може представљати профит, тежину, количину или било коју другу мерљиву вредност (Fournier-Viger et al., 2019a). Скуп ставки сматра се високо корисним уколико његова укупна корисност премашује праг минималне корисности (minUtil), што га чини значајним за реалне примене у различитим доменима.

HUIM је нашао широку примену у области малопродаје, за препознавање најпрофитабилнијих комбинација производа, оптимизацију распореда производа на полицама, анализу потрошачких навика и креирање маркетиншких стратегија (Ismail et al., 2017; Manike & Om, 2014; Bapna et al., 2020). Поред малопродаје, користи се и у предвиђању образаца (Patil & Bojewar, 2017), анализи документације о патентима (Altuntas & Gök, 2021), обради веб-логова (Ahmed et al., 2010) и бројним другим областима где значај одређених комбинација ставки није искључиво условљен њиховом фреквенцијом.

Упркос својој применљивости, HUIM се суочава са важним ограничењем: корисност скупа ставки расте са дужином скупа, јер је укупна корисност дефинисана као збир корисности свих ставки (sumUtil). Ова корелација између дужине скупа ставки и корисности доводи до фаворизовања дужих скупова, што неретко резултира неуравнотеженом, неправичном или недовољно корисном анализом.

Ова ограничења су иницирала развој новог истраживачког правца под називом откривање скупова ставки високо-просечне корисности (High-Average Utility Itemset Mining – HAUIM) (Lin et al., 2016). Овај приступ подразумева нормализацију корисности дужином скупа ради добијања прецизније процене значаја његових елемената. Примена просечне вредности уместо кумулативне суме корисности омогућава идентификацију образаца који пружају објективнији увид у стварни значај ставки у бази.

Имплементација овог концепта довела је до развоја низа алгоритама, почевши од HAUP-Tree, након чега се развијају напредније варијанте попут HAUI-Tree (Lin et al., 2010) и HAUI-Miner (Lin et al., 2016). Тренутно најзначајније резултате у погледу брзине извршавања и уштеде меморије постиже EHAUPM (Lin et al., 2017a), који се ослања на оптимизовану архитектуру засновану на стаблу и формирање матрице процењене просечне корисности заједничког појављивања ставки.

Додатни изазов за традиционалне HUIM методе представља претпоставка да све ставке поседују искључиво позитивну корисност. У стварним тржишним околностима негативна корисност је учестала појава, будући да се одређени артикли намерно продају са губитком ради стимулације продаје комплементарних производа. Због широке примене овог концепта у малопродаји, постало је неопходно развити алгоритме који могу поуздано да обрађују податке са негативним вредностима корисности.

Иако је развијено више алгоритама за ову намену (Chu et al., 2009; Fournier-Viger & Zida, 2015; Fournier-Viger, 2014), већина њих генерише недовољно прецизне резултате. Такви недостаци долазе до изражаја у случајевима када скуп ставки садржи ставке мешовите корисности или ставке које се јављају у трансакцијама са негативном укупном корисношћу. Наведени проблеми указују на постојање значајног истраживачког простора који захтева нове, тачније и робусније алгоритме.

Други истраживачки правац обухваћен овом дисертацијом односи се на откривање секвенцијалних образаца (Sequential Pattern Mining – SPM). За разлику од откривања скупова ставки, овај приступ узима у обзир редослед догађаја, што га чини посебно примењивим у анализи ДНК секвенци, клик-стримова, локацијских података и различитих временских серија. Циљ SPM-а је идентификација уређених подскупова који се појављују у најмање онолико секвенци колико је дефинисано прагом подршке (Garofalakis et al., 2002).

Недостатак SPM метода огледа се у занемаривању корелација између ставки унутар појединачних трансакција. Насупрот томе, асоцијативна правила успешно идентификују ове међузависности, али не узимају у обзир временски редослед догађаја. Због ове специфичности, асоцијативна правила се често примењују у областима попут предвиђања епидемија (Cesario, 2023), при чему се квалитет резултата додатно унапређује техникама претпроцесирања, као што је кластеровање података (Liu et al., 2021, Cesario et al., 2005).

Како би се огроман број генерисаних образаца свео на мањи, информативнији скуп, током развоја SPM метода уведена су различита ограничења, попут концепта затворених (closed) (Yan et al., 2003), топ-к (top-k) (Tzvetkov et al., 2005) и максималних (maximal) образаца (Fournier-Viger et al., 2013). Ипак, даља потреба за бољом интерпретабилношћу резултата довела је до увођења концепта периодичности, дефинисаног као појављивање шаблона у правилним временским интервалима (Tanbeer et al., 2009).

Овакав приступ омогућио је идентификацију образаца који се појављују циклично, што је од изузетног значаја у доменима где феномени имају изражен понављајући карактер. Примена периодичности је касније значајно диверсификована, што је довело до развоја нових начина израчунавања (Fournier-Viger et al., 2017), интеграције са другим типовима образаца (Fournier-Viger et al., 2016) и комбиновања са додатним ограничењима (Fournier-Viger et al., 2021).

Развој концепта периодичности обухватио је и увођење унутар-секвенцијалних образаца (Fournier-Viger et al., 2019b), који се циклично јављају унутар појединачних секвенци, али истовремено представљају заједничке моделе понашања присутне у већем броју различитих секвенци. На овај начин омогућено је откривање суптилних, понављајућих образаца који би остали невидљиви уколико би се свака секвенца анализирала изоловано. Овакав приступ омогућава препознавање постојаних и шире применљивих образаца који нису карактеристични

само за појединачне примере, већ указују на опште законитости и трендове скривене у подацима.

Упркос значајном напретку у области, постојеће технике откривања секвенцијалних образаца и асоцијативних правила и даље имају бројна ограничења. У настојању да се превазиђу слабости оба концепта, развијена су секвенцијална правила (Sequential Rules) (Mannila et al., 1997). Она обухватају и флексибилнију варијанту делимично уређених секвенцијалних правила. Секвенцијална правила садрже информације о редоследу догађаја и корелацијама међу ставкама. На тај начин омогућено је откривање информативнијих, интерпретабилнијих и потенцијално узрочних образаца у секвенцијалним подацима.

Захваљујући својој информативности, секвенцијална правила примењују се у бројним реалним доменима: препоруци веб-страница (Singh et al., 2017), откривању финансијских превара (Singh et al., 2020), препорукама локација (Amirat et al., 2018) и разним задацима предвиђања (Sagare et al., 2020; Nguyen et al., 2018; Fahed et al., 2020).

Упркос томе, алгоритми за генерисање правила (Fournier-Viger et al., 2011; Pei et al., 2004; Fournier-Viger et al., 2014a; Fournier-Viger et al., 2012) и даље се суочавају са проблемом великог броја генерисаних правила и ограниченим могућностима за процену њихове стварне значајности. Овај недостатак је посебно изражен у контексту периодичног и рекурентног понашања. Концепти периодичности и рекурентности већ су успешно примењени у алгоритмима за откривање секвенцијалних образаца (Fournier-Viger et al., 2019b; Fournier-Viger et al., 2019c). Међутим, њихова примена у откривању правила и даље је ограничена, иако би могла значајно да унапреди анализу понашања корисника, потрошачких навика, финансијских токова и других динамичних система.

Рекурентност је нарочито значајна јер омогућава праћење учесталости понављања одређеног правила унутар секвенци истог корисника. Овај аспект је суштински важан у областима као што су анализа куповних навика, маркетинг и израда персонализованих препорука. Увођење рекурентности у поступак откривања правила допринело би значајном смањењу броја генерисаних правила, уз истовремено повећање њихове важности и употребне вредности.

Обе области, откривање скупова ставки високе корисности и откривање секвенцијалних правила, показују сличну еволуцију. Иницијалне технике пружале су корисне, али ограничене резултате. Временом је то довело до развоја напреднијих техника које уважавају комплексност реалних података.

1.1. Предмет и циљ истраживања

Предмет истраживања ове докторске дисертације представља свеобухватну анализу, развој, методолошко утемељење и ригорозну експерименталну евалуацију два иновативна алгоритма машинског учења чији је примарни задатак превазилажење кључних методолошких и практичних ограничења у домену откривања законитости у подацима (Data Mining). Први део истраживања обухвата детаљан развој алгоритма PISRuleGrowth, концентрисаног на откривање нове категорије образаца, периодичних унутар-секвенцијалних правила заједничких за више секвенци. Други део истраживања односи се на развој алгоритма MEHAUPM, којим се унапређују постојеће методе за откривање скупова ставки високо-просечне корисности у базама података са динамички променљивим предзнаком корисности (мешовитим профитом), решавајући проблеме нетачне евалуације у финансијским трансакцијама.

Основни проблем овог истраживања произилази из ограничења традиционалних модела у моделовању динамичности и структурне сложености реалних података, који се манифестују кроз два кључна изазова. Први изазов се односи на методолошку ограниченост постојећих приступа у области откривања секвенцијалних правила. Наиме, савремени алгоритми за откривање секвенцијалних правила (Sequential Rule Mining) учесталост правила мере искључиво на нивоу целе секвенце, што значи да се свако правило евидентира само једном, без обзира на број његових реалних појава унутар посматране секвенце. Овакво ограничење узрокује систематско потцењивање стварне учесталости правила, будући да се њихове вишеструке појаве унутар појединачне секвенце не евидентирају. Због тога се губи могућност да се прецизно процени реална заступљеност и значај тих правила у бази података. Такође, постојећи алгоритми нису способни да идентификују обрасце који показују регуларност и периодичност, како унутар појединачне секвенце (нпр. понављајуће понашање једног ентитета), тако и кроз више секвенци (нпр. слични обрасци понашања код више ентитета). Овим ограничењем се губи могућност откривања структурно значајних и понављајућих образаца у бази, што је кључно за дубинску анализу сложених секвенцијалних података.

Други критични изазов припада области откривања скупова ставки високе корисности, где се јавља проблем нетачне евалуације корисности када база садржи ставке са променљивим предзнаком корисности. Постојећи алгоритми често дају нетачне резултате јер се прекомерно ослањају на укупну корисност трансакције (Transaction Weighted Utility) као горњу границу за елиминацију ставки. Ова зависност, иако је примарно уведена ради оптимизације брзине претраге, на крају доводи до погрешног елиминисања валидних скупова ставки који, иако садрже артикле са негативном корисношћу у одређеним трансакцијама, ипак формирају високо профитабилне комбинације када се посматрају заједно. Наведена методолошка мањкавост посебно долази до изражаја у финансијским скуповима података који садрже артикле са мешовитим вредностима корисности. Ова ситуација је типична током периода распродаја или промотивних акција, при чему артикли динамички мењају свој предзнак профита, што доводи до значајног нарушавања интегритета добијених аналитичких резултата. Ова два проблема заједно представљају значајан научни и методолошки јаз који захтева иновативан и темељно проверен одговор.

Сходно дефинисаним проблемима, мотивација за ово истраживање је дубоко укорењена у потреби да се унапреди практична применљивост алгоритама из ових области у индустријама које генеришу секвенцијалне и трансакционе податке, као што су електронска трговина, маркетинг и логистика. Академска мотивација се огледа у потреби за систематским развојем алгоритама који су и ефикасни и интерпретабилни. Алгоритми који откривају секвенцијална правила и скупове ставки високе корисности задржавају своју високу интерпретабилност, што је кључно за примену у областима где је транспарентност одлуке неопходна. За разлику од модерних приступа машинском учењу који често функционишу као "црне кутије", ови алгоритми генеришу јасно дефинисана правила која су директно разумљива људским операторима. Практична мотивација проистиче из еволуције пословних захтева, који све више изискују робусне аналитичке моделе способне да обезбеде високу прецизност и суштинско разумевање података. Пословни процеси, укључујући креирање персонализованих маркетиншких кампања, анализу потрошачке корпе и оптимизацију сезонских промоција, захтевају способност модела да идентификују обрасце понашања на нивоу појединца и на нивоу група корисника, као и да прецизно уоче периодичност и процене профитабилност. Ограниченост постојећих алгоритама да ефикасно детектују периодичност у понашању корисника, као и да адекватно обраде мешовити профит, представља кључну практичну препреку приликом доношења оптималних стратешких одлука.

Полазећи од утврђених истраживачких проблема, академског оквира и критичке анализе постојеће литературе, основни циљ ове докторске дисертације је развој, имплементација и валидација два иновативна алгоритма машинског учења — PISRuleGrowth и MEHAUPM. На

тај начин настоји се обезбедити методолошки засновано решење за два суштинска проблема у домену откривања законитости у подацима. Остварење овог циља захтева, поред формалног утемељења, и спровођење опсежне компаративне анализе перформанси (укључујући време извршавања и потрошњу меморије) и квалитативних карактеристика генерисаних образаца (укључујући тачност, информативност и број генерисаних скупова) у односу на водећа постојећа решења. Овај део истраживања ће прецизно утврдити како се предложени модели понашају под различитим условима густине података и различитим праговима, чиме ће се доћи до објективних закључака о њиховој применљивости.

1.2. Полазне хипотезе

Полазна претпоставка овог истраживања јесте да се ограничења постојећих приступа у домену откривања секвенцијалних правила и скупова ставки високе корисности могу превазићи увођењем модела који прецизније одражавају динамичност, структурну сложеност и променљивост реалних података. У том смислу, рад полази од становишта да укључивање унутар-секвенцијалних појављивања, периодичности и заступљености образаца кроз више секвенци, као и адекватна обрада мешовитих позитивних и негативних вредности корисности, може довести до потпунијег, тачнијег и аналитички вреднијег откривања законитости у подацима.

Главна хипотеза истраживања гласи:

H0: Унапређење формалног модела откривања образаца тако да обухвати унутар-секвенцијалну учесталост, периодичност и динамичку промену предзнака корисности омогућава прецизније, потпуније и практично вредније откривање законитости у секвенцијалним и трансакционим базама података, уз конкурентне или побољшане перформансе у односу на релевантне постојеће приступе.

Из главне хипотезе произилазе четири посебне хипотезе:

H0.1: Увођење унутар-секвенцијалног посматрања појављивања правила, заједно са ограничењима периодичности и заступљености кроз више секвенци, омогућава откривање посебне класе понављајућих временских правила која традиционални приступи откривању секвенцијалних правила не могу адекватно да идентификују.

H0.2: Алгоритамска реализација откривања периодичних унутар-секвенцијалних правила, заснована на ефикасном праћењу њихових појављивања и оптимизованом сужавању простора претраге, омогућава њихово генерисање уз прихватљиве и конкурентне временске и меморијске перформансе у односу на постојеће алгоритме за откривање секвенцијалних правила.

H0.3: Коректно моделирање скупова ставки са мешовитим позитивним и негативним вредностима корисности спречава преурањено елиминисање валидних образаца и омогућава тачније откривање скупова ставки високо-просечне корисности у трансакционим базама са динамичком променом предзнака корисности.

H0.4: Примена просечне корисности у условима мешовите корисности доводи до издвајања концизнијих и аналитички вреднијих скупова ставки у односу на приступе засноване на укупној корисности или на неадекватној обради негативних вредности корисности.

1.3. Научне методе истраживања

За потребе овог истраживања користиће се комбинација научних метода како би се обезбедио свеобухватан и поуздан приступ. На самом почетку, применом аналитичко-дедуктивне методе извршиће се преглед релевантне научне литературе и постављање чврстог истраживачког оквира. Након тога, метода развоја софтвера биће примењена за практичну реализацију предложених алгоритама уз детаљно документовање сваке фазе.

Централни део методологије представљаће експериментална метода. Њоме ће се извршити ригорозно тестирање и евалуација развијених алгоритама на одговарајућим реалним скуповима података. Добијени резултати ће затим бити подвргнути компаративној анализи, у којој ће се перформансе предложених алгоритама директно упоредити са перформансама постојећих референтних решења. Коначно, статистичке методе ће се користити за обраду и анализу експерименталних података, што ће омогућити потврђивање постављених хипотеза и извођење објективних закључака.

1.4. Структура и организација рада

Садржај ове докторске дисертације организован је у шест међусобно повезаних поглавља, која се детаљно баве теоријским и практичним аспектима развијених алгоритама.

Након првог поглавља, које пружа свеобухватан увод у области откривања законитости у подацима и дефинише предмет и циљ истраживања, друго поглавље доноси критичку анализу постојеће литературе. Овај део рада је подељен на потпоглавља која засебно разматрају развој и тренутно стање у областима откривања секвенцијалних правила и откривања скупова ставки високе корисности.

У трећем поглављу дато је формално утемељење рада кроз приказ постојећих дефиниција и увођење нових математичких формулација неопходних за решавање идентификованих изазова, чиме се заокружује прецизно дефинисање истраживачког проблема. Четврто поглавље представља срж техничког доприноса дисертације; у њему су детаљно описани архитектура и функционалност алгоритама PISRuleGrowth и MEHAUPM, укључујући пратеће методе и псеудокодове који објашњавају логику њиховог рада.

Експериментална валидација спроведена је у петом поглављу, где су кроз опсежне тестове на реалним и синтетичким скуповима података анализиране перформансе оба алгоритма, уз поређење са релевантним постојећим решењима. Коначно, у шестом поглављу су сумирани резултати истраживања, изведени закључци о научној вредности предложених модела и изложени правци за будући развој у овим областима.

2. Теоријски оквир и сродне методе откривања образаца

Критичка анализа постојећих научних сазнања представља неопходан корак ка разумевању контекста у којем су развијени предложени алгоритми. Ово поглавље пружа детаљан преглед еволуције метода у двема кључним областима откривања образаца које су предмет овог истраживања. Првобитни фокус стављен је на развој техника за откривање секвенцијалних правила, са посебним освртом на ограничења традиционалних приступа у контексту периодичности и унутар-секвенцијалности. У наставку је изложена генеза метода за проналажење скупова ставки високе корисности, уз анализу изазова који се јављају при обради трансакција са променљивим предзнаком корисности. Систематизација претходних истраживања омогућава јасно дефинисање методолошког јаза и истиче потребу за

иновативним решењима која обједињују ефикасност претраге са већом информативном вредношћу добијених образаца.

2.1. Теоријски оквир области откривања секвенцијалних правила и периодичних образаца

Откривање секвенцијалних правила (Sequential Rule Mining – SRM) представља значајан задатак у оквиру откривања образаца, са циљем проналажења свих догађаја за које постоји одређена вероватноћа да ће након њих следити други догађаји у бази секвенци. SRM алгоритми се могу поделити у две категорије на основу начина на који дефинишу секвенцијално правило. Према првој, строжој дефиницији, редослед ставки у левом делу правила/ претходнику (antecedent) и десном делу правила/ следбенику (consequent) мора увек бити исти (Lo et al., 2009). Значајан недостатак који се јавља при коришћењу ове дефиниције огледа се у генерисању великог броја превише специфичних правила. Свака варијација у редоследу ставки унутар леве или десне стране правила третира се као засебно правило, што доводи до смањене прегледности добијених резултата. Поред тога, подршка (support) се израчунава на основу учесталости ових специфичних правила, што доводи до генерисања бројних редундантних правила која се разликују само по редоследу ставки унутар једног дела правила. Штавише, када специфичним правилима недостаје довољна појединачна подршка, сва она могу бити изостављена, што потенцијално може резултовати тиме да правила уопште не буду генерисана. Потребно је нагласити да вредности подршке и поузданости (confidence) код оваквих правила не омогућавају веродостојан приказ њихове стварне релевантности. Разлог је у томе што се подршка једног општег правила расподељује на више специфичних правила, од којих свако садржи само део укупне подршке. Ако једно опште правило $\{a,b\}\Rightarrow\{c\}$ има поузданост од 50%, резултујућа специфична правила $\{a,b\}\Rightarrow\{c\}$ и $\{b,a\}\Rightarrow\{c\}$ би можда имала поузданост од 30%, односно 20%. Ако бисмо тражили минимум од 50% поузданости, занемарили бисмо оба специфична правила и изгубили значајне информације, иако опште правило испуњава наше критеријуме. Стога смо одлучили да користимо делимично уређена секвенцијална правила (partially-ordered sequential rules), скраћено POS правила, заједничка за више секвенци.

Први алгоритам развијен за генерисање делимично уређених секвенцијалних правила био је CMRules (Fournier-Viger et al., 2010). Његов принцип рада заснован је на претварању базе секвенци у трансакциону базу и примени метода за откривање асоцијативних правила (association rule mining) ради издвајања секвенцијалних правила. Након тога, база се поново скенира како би се елиминисала правила која не испуњавају минималну подршку или поузданост. Наредни значајан искорак у овој области представља развој алгоритма RuleGrowth (Fournier-Viger et al., 2011), који се заснива на приступу раста образаца (pattern-growth approach). Увођењем додатних услова за проширивање правила спречено је генерисање сувишних правила, чиме је побољшана ефикасност откривања правила. Предложен је и ERMiner (Fournier-Viger et al., 2014a) као бржа верзија RuleGrowth алгоритма. Он је заснован на класама еквиваленције (equivalence classes) и користи нову структуру података, ретку матрицу бројања (sparse count matrix), ради сужавања простора претраге. Међутим, повећање брзине резултовало је већом потрошњом меморије у односу на RuleGrowth. Претходно наведени алгоритми користе различите методе, али генеришу иста правила. Њихов циљ је проналажење свих правила у бази секвенци која се појављују у минималном броју секвенци (подршка), уз минималну поузданост коју дефинише корисник.

Током година су развијени бројни алгоритми који, уместо откривања свих могућих правила, откривају она која задовољавају специфичне критеријуме претраге. Један од њих је TRuleGrowth (Fournier-Viger et al., 2012), који као даље унапређење RuleGrowth алгоритма уводи ограничење клизећег прозора (sliding-window constraint). Сходно томе, измењена је

дефиниција подршке правила, по којој корисник дефинише максималну величину прозора на основу чега се израчунава број узастопних скупова ставки између претходника (antecedent) и следбеника (consequent). На тај начин се узимају у обзир само она правила чије се појављивање одвија унутар задатог временског оквира или броја узастопних догађаја.

У ситуацијама када су ресурси за анализу ограничени, погоднije решење представља алгоритам TopSeqRules (Fournier-Viger & Tseng, 2011). Код овог приступа, корисник дефинише само параметар k, који одређује број најчешћих правила која је потребно генерисати. Тиме се елиминише потреба за унапред дефинисаним праговима подршке и поузданости, чије одређивање често захтева детаљно предзнање о структури саме базе и изискује додатно време.

На основу TopSeqRules и MCMCA алгоритама (Kryszkiewicz, 1998), развијен је алгоритам TNS (Fournier-Viger & Tseng, 2013), чији је циљ откривање најзначајнијих нередундантних секвенцијалних правила. Ова правила карактеришу минимални претходници и максимални следбеници. Према овој методологији, правило се сматра сувишним ако је његова лева страна подскуп, а десна страна надскуп другог правила, уз услов да оно истовремено поседује једнаку или нижу подршку и поузданост у поређењу са тим референтним правилом.

Значај делимично уређених секвенцијалних правила потврђен је кроз њихову широку примену. Упркос томе, претходно наведене приступе прате одређени недостаци: појављивање правила се своди на само једно по секвенци, чак и када се оно унутар ње јавља вишеструко, док се учесталост израчунава искључиво на основу броја секвенци. Додатно, овим правилима недостају ограничења периодичности и ограничења везана за праћење протеклог времена до следећег појављивања правила. Насупрот томе, ова ограничења су коришћена у бројним истраживачким радовима у сродној области откривања секвенцијалних образаца (sequential pattern mining), где су креирани алгоритми за проналажење периодичних образаца у једној (Fournier-Viger et al., 2017; Fournier-Viger et al., 2016; Fournier-Viger et al., 2019c) и више секвенци (Fournier-Viger et al., 2019b; Fournier-Viger et al., 2020). Управо смо у овим студијама пронашли мотивацију да концепте периодичности и временских ограничења преточимо у домен секвенцијалних правила, креирајући алгоритам под називом периодични унутар-секвенцијални RuleGrowth алгоритам, који ће се у даљем тексту наводити као PISRuleGrowth.

2.2. Теоријски оквир области откривања скупова ставки високе корисности

Откривање скупова ставки високе корисности (High-utility itemset mining) појавило се као истраживачка тема са представљањем Two-Phase алгоритма 2005. године (Liu et al., 2005). Оно решава недостатке учесталих скупова ставки (frequent itemset mining), код којих се не узимају у обзир атрибути ставки, као што су количина или профит. Дефинисањем корисности (utility) као мере значаја ставке за корисника (Hu & Mojsilovic, 2007), скупови ставки високе корисности представљају обрасце који су истовремено статистички и практично значајни. У складу са наведеним, овај приступ налази широку примену у разноврсним областима, укључујући откривање образаца кроз веб-сајтове (Ahmed et al., 2009), анализу секвенци приступа вебу (Ahmed et al., 2011), понашање корисника у мобилној трговини (Shie et al., 2011), анализу регулације гена (Zihayat et al., 2017), откривање скривених профитабилних инвестиционих образаца (Lin et al., 2017b), као и препоруку академске литературе (Liang et al., 2011). Међутим, упркос овој широкој примени, овај приступ се и даље највише примењује у секторима продаје и малопродаје, где је крајњи циљ максимизација прихода. Типични задаци обухватају анализу потрошачке корпе ради идентификовања високопрофитних група ставки за унакрсну продају (cross-selling) (Gan et al., 2018b; Jabbar et al., 2015; Demir et al., 2019; Kavitha & Geetha, 2016; Weng, 2016), као и планирање позиционирања производа које укључује

трошкове залиха и ограничења у погледу рока трајања производа (Lan et al., 2011; Mondal et al., 2021; Gan et al., 2022).

Упркос наведеним унапређењима, фундаментални недостатак традиционалног откривања скупова ставки високе корисности представља претпоставка о искључиво позитивним вредностима корисности, која се у реалним околностима често показује као неодржива. Пословни субјекти неретко толеришу негативну корисност појединих производа у циљу максимизације укупног профита, прихватају тренутни губитак ради дугорочног позиционирања на тржишту, или се суочавају са ситуацијама у којима трошкови циклично надмашују приходе, својственим за сезонско пословање. Да би се ови изазови превазишли, 2009. године је представљен HUINIV-Mine (Chu et al., 2009), као први алгоритам који омогућава обраду негативних вредности корисности у оквиру откривања скупова ставки високе корисности. Међутим, ефикасност алгоритма HUINIV-Mine била је компромитована потребом за три скенирања базе података и високим меморијским оптерећењем. Ови изазови послужили су као основа за развој читавог низа каснијих решења усмерених на оптимизацију процеса откривања скупова ставки високе корисности.

Један значајан пример је алгоритам FHN, развијен 2014. године (Fournier-Viger, 2014), који се и даље сматра једним од најсавременијих алгоритама (state-of-the-art) у овој области. Он примењује три нове структуре засноване на листама, за позитивне ставке, негативне ставке и процену заједничког појављивања ставки. Овај алгоритам је развијен као унапређење већ ефикасног FHM (Fournier-Viger et al., 2014b) алгоритма, при чему је доказано да је вишеструко ефикаснији у погледу времена и меморије у односу на HUINIV-Mine. Током 2018. године предложен је алгоритам EHIN (Singh et al., 2018), који примењује концепт раста стабла образаца (pattern-growth tree approach). Он врши спајање трансакција како би се додатно сузио простор претраге и захтева мање временских и меморијских ресурса од FHN алгоритма, нарочито код густих скупова података.

Стога се обрада негативних вредности корисности наметнула као неопходно проширење различитих варијација алгоритама за проналажење скупова ставки високе корисности. Први алгоритам намењен откривању топ-к скупова ставки са негативним вредностима корисности био је TopHUI, настао као дериват FHN алгоритма (Gan et al., 2020). Тренутно најсавременији и најкомплекснији приступ представља TKN алгоритам (Ashraf et al., 2022). Он се ослања на структуре низова ради ефикаснијег израчунавања, уводи иновативне стратегије за подизање прага и скраћивање стабла претраге, те смањује зависност од преостале корисности трансакција (RTWU). Поред тога, TKN обједињује дупликате трансакција како би додатно унапредио перформансе система. Такође, алгоритам PHMN (Lai et al., 2023) представља значајно проширење усмерено на проналажење периодичних образаца у присуству негативних вредности, при чему уводи просечну периодичност као кључни критеријум анализе.

Ипак, постојећи алгоритми показују одређене недостатке у прецизности прорачуна приликом обраде података из реалних и динамичких система. Они често нису у стању да адекватно обраде ставке чије вредности профита флуктуирају током времена, наизменично попримајући позитивне и негативне вредности. Ово ограничење је критично јер је профитабилност производа или услуга по својој природи динамична. Она често поприма негативне вредности током промотивних кампања или стратегија унакрсне продаје, након чега се поново враћа на позитивне износе. TKN алгоритам за све ставке претпоставља статичну вредност профита, док остали алгоритми само статично фиксирају позитиван или негативан предзнак. Главни фактор који доприноси проблему обраде негативних вредности јесте прекомерно ослањање на TWU вредност (transaction-weighted utility) као гранични параметар. Недостаци ових алгоритама детаљније су изложени у наставку:

- PHMN алгоритам се показао као најпрецизнији, а до погрешних резултата је долазио једино у случајевима када ставке имају негативну укупну корисност трансакција.

Нетачност проистиче из третирања ставки које се појављују у негативним трансакцијама као да саме по себи имају негативне вредности. Оваква претпоставка доводи до њиховог преурањеног одбацивања и онемогућава њихово комбиновање у скупове ставки високе корисности.

- FHN алгоритам се суочава са истим проблемима у прецизности као и PHMN, али уз додатно ограничење. Онемогућава се формирање скупова ставки који се састоје искључиво од оних елемената чија је укупна корисност негативна. На пример, уколико две ставке појединачно имају негативну укупну корисност, њима није дозвољено самостално формирање заједничког скупа. Такве ставке могу чинити скуп само у комбинацији са другим елементима који имају позитивну укупну корисност. Ово представља значајан недостатак јер две ставке са индивидуално негативном корисношћу могу заједно остварити позитивну збирну корисност унутар специфичних трансакција, али FHN алгоритам такве потенцијалне скупове унапред одбацује.

- Недостаци карактеристични за FHN присутни су и код HUINIV-MINE алгоритма, уз додатни проблем класификације мешовитих вредности корисности као потпуно негативних. Због тога две ставке, које поред негативних имају и позитивне трансакционе вредности уз укупно позитивну корисност, не могу самостално формирати скуп. Њихово обједињавање у скуп изводљиво је искључиво уз присуство других ставки које имају само позитивне вредности.

- EHIN се показао као најмање прецизан, задржавајући исте недостатке као и HUINIV-Mine. Поред тога, његово претерано ослањање на прорачуне преостале корисности (remaining utility) доводи до откривања још мањег броја валидних скупова ставки високе корисности, чиме се додатно продубљују проблеми карактеристични за анализу негативне корисности.

Сумирајући претходна разматрања, истраживања у области откривања скупова ставки високе корисности уз присуство негативних вредности корисности изнедрила су бројна решења која значајно проширују употребну вредност овог концепта у реалним пословним окружењима. Ипак, детаљна анализа је указала на озбиљне недостатке у прецизности водећих алгоритама при обради специфичних случајева. Овај методолошки изазов ефикасно се адресира усмеравањем фокуса на тачност израчунавања корисности код скупова ставки мешовите корисности. Надаље, истиче се оправданост коришћења просечне корисности (average utility) као објективнијег критеријума у односу на тоталну корисност, чиме се постиже већи степен подударности између теоријских резултата и реалних потреба примене. Управо су наведени увиди послужили као полазиште за развој модификованог алгоритма, чији је примарни фокус усмерен на унапређење прецизности, откривање скупова ставки високо-просечне корисности и прецизан прорачун мешовитих (односно динамичких) вредности корисности.

На основу анализе сродних метода из обе области, може се закључити да су постојећи приступи значајно унапредили могућности откривања образаца у секвенцијалним и трансакционим подацима, али да и даље показују ограничења у моделирању динамичности и сложених структура реалних података. У домену секвенцијалних правила, недовољна пажња је посвећена унутар-секвенцијалној учесталости, периодичности и временским зависностима, док у области откривања скупова ставки високе корисности доминирају проблеми нетачне евалуације код мешовитих и променљивих вредности корисности. Управо ови недостаци намећу потребу за успостављањем прецизнијег формално-аналитичког оквира, који ће послужити као основа за развој предложених модела и унапређење тачности и квалитета добијених резултата.

3. Основни појмови и формулисање проблема истраживања

Ово поглавље је посвећено анализи релевантних концепата и увођењу нових дефиниција које чине основу за систематично разумевање предмета истраживања. Посебан фокус стављен је на прецизну формулацију истраживачких проблема у обе области, чиме се успоставља теоријски оквир за даљи рад.

3.1. Дефинисање проблема периодичних унутар-секвенцијалних правила

Дефиниција 1. База података секвенци састоји се од скупа секвенци S={s1,s2,…,sX} који садржи скуп ставки I={i1,i2,…,iY}. Секвенца представља уређену листу скупова ставки (скупова елемената) којој је додељен јединствени идентификатор секвенце (sid) (Agrawal & Srikant, 1995).

Пример базе података секвенци приказан је у Табели I. Она се састоји од две секвенце са припадајућим идентификаторима. Сваки број представља појединачну ставку, док сваки скуп бројева унутар витичастих заграда означава трансакцију или скуп ставки. Прва секвенца садржи девет трансакција, при чему се прва трансакција састоји од две ставке.

Табела I

Пример секвенцијалне базе

sid 1	tids:	1	2	3	4	5
	itemsets:	({1,2},	{3},	{1,3,2},	{1},	{2,3},
		6	7	8	9	
		{1},	{3,2},	{3},	{1,2})	
sid 2	tids:	1	2	3	4	5
	itemsets:	({2,1,3},	{2,3},	{1},	{3,1},	{2},
		6	7	8	9	
		{1,2,3},	{1,3},	{2},	{1,3})	

Дефиниција 2. Секвенцијално правило, означено као X⇒Y, представља однос између два неуређена скупа ставки X и Y који припадају истој секвенци и немају заједничких елемената. Скуп ставки Y мора се појавити након скупа ставки X унутар секвенце (Fournier-Viger et al., 2010).

На пример, правило {1,2}⇒{3} у првој секвенци представља секвенцијално правило из следећих разлога: број 3 се појављује након ставки 1 и 2, сви елементи су различити и налазе се у оквиру исте секвенце.

Дефиниција 3 (Величина правила). Правило X⇒Y има величину a·b уколико скуп X садржи a, а скуп Y садржи b ставки. Друго правило је веће од X⇒Y ако је један од његових скупова (леви или десни) бројнији, док је преостали скуп по величини већи или једнак скупу X, односно Y (Fournier-Viger et al., 2011).

Дефиниција 4 (Валидност). Проблем откривања секвенцијалних правила заједничких за више секвенци дефинише се као проналажење свих валидних правила у бази података секвенци (Fournier-Viger et al., 2010).

Да би се правила сматрала валидним, она морају задовољити минималне вредности подршке и поузданости. Ови параметри представљају кориснички дефинисане прагове, те њихове вредности нису константне. Сходно томе, исто правило може бити првобитно валидно, али након што корисник постави више захтеве, оно постаје невалидно.

На пример, претходно је утврђено да {1,2}⇒{3} представља секвенцијално правило. Ипак, без израчунатих вредности подршке и поузданости, таква правила се означавају као „правила кандидати" све док се не утврди њихова валидност.

У наредним одломцима биће представљене нове дефиниције, првенствено изведене из области откривања секвенцијалних образаца (Fournier-Viger et al., 2019b), али прилагођене за потребе откривања секвенцијалних правила. Поједине дефиниције су оригиналне и развијене су специфично за потребе овог рада.

Дефиниција 5. Секвенцијално правило унутар секвенце (енгл. Intra-sequential rule – IS), односно делимично уређено правило X⇒Y, дефинише се као секвенцијално правило које се унутар једне секвенце појављује више пута. Ово подразумева да се ставке из скупа X, праћене ставкама из скупа Y, појављују у појединачним трансакцијама секвенце најмање два пута узастопно. Појављивање скупа X након којег следи Y може се сматрати појавом IS правила само у случају да ни X ни Y нису део другог појављивања истог правила које је већ пребројано. Пребројана појављивања називају се „валидним појављивањима". Важно је разликовати валидно појављивање правила од самог валидног правила, при чему први термин указује искључиво на то да ће та конкретна појава бити урачуната у број појављивања правила. Појављивање правила може се изразити кроз парове идентификатора трансакција (tid) у којима се налазе X и Y, под условом да трансакција са скупом Y следи након трансакције са скупом X, чиме се заједно формира секвенцијално правило.

Дефиниција 6. Подршка секвенцијалног правила X⇒Y унутар секвенце s представља број његових валидних појављивања у s подељен са укупним бројем трансакција (|s|), односно скупова ставки у тој секвенци, што се назива дужином секвенце. Стога,

$$sup(X \Rightarrow Y) = validOccur(X \Rightarrow Y)/|s| \quad (0.1)$$

На пример, за sid1 је претходно утврђено да се правило {1,2}⇒{3} може сматрати секвенцијалним. Трансакције су означене идентификаторима (tid), који се броје слева надесно почев од 1 (tid1, tid2, tid3…). Правило {1,2}⇒{3} се појављује у следећим трансакцијама: {(1;2),(3;5),(6,7;8)}, што значи да се ставке 1 и 2 појављују у првој, а затим 3 у другој трансакцији. Претрага за ставкама антецедента наставља се тек након што се скуп ставки из консеквента појави у трансакцији. Према томе, ово правило има 3 валидна појављивања и може бити кандидат за валидно IS правило (у зависности од изабраних прагова за minSup и minConf). Поједина појављивања овог правила не могу бити укључена, као што је случај са трансакцијама (4,5;7). Разлог је у томе што се tid4 налази унутар већ избројаног интервала правила (3;5), па се не може користити за формирање новог појављивања, јер би то довело до преклапања са већ евидентираним tid5. Подршка правила {1,2}⇒{3} израчунава се на следећи начин: validOccur({1,2}⇒{3})/|sid1|=3/9=0,33.
Дефиниција 7. Поузданост правила у секвенци s се рачуна као:

$$sup(X \Rightarrow Y)/sup(X), \quad (0.2)$$

или још прецизније у овом контексту, можемо је формулисати и као:

$$validOccur(X \Rightarrow Y)/validOccur(X), \quad (0.3)$$

зато што вредности нису подељене дужином секвенце s. Она представља количник између броја појављивања комплетног правила (где након леве стране следи десна) и броја случајева у којима се појављује само лева страна без накнадног појављивања десне стране унутар секвенце.

Број појављивања правила {1,2}⇒{3} већ је израчунат у претходном примеру и износи 3. Како бисмо одредили поузданост овог правила у sid1, неопходно је израчунати број валидних појављивања скупа ставки {1,2} у тој секвенци. Валидна појављивања скупа {1,2} (означена идентификаторима трансакција у којима се појављују унутар sid1) су: {(1) ,(3), (4,5), (6,7), (9)}, што даје укупно validOccur({1,2})=5. Коришћењем формуле (1.2), поузданост се рачуна као: conf({1,2}⇒{3} ,sid1) = validOccur({1,2}⇒{3}) / validOccur({1,2}) =3/5=0,6.

Према томе, секвенцијално правило {1,2}⇒{3} у секвенци sid1 има подршку 0,33 и поузданост 0,6. Ово правило се сматра валидним IS правилом у секвенци sid1 уколико су кориснички дефинисани прагови за minSup и minConf мањи од 0,33, односно 0,6

Дефиниција 8. Честа секвенцијална правила су она чија је подршка једнака или већа од дефинисаног минималног прага подршке (Fournier-Viger et al., 2019b). Сходно томе, сва валидна секвенцијална правила унутар секвенце (IS) истовремено су и честа унутар-секвенцијална правила.

Правило {1,2}⇒{3} представља често IS правило јер је потврђено као валидно унутар секвенце sid1. У претходним истраживањима (Fournier-Viger et al., 2017; Fournier-Viger et al., 2019c), скуп ставки X сматран је периодичним уколико је његов максимални период био мањи од прага maxPer. Максимални период скупа ставки дефинише се као највећи број скупова ставки између два узастопна појављивања скупа X унутар те секвенце. Међутим, примена maxPer прага има одређене недостатке. Ако се постави превисоко, сваки скуп ставки би се сматрао периодичним, док би прениско постављен праг довео до тога да поједини обрасци остану неоткривени само због одступања у једном периоду (једног додатног скупа ставки између њих). Узимајући ово у обзир, стандардна девијација је уведена као нова мера периодичности скупова ставки (Fournier-Viger et al., 2019b; Fournier-Viger et al., 2020). Вредност стандардне девијације представља варијацију посматраних вредности у односу на њихову просечну вредност. Стога мања вредност стандардне девијације указује на стабилне периоде (Fournier-Viger et al., 2017). На основу наведених чињеница, у овом раду се опредељујемо за коришћење стандардне девијације периода као мере за периодичност унутар-секвенцијалних правила.

Дефиниција 9. Према Теореми 2 (Fournier-Viger et al., 2020), стандардна девијација обрасца X у секвенци s се може рачунати као:

$$std(X,s)=\sqrt{\left(\sum_{i=1}^{k+1}\left(\llbracket per \rrbracket_i\right)^2\right)/(k+1)-\left(\llbracket |s|/(k+1) \rrbracket\right)^2}, \quad (1.4)$$

где је k sup(X,s) за сваки период X у s, а |s| је дужина секвенце.
Одговарајућа формула за периодична унутар-секвенцијална правила креирана је заменом обрасца X правилом X⇒Y, као и заменом подршке sup(X,s) вредношћу validOccur(X⇒Y,s). Главни изазов представља питање шта тачно треба сматрати периодом између појављивања секвенцијалних правила. Постоји неколико могућности: број трансакција између краја скупа Y и почетка наредног скупа X, између почетка једног Y и почетка следећег Y, између краја једног X и краја следећег X, и тако даље. Одлучили смо се за опцију која се најбоље усклађује са осталим праговима и која највернији приказује оно што се може сматрати периодом унутар-секвенцијалног правила. Прву наведену могућност смо одбацили због њене тенденције да производи изразито непропорционалне вредности у односу на величину секвенце, што резултира нереално ниском стандардном девијацијом. Сходно томе, периоде секвенцијалних правила дефинисали смо као размаке између последње трансакције скупа Y и последње трансакције наредног појављивања скупа Y.

На пример, израчунаћемо стандардну девијацију правила {1,2}⇒{3} у секвенци sid1. Да бисмо то урадили, прво морамо одредити периоде за ово правило. Како већ имамо валидна појављивања {(1;2),(3;5),(6,7;8)}, можемо приступити прорачуну периода. Периоди су: 2 (од 0 до 2), 3 (између 5 и 2), 3 (између 8 и 5) и 1 (између краја секвенце, 9, и 8). Скуп периода за {1,2}⇒{3} у sid1 је {2, 3, 3, 1}, што је у складу са дефиницијом да број чланова (k+1) буде за један већи од броја валидних појављивања. Сада када имамо све чланове, можемо израчунати стандардну девијацију:

$$\text{std}(1, 2 \Rightarrow 3) = \sqrt{(2^2 + 3^2 + 3^2 + 1^2)/(3 + 1) - (|9|/(3 + 1))^2} = 0.83.$$

За праг maxStd који је већи од 0,83, правило {1,2}⇒{3} се сматра периодичним у секвенци sid1.

У овом тренутку располажемо свим информацијама о учесталости и периодичности правила {1, 2}⇒{3} унутар секвенце sid1. Наведени параметри били би сасвим довољни у случају да је примарни фокус истраживања развој алгоритма за анализу правила унутар једне секвенце или кроз више секвенци. Међутим, наш циљ је откривање периодичних унутар-секвенцијалних правила која су заједничка за више различитих секвенци, а ниједан од досадашњих параметара не повезује појављивање правила у различитим секвенцама. Они пружају информације искључиво о појављивању правила у свакој секвенци појединачно. Због тога уводимо нови праг под називом коефицијент периодичности секвенци, који у нашем случају дефинишемо као однос периодичности секвенцијалних правила.

Дефиниција 10 Коефицијент периодичности секвенце (енгл. Sequence periodic ratio): обрасца X представља однос између броја секвенци које садрже тај образац и укупног броја секвенци у бази података S (Fournier-Viger et al., 2017), што се формално изражава као:

$$ra(X)=numSeq(X)/|D|. \qquad (0.4)$$

Одговарајућа формула за коефицијент периодичности секвенцијалног правила изводи се заменом обрасца X секвенцијалним правилом X⇒Y, при чему се numSeq(X) мења у numSeq(validOccurOfValidISRule(X⇒Y)), односно у број секвенци које садрже валидна појављивања валидних унутар-секвенцијалних правила. Минимални број појављивања правила постављен је на два, будући да правила не могу бити периодична уколико се појављују само једном. Сходно томе, секвенце у којима је број појављивања правила X⇒Y мањи од наведеног прага не садрже валидна појављивања тог правила.

Дефиниција 11. Секвенцијално правило X⇒Y је периодично унутар-секвенцијално (PIS) правило ако је однос броја секвенци у којима је ово правило валидно и укупног броја секвенци у бази већи од кориснички дефинисаног прага minRa, односно ra(X⇒Y)>minRa. Правило се сматра валидним секвенцијалним правилом унутар секвенце s ако су испуњени следећи услови: sup(X⇒Y,s)>minSup, conf(X⇒Y,s)>minConf и std(X⇒Y,s)<maxStd.

За израчунавање вредности ra за правило {1,2}⇒{3}, неопходно је прво одредити вредности параметара у секвенци sid2. Оне износе: sup(1,2⇒3,sid2)=0,33, conf(1,2⇒3,sid2)=0,6 и std(1,2⇒3,sid2)=1,48. Уз задате прагове minSup=0,3, minConf=0,5 и maxStd=2, параметри правила {1,2}⇒{3} испуњавају постављене критеријуме у обе секвенце. Уколико се праг minRa постави на 0,6, израчунавање коефицијента у овом случају износи: ra(1,2⇒3)=2/2=1, што је веће од minRa=0,6, те се {1,2}⇒{3} може сматрати периодичним унутар-секвенцијалним правилом заједничким за више секвенци.

Међутим, промена прага maxStd на вредност 1 доводи до другачијих резултата. С обзиром на то да је std(1,2⇒3,sid2)=1,48>maxStd=1, ово правило престаје да буде валидно периодично унутар-секвенцијално правило у секвенци 2 (sid2). Сходно томе, вредност коефицијента се

мења јер сада само једна секвенца садржи ово правило у валидном облику. Нови прорачун износи: ra(1,2⇒3)=1/2=0,5, што је мање од minRa=0,6, чиме се не испуњава задати праг. На основу тога можемо закључити да, уз овако дефинисане параметре, {1,2}⇒{3} није периодично унутар-секвенцијално правило заједничко за више секвенци.

Дефиниција проблема

На основу задате базе секвенци D, три минимална кориснички дефинисана прага (minSup, minConf, minRa) и једног максималног прага (maxStd), циљ овог истраживања је откривање свих периодичних унутар-секвенцијалних правила заједничких за више секвенци. Другим речима, тежи се проналажењу свих оних правила у бази података која су заступљена у одређеном проценту секвенци као периодична унутар-секвенцијална правила. То подразумева да је број секвенци у којима та правила остварују вредности параметара веће од minSup и minConf, а мање од maxStd, изнад задатог прага minRa.

3.2. Дефинисање проблема скупова ставки високо-просечне корисности са мешовитим вредностима корисности

Дефиниција 1 (Трансакциона база података): Трансакциона база података је скуп трансакција које су дефинисане јединственим идентификатором познатим као tid. Поред тога, свака ставка у трансакцији је повезана са две вредности корисности: једном која обично указује на количину и другом која означава профит (или значај/важност) (Fournier-Viger, 2014).

Дефиниција 2 (Укупна корисност ставке у трансакцији): Корисност ставке у трансакцији представља производ њене количине и њеног профита (или друге мере значаја) (Fournier-Viger, 2014).

Дефиниција 3 (Просечна корисност скупа ставки у трансакцији): Просечна корисност k-скупа ставки у трансакцији представља количник суме појединачних корисности ставки и k (Chu et al., 2009).

Дефиниција 4 (Укупна корисност / Просечна корисност скупа ставки у бази података, sumUtils): Укупна корисност k-скупа ставки у бази података представља суму његових корисности кроз све трансакције у којима се он појављује. Просечна корисност се добија дељењем укупне корисности са k (Chu et al., 2009).

Дефиниција 5 (Скуп ставки високе корисности / Скуп ставки високо-просечне корисности, HUI/HAUI): Скуп ставки је скуп ставки високе корисности / високо-просечне корисности уколико његова корисност у бази података није мања од одговарајућих прагова минималне корисности / минималне просечне корисности (Chu et al., 2009).

Дефиниција 6 (Максимална корисност трансакције, tmu): Максимална корисност трансакције представља највишу вредност корисности појединачне ставке унутар те трансакције. Она служи као прецењена горња граница за предвиђање потенцијалне корисности проширења скупа ставки (Lin et al., 2017a).

Дефиниција 7 (Горња граница просечне корисности, auub): Горња граница просечне корисности представља оптимистичну процену просечне корисности скупа ставки. Дефинише се као сума максималних вредности корисности свих трансакција које садрже тај скуп ставки (Lin et al., 2017a).

Дефиниција 8 (Преостала корисност, remu): Преостала корисност пружа процену горње границе за корисност било ког надскупа тренутног скупа ставки. Постоје две уобичајене варијанте:

- Преостала корисност заснована на суми (Sum-based remu): сума корисности свих ставки које следе након тренутног скупа ставки у сортираној трансакцији (Liu et al., 2005).
- Преостала корисност заснована на максимуму (Max-based remu): највиша вредност корисности међу ставкама које следе након тренутног скупа ставки у сортираној трансакцији (Lin et al., 2017a).

Иако је дефиниција заснована на максимуму коришћена у EHAUPM алгоритму, ми усвајамо ревидирану верзију дефиниције засноване на суми, уз образложење дато у наредном потпоглављу. Користили смо преосталу корисност која се састоји искључиво од позитивних вредности.

Дефиниција 9 (Ревидирана максимална корисност трансакције, rmu): За дати скуп ставки унутар трансакције, ревидирана максимална корисност трансакције представља највећу вредност корисности међу свим ставкама које чине тај скуп или се јављају након њега у сортираној трансакцији. Овако ажурирана граница узима у обзир потенцијалне вредности корисности које могу донети преостале ставке.

Дефиниција 10 (Модификована листа просечне корисности, MAUList): За сваки скуп ставки, MAUList садржи податке на нивоу трансакције који укључују: идентификатор трансакције, стварну корисност, ревидирану максималну корисност трансакције и процењену преосталу корисност (Lin et al., 2017a).

Дефиниција 11 (Матрица копојављивања процењене просечне корисности, EAUCM): EAUCM је помоћна табела (lookup table) која чува вредности горње границе просечне корисности (auub) за сваки могући пар перспективних појединачних ставки. Она ефикасно одсеца веће скупове ставки (са 3 или више ставки) уколико било који пар унутар њих има недовољну вредност auub, на основу претпоставке да неперспективни парови подразумевају и неперспективне надскупове (Lin et al., 2017a).

Дефиниција 12 (Скуп ставки са негативном корисношћу): Скуп ставки са негативном корисношћу је онај скуп ставки чија је укупна корисност кроз све трансакције негативна (Fournier-Viger, 2014).

Дефиниција 13 (Ставка/скуп ставки са мешовитом корисношћу): Ставка или скуп ставки са мешовитом корисношћу поприма и позитивне и негативне вредности корисности кроз различите трансакције унутар дате базе података. Такве ставке одликују динамичне промене знака њиховог доприноса корисности дуж целе базе. Као последица тога, њихова збирна укупна корисност у бази података може бити позитивна, негативна или једнака нули.

Дефиниција проблема

Проблем откривања скупова ставки високо-просечне корисности са мешовитим вредностима корисности подразумева идентификовање свих скупова ставки чија корисност није мања од минималног прага који задаје корисник, и то у бази података у којој вредности корисности могу динамички варирати између позитивних и негативних кроз различите трансакције.

У овом поглављу је постављена теоријска основа за откривање периодичних унутар-секвенцијалних правила и скупова ставки високо-просечне корисности са мешовитим вредностима корисности. Формализовањем ових концепата успостављени су прецизни критеријуми за препознавање образаца који су истовремено временски стабилни и вредносно

значајни у оквиру трансакционих и секвенцијалних база података. Овако дефинисан оквир омогућава развој ефикасних метода за оптимизацију простора претраге, што чини полазну тачку за алгоритамска решења представљена у наставку рада.

4. Развој и имплементација предложених алгоритама

У овом поглављу приказани су алгоритми PISRuleGrowth и MEHAUPM, који представљају кључне компоненте предложеног истраживачког оквира. Ради јасног и систематичног излагања, њихов рад је формално представљен помоћу псеудокодова, чиме се омогућава апстракција од конкретних имплементационих детаља и јасно истицање основне логике алгоритама. Поред псеудокода, дат је и опис најважнијих корака, коришћених структура података и правила проширивања и одсецања, како би се олакшало разумевање начина на који алгоритми функционишу.

4.1. PISRuleGrowth: архитектура и логика алгоритма

У овом одељку описан је предложени PISRuleGrowth алгоритам, нови алгоритам дизајниран за откривање периодичних правила унутар секвенци која су заједничка за више различитих секвенци. Алгоритам је заснован на RuleGrowth (Fournier-Viger et al., 2011) алгоритму и користи приступ раста образаца (pattern-growth) за генерисање правила. Алгоритам се логички може поделити на четири главна дела који представљају различите процедуре: скенирање базе података, главну процедуру, као и процедуре expandleft и expandright. Ови делови су такође основни сегменти RuleGrowth алгоритма, али је њихов садржај измењен и допуњен бројним методама како би се остварио циљ откривања периодичних унутар-секвенцијалних правила.

Упркос привидним сличностима, кључна разлика између овог приступа и свих осталих алгоритама за откривање правила (Fournier-Viger et al., 2011; Fournier-Viger et al., 2014a; Fournier-Viger et al., 2010) огледа се у начину претраге: већина постојећих решења фокусира се на једно појављивање по секвенци, док се овде идентификују сва појављивања правила у свакој појединачној секвенци. Ово је алгоритам за анализу унутар секвенци (intra-sequential) који проналази учестале и периодичне обрасце у свакој секвенци, а затим издваја оне са важећим појављивањима у минималном броју секвенци. Важно је напоменути да периодичност претходно није била уведена као ограничење код делимично уређених секвенцијалних правила. У наставку је представљена главна процедура алгоритма.

Algorithm 1: PISRuleGrowth algorithm's pseudo-code

Algorithm (database, outputFile, minSup, minConf, maxStd, minRa)

1. mapItemsOccur ⟵ CALL databaseScan ()
2. databaseSize ⟵ database.Length
3. seqLengths ⟵ ∅
4. FOR each d, such that d ∈ [0, databaseSize-1]:
5. seqLengths.Set(d) ⟵ database.Get(d).Length
6. END FOR (line 4)
7. database ⟵ ∅
8. FOR each pair of itemI, itemJ such that itemI, itemJ ∈ mapItemsOccur

```
9.    I ⟵ itemI.Item
10.   J ⟵ itemJ.Item
11.   IF (I == J) OR (I > J) THEN CONTINUE (line 8)
12.   occurI ⟵ itemI.Get(I)
13.   occurJ ⟵ itemJ.Get(J)
14.   param_value_ij ⟵ ∅
15.   sids_for_expansion ⟵ ∅
16.   FOR each s in occurI
17.         occurr_I_InRule, occur_J_InRule ⟵ CALL get_SingIe_Item_Rule_Occur (s.Tids.First,
occurJ.Get(s.Sid).First, seqLengths.Get(s.Sid))
18.     IF occur_I_InRule ≠ ∅ THEN
              parameterValues ⟵ CALL is_PIS_Rule (occur_I_InRule, occur_J_InRule,
s.Tids.Length, seqLengths.Get(s.Sid))
19.       IF parameterValues ≠ ∅ THEN
20.         sids_for_expansion.Add(s.Sid)
21.           IF parameterValues.Length == 3 THEN
22.             param_value_ij.Add(s.Sid, parameterValues)
23.     [ the same lines – from 17 to 22 are used for searching for rule J→I, by swapping everywhere
I and J]
24.   END FOR (line 16)
25.   raIJ ⟵ param_value_ij.Length / databaseSize
26.   IF sids_for_expansion ≠ ∅ AND raIJ > minRa THEN
            IF param_value_ij ≠ ∅ THEN
            CALL Save_Rule (raIJ, itemI, itemJ,
  param_value_ij)
27.     CALL ExpandLeft (itemI, itemJ, sids_for_expansion)
28.     occur_I_Count ⟵ ∅
29.     FOR each s in occurI
30.       occur_I_Count.Add(s.sid, s.Tids.Length)
31.     END FOR (line 29)
32.     CALL ExpandRight (itemI, itemJ, sids_for_expansion, occur_I_Count)
33. [ the same lines – from 25 to 32 are used for searching for rule J→I, by swapping everywhere I
    and J]
34. END FOR (line 8)
```

PISRuleGrowth прихвата шест улазних аргумената: секвенцијалну базу података, путању за чување излазне датотеке са генерисаним правилима, као и кориснички дефинисане прагове minSup, minConf, maxStd и minRa изражене у процентима. Како бисмо избегли сувишна објашњења у процедурама и методама, помињемо само правило I⇒J, док изостављамо J⇒I. Разлог томе је што генерисање правила J⇒I суштински подразумева понављање истих позива метода и процедура коришћених за I⇒J након његовог извршавања, уз једину разлику у замени параметара који садрже „I” онима који садрже „J”.

Временска сложеност PISRuleGrowth алгоритма одређује се преко агрегације утрошка времена појединачних операција експанзије. За свако генерисано правило r, нека је Cr скуп кандидата за проширење, који чине само оне ставке које се јављају у довољном броју заједничких секвенци са правилом и задовољавају све предуслове за укључивање. Нека је Sr=|sidsIJ| број секвенци у којима правило I→J има минималну потребну подршку, а L просечна дужина секвенци у бази. Тада се трошак проширења једног правила може изразити као

$$O(\sum_{r}(C_r \times S_r \times L_r)$$

Ова формула јасно показује да перформансе алгоритма не зависе директно од величине улазне базе, већ искључиво од динамички одређеног подскупа релевантних ставки и секвенци за свако правило. Прагови minRa, minSupp, minConf и maxStd драстично утичу на величину скупова Cr и Sr, што објашњава ефикасност алгоритма чак и на великим базама података. Рекурзивна природа експанзије подразумева да се укупна сложеност увећава дуж путања проширења, али сваки корак ограничава претрагу искључиво на делове базе који су релевантни за текуће правило.

Меморијска сложеност алгоритма PISRuleGrowth директно је пропорционална укупном броју појављивања свих честих ставки у бази података које задовољавају постављене прагове подршке. Ова зависност произлази из потребе да се за сваку честу ставку ефикасно прати свако њено појављивање у свакој секвенци, укључујући тачне временске ознаке, како би се могле израчунавати метрике периодичности и подршке. Алгоритам, стога, захтева простор реда

O(T)

где T представља збирни број појављивања свих ставки које испуњавају услове минималне подршке (minSupp) и минималног учесталости (minRa).

Ова линеарна зависност осигурава да меморијски захтев није већи од неопходног за израчунавање периодичних правила, пошто се не чувају подаци о нерелевантним ставкама. Прагови minRa и minSupp делују као филтери који драстично смањују укупан број чуваних појављивања, омогућавајући алгоритму да ефикасно ради чак и на већим базама података. Дакле, меморијски трошак је директно повезан са количином релевантних података након примене ограничења, чиме се постиже равнотежа између прецизности и скалабилности.

4.1.1. Скенирање базе података

Након учитавања базе података, следећи корак је њено скенирање. Резултат овог скенирања је хеш мапа која се назива „мапа ставки“ или „мапа појављивања ставки“. Реч је о мапи унутар мапе, где је кључ спољашње мапе ставка, а унутрашња мапа садржи идентификатор секвенце (sid) и листу појављивања те ставке у датој секвенци. Ова мапа се у даљим корацима алгоритма користи уместо оригиналне базе података.

Скенирање започиње итерацијом кроз ставке у свакој секвенци и додавањем идентификатора трансакција (tid) у мапу где је та ставка кључ уколико се појави. Када се заврши итерација кроз све ставке у једној секвенци, алгоритам уклања сва појављивања у тој секвенци за сваку ставку чија је подршка мања од минималне подршке. Након завршетка првог скенирања базе података, мапа ставки садржи сва појављивања ставки у секвенцама у којима су учестале.

Следећи корак је уклањање ставки које имају вредност мању од минималног периодичног односа секвенцијалног правила (скраћено минимални однос) из мапе ставки. Затим алгоритам додељује информације о величини базе података и дужинама секвенци одговарајућим променљивама, чиме постаје непотребно реферисање на оригиналну базу података. На тај начин мапа ставки сада садржи све неопходне информације из базе података.

Procedure 1: Database scan procedure’s pseudo-code

Database scan ()

// list of global variables used in method: database, //mapItemsOccur, minSup, minRa

```
1.  mapItemsOccur ⟵ ∅
2.  FOR each sequence s in database:
3.    FOR each itemset it in s
4.      FOR each item i in it
5.        IF mapItemsOccur.Get(i)==∅ THEN mapItemsOccur.Add(i)
6.        IF mapItemsOccur.Get(i).Get(s.Sid) == ∅ THEN
      mapItemsOccur.Get(i).Add(s.Sid).Add(it.Tid)
7.        ELSE mapItemsOccur.Get(i).Get(s.Sid).Add(it.Tid)
8.      END FOR (line 4)
9.    END FOR (line 3)
10.   FOR each itemI in mapItemsOccur:
11.     IF ((mapItemsOccur.Get(i).Get(s.Sid).Length / database.Get(s).Length < minSup) OR
      ((mapItemsOccur.Get(i).Get(s.Sid)Length) < 2) THEN
12.       FOR each tid in mapItemsOccur.Get(i).Get(s.Sid):
            s.Get(tid).Remove(i)
          mapItemsOccur.Get(i).Remove(s.Sid)
            IF mapItemsOccur.Get(i).Length == 0 THEN mapItemsOccur.Remove(i)
13.       END FOR (line 15)
14.   END FOR (line 13)
15. END FOR (line 2)
16. FOR each i in mapItemsOccur
17.   IF (mapItemsOccur.Get(i).Length / database.Length) ≤ minRa) THEN
      mapItemsOccur.Remove(i)
18. END FOR (line 20)
```

Представљена процедура је дизајнирана да оствари неколико кључних предности у погледу перформанси. Најпре, ставке које не испуњавају захтевани праг учесталости и минималну заступљеност елиминишу се већ током скенирања базе. Овакво решење је ефикасније јер искључује потребу за накнадном обрадом података. Осим тога, благовремено чишћење мапе од сувишних ставки директно скраћује време извршавања услед мањег броја итерација у петљама и истовремено смањује заузеће меморије.

Једна од предности коришћења мапе ставки уместо директног приступа бази података јесте избегавање поновног скенирања базе при сваком покушају проширивања правила. У оквиру RuleGrowth алгоритма, свака секвенца се скенира како би се пронашла ставка која може проширити правило, односно ставка која се појављује у истој секвенци као и постојеће правило. Насупрот томе, у предложеном алгоритму, ове информације су већ садржане у мапи ставки. Алгоритам једноставно преузима листу идентификатора секвенци (SID листа) за сваку

ставку како би идентификовао оне које се појављују у истом низу, чиме се остварује значајна уштеда времена.

4.1.2. Главна процедура

У овом одељку детаљније је описана главна процедура алгоритма, чији је псеудокод представљен у поглављу 4. Након скенирања базе, уводе се две променљиве како би се избегла потреба за додатним претраживањем података: databaseSize, која бележи број секвенци у бази, и seqLengths, мапа у којој се чувају дужине свих секвенци на основу њихових идентификатора (SID). У следећој фази, алгоритам приступа генерисању правила која се састоје од по једне ставке са леве и десне стране. Овај процес се ослања на претходно пронађене учестале ставке, по принципу сличном алгоритму RuleGrowth (Fournier-Viger et al., 2011). Ове ставке ћемо у даљем тексту означавати као I и J. Притом се одбацују случајеви у којима је ставка I већа од ставке J, или су оне идентичне. За сваки пар ставки I и J, алгоритам на једноставан начин утврђује у којим се заједничким секвенцама оне појављују тако што директно из мапе ставки преузима податке о њиховим идентификаторима (SID).

Алгоритам кроз следеће кораке за сваку заједничку секвенцу у којој се појављују ставке I и J, проверава да ли формирано правило задовољава услове периодичности унутар те секвенце. Први корак подразумева позивање методе get_Single_Item_Rule_Occur како би се проверило да ли ставке могу да формирају правило I⇒J у датој секвенци. Резултат ове методе је листа валидних појављивања правила у тој секвенци. Ако резултат ове методе није празан скуп, односно ако ставке могу да формирају правило у посматраној секвенци, алгоритам прелази на следећи корак. Тада се позива метода is_PIS_Rule како би се испитало да ли је потенцијално правило периодично унутар те секвенце. Ова метода враћа низ израчунатих вредности параметара правила.

У трећем кораку се анализирају добијени резултати. Уколико низ садржи барем једну вредност, алгоритам додаје идентификатор (SID) текуће секвенце у листу за потенцијално проширивање правила (у даљем тексту: листа за проширење). Уколико су присутне све три вредности, у мапу периодичних секвенци се додаје запис који обједињује SID секвенце и припадајуће вредности параметара, чиме се означава да је правило периодично унутар те секвенце. Листа секвенци за проширење се користи као параметар приликом позивања метода за проширење, уместо листе заједничких секвенци ставки I и J. Будући да садржи само оне заједничке секвенце у којима правило задовољава минималну подршку, њена величина је мања од листе свих заједничких секвенци, што доводи до смањења времена извршавања и потрошње меморије смањењем броја итерација у петљама метода за проширење.

Мапа периодичних секвенци користи се за чување свих секвенци, заједно са одговарајућим параметрима, у којима се текуће правило I ⇒ J сматра периодичним унутар секвенце. Ова мапа се затим користи за одређивање да ли ће правило бити класификовано као PIS правило. Ови кораци су неопходни јер су правила која се траже периодична на нивоу појединачних секвенци, стога алгоритам мора испитати појављивање и специфична својства правила у свакој од њих. Поступак се понавља за сваку заједничку секвенцу у којој се налазе ставке I и J, како би се у потпуности попуниле одговарајуће листе за проширење и мапе периодичности.

Након што се обради и последња секвенца у којој се појављује ставка I, алгоритам израчунава однос периодичности правила тако што број секвенци из мапе периодичности подели са укупним бројем секвенци у бази. Уколико одговарајућа листа за проширење садржи секвенце и ако је овај однос већи од минималног прага (minRa), правило испуњава услов за даље проширење.

Листа за проширење садржи секвенце у којима правило I⇒J има довољну подршку. Када се правило прошири додатном ставком, новонастало правило може имати само мању подршку од првобитног (Fournier-Viger et al., 2011). Стога, проширивање правила које већ не задовољава минималну подршку може резултирати само новим правилима која такође не би могла бити PIS правила. Додатно, поменути параметар односа пружа информацију о проценту секвенци у којима је правило I⇒J PIS правило. Додавање нових ставки у правило може само смањити тај проценат. Стога, оба услова морају бити испуњена како би се приступило проширивању правила и избегли сувишни позиви метода и итерације.

Осим тога, уколико мапа периодичних секвенци није празна, правило I⇒J представља унутар-секвенцијално периодично правило које је заједничко за више секвенци, те се позива метода save_Rule како би се оно уписало у излазну датотеку. Без обзира на овај последњи услов, алгоритам позива методу ExpandLeft ради проширења правила са леве стране. Након тога, креира се мапа подршке ставке I у свим секвенцама, која се састоји од идентификатора (SID) и броја појављивања те ставке у датој секвенци. Ово се ради како би се избегло рекурзивно израчунавање унутар методе ExpandRight, која се позива након креирања мапе.

Овај поступак се у главној методи понавља за сваки пар ставки I и J. Извршавање алгоритма се завршава када више нема преосталих ставки.

4.1.3. ExpandLeft

Псеудокод ове процедуре дат је у наставку, након чега следи њено детаљније објашњење.

Procedure 2: ExpandLeft procedure's pseudocode

```
ExpandLeft (itemsetI, itemsetJ, sidsIJ)
// list of global variables used in method: databaseSize, //seqLengths, mapItemsOccur, minRa
1.  map_items_sidsIC ⟵ ∅
2.  FOR each itemC in mapItemsOccur
3.    ic ⟵ ItemC.Item
4.    IF (itemsetI.Get(ic)) ≠ ∅ OR (itemsetJ.Get(ic)) ≠ ∅ THEN CONTINUE (line 2)
5.    FOR each i in itemsetI
6.      IF i > ic THEN CONTINUE (line 2)
7.    END FOR (line 6)
8.    sidsIC ⟵ ∅
9.    FOR each s in sidsIJ
10.     IF itemC.Get(s) ≠ ∅ THEN sidsIC.Add(s)
11.   END FOR (line 11)
12.   IF (sidsIC ≠ ∅) AND ((sidsIC.Length / databaseSize) > minRa)) THEN
      map_items_sidsIC.Add(ic).Add(sidsIC)
```

13. END FOR (line 2)
14. FOR each map_item in map_items_sidsIC
15. ic ⟵ map_item.Item
16. occur_ic ⟵ ∅
17. occur_cj ⟵ ∅
18. itemsetIC ⟵ {itemsetI, ic}
19. num_occur_ic ⟵ CALL countIC (itemsetIC, map_item)
20. IF (num_occur_ic == ∅) OR ((num_occur_ic.Length / databaseSize) < minRa)) THEN CONTINUE (line 18)
21. occur_icj ⟵ CALL getRuleOccur (itemsetIC, itemsetJ, map_item)
22. IF (occur_icj == ∅) OR ((occur_icj.Get(0).Get(0).Length / databaseSize) ≤ minRa)) THEN CONTINUE (line 18)
23. FOR s in map_item
24. IF occur_icj.Get(s) ==∅ THEN CONTINUE (line 29)
25. IF occur_icj.Get(s).Get(0).Get(0).Length < 2 THEN CONTINUE (line 29)
26. occur_ic.Add(s).Add(occur_icj.Get(s).Get(0))
27. occur_cj.Add(s).Add(occur_icj.Get(s).Get(1))
28. END FOR (line 29)
29. param_value_icj ⟵ ∅
30. sids_for_expansion ⟵ ∅
31. FOR each s in occur_ic
32. param_value ⟵ CALL is_PIS_Rule(s.Get(0), occur_cj.Get(s).Get(1), num_occur_icj.Get(s), seqLengths.Get(s))
33. IF param_value.get(0) ≠ ∅ THEN sids_for_expansion.Add(s)
34. IF param_value.Get(0) ≠ ∅ AND param_value.Get(1) ≠ ∅ AND param_value.Get(2) ≠ ∅ THEN
35. param_value_icj Add(s).Add(param_value)
36. END FOR (line 39)
37. ra_icj ⟵ param_value_icj.Length / databaseSize
38. IF (sids_for_expansion ≠ ∅) AND (ra_icj > minRa) THEN
39. IF param_value_icj ≠ ∅ THEN CALL Save_Rule (ra_icj, itemsetIC, itemsetJ, param_value_icj)
40. CALL ExpandLeft (itemsetIC, itemsetJ, sids_for_expansion)
41. END FOR (line 18)

Приликом позивања ове методе, као улазни аргументи прослеђују се скупови ставки I и J, као и листа за проширење за правило I⇒J. Алгоритам врши провере како би пронашао ставке садржане у мапи ставки које могу проширити леву страну правила. Ставке које задовољавају прелиминарне услове, сличне онима из главне процедуре, у даљем тексту се називају ставка C или нова ставка.

Након тога, алгоритам тражи секвенце које се истовремено налазе у листи за проширење и у листи секвенци које садрже нову ставку. Уколико је број заједничких секвенци довољно велики, тако да је периодични однос секвенцијалног правила већи од минималног прага, те ставке се додају у мапу. Алгоритам на овај начин елиминише све ставке C које би формирале правило IC⇒J које не испуњава критеријум минималног односа и самим тим не би могле бити периодична унутар-секвенцијална правила.

Сврха ове мапе, која се назива „мапа ставки заједничких секвенци", јесте да сачува ставке C, заједно са листом идентификатора (SID) секвенци у којима се оне појављују када алгоритам покушава да прошири правило тим ставкама. Креирање ове мапе ће свести број итерација у наредном процесу на минимум, јер се број ставки C редукује: уместо свих ставки садржаних у секвенцама из листе за проширење, разматрају се само оне ставке које имају потенцијал да прошире правило I⇒J у периодично унутар-секвенцијално правило.

Након што су све ставке проверене, алгоритам покушава да прошири правило I⇒J ставкама C које се налазе у мапи ставки заједничких секвенци, креирајући правило IC⇒J. За сваку ставку C из поменуте мапе, алгоритам извршава одређене кораке. Неиспуњење услова у било ком од ових корака доводи до преласка на следећу ставку C у овој мапи.

Први корак је позивање методе countIC како би се израчунао број појављивања скупа ставки IC унутар мапе ставки заједничких секвенци. Затим се проверава да ли број секвенци у којима се појављује скуп ставки IC задовољава праг минималног односа како би се прешло на следећи корак. Уколико скуп ставки IC нема довољан број секвенци да би испунио праг минималног односа, потенцијално правило IC⇒J би могло имати само још мање секвенци, те је покушај проширивања правила ставком C неоправдан.

Други корак је позивање методе getRuleOccur ради добијања мапе појављивања правила IC⇒J у свим заједничким секвенцама скупа ставки I и ставке C. У овом кораку алгоритам издваја секвенце у којима је IC⇒J заправо PIS правило, што смањује број секвенци које треба обрадити у четвртом кораку. Да би се наставило даље, резултат ове методе не сме бити празан, а број пронађених секвенци мора бити већи од прага минималног односа.

Трећи корак обухвата креирање две засебне мапе појављивања за леву и десну страну правила IC⇒J, на основу података добијених у претходном кораку. Том приликом се из даље обраде искључују све секвенце у којима се правило појављује мање од два пута. На самом крају овог корака, преостаје само да се још једном потврди да је задовољен праг минималног односа.

Четврти корак подразумева израчунавање вредности параметара за сваку појединачну секвенцу на основу мапе појављивања леве стране правила IC⇒J, коришћењем методе is_PIS_Rule. Уколико је задовољен праг подршке, алгоритам додаје тренутну секвенцу у листу за проширење правила IC⇒J. Ако су притом испуњени сви остали прописани прагови, секвенца се додаје и у мапу периодичних секвенци, односно мапу секвенци у којима је правило IC⇒J периодично, при чему кључеве представљају идентификатори (SID) тих секвенци.

Након што се обради и последња секвенца која садржи правило IC⇒J, алгоритам израчунава однос периодичности правила тако што број секвенци из мапе периодичних секвенци подели са укупним бројем секвенци. Наставак поступка условљен је тиме да правило поседује ставке у листи за проширење, као и да израчунати однос премаши минимални праг. Поред тога, како би правило постало коначни излаз, односно периодично унутар-секвенцијално правило

заједничко за више секвенци, његова мапа периодичних секвенци мора садржати елементе. Независно од испуњености претходног услова, метода ExpandLeft се поново позива ради даљег проширења правила улево. Ови кораци се понављају за сваку ставку C из мапе ставки заједничких секвенци све док се проширење правила I⇒J не покуша са сваком ставком из те листе, након чега се метода завршава.

4.1.4. ExpandRight

Ова метода је веома слична методи ExpandLeft, те ћемо се у наставку фокусирати на кључне разлике. Подразумева се да нова ставка C проширује десну страну правила, чиме се формира правило I⇒JC. Сходно томе, свуда где се у методи ExpandLeft појављује скуп ставки IC, овде га можемо заменити скупом JC.

Улаз ове методе садржи још један параметар у односу на ExpandLeft, а то је мапа sid-ова са бројем појављивања скупа ставки I. Овај параметар је уведен како би се избегло поновно израчунавање броја појављивања скупа ставки I у свакој секвенци, за сваки рекурзивни позив методе ExpandRight. Он је неопходан за израчунавање поузданости правила у свакој секвенци. Ово није било применљиво у методи ExpandLeft, јер се скуп ставки I током проширивања мењао, па је био различит у сваком рекурзивном позиву.

Кључна структурна разлика је у томе што ExpandLeft врши директне рекурзивне позиве, док ExpandRight најпре позива методу ExpandLeft, а тек затим наставља са сопственим рекурзивним позивима.

У наставку ћемо детаљније објаснити претходно поменуте методе које алгоритам користи. Разумевање ових метода је од пресудног значаја за схватање самог алгоритма, као и начина на који он обрађује специфичну унутар-секвенцијалну природу тражених правила. Методе ћемо изложити према редоследу њиховог појављивања у алгоритму.

4.1.5. Метода get_Single_Item_Rule_Occur

Метода get_Single_Item_Rule_Occur као улазне параметре прима појављивања ставки I и J у текућој секвенци, као и дужину те секвенце. Она користи две помоћне променљиве чије се вредности мењају током извршавања методе. Вредност прве променљиве представља последњи tid завршетка правила I ⇒ J, а њена сврха је да спречи погрешно израчунавање појављивања правила тако што означава tid од ког је дозвољено тражити ново појављивање. На пример, тиме се избегава ситуација у којој би алгоритам пребројао исто појављивање ставке I са два различита појављивања ставке J, иако постоји само једно појављивање са јединственим tid вредностима.

Сврха друге дефинисане променљиве је да се умањи број пролаза кроз појаве ставке J, тако што се њена вредност поставља на последњи обрађени tid те ставке. За сваку трансакцију скупа ставки I чији је tid већи од вредности прве променљиве, метода затим тражи наредно појављивање скупа ставки J, почев од те трансакције.

Итерација кроз трансакције ставке J мора да започне од вредности друге дефинисане променљиве. Када се такво појављивање ставке J пронађе, сматра се да је у тој секвенци пронађено једно појављивање правила I ⇒ J. Идентификатори трансакција (tid) ставки I и J које чине то појављивање правила додају се, редом, у посебне листе појављивања леве и десне стране правила.

Када више нема појављивања ставке I у посматраној секвенци и када је однос броја tid вредности правила у било којој од наведених листа и дужине секвенце већи од минималне подршке, метода враћа листе појављивања леве и десне стране правила у тој секвенци. Сходно томе, излаз ове методе је или скуп појављивања правила у датој секвенци, уколико је оно потенцијални кандидат за периодично унутар-секвенцијално правило у тој секвенци, или null вредност ако не испуњава потребне услове.

4.1.6. Метода is_PIS_Rule

Метода is_PIS_Rule има више улазних аргумената: листе појављивања ставки I и J које чине правило I ⇒ J, број појављивања ставке I у тренутној секвенци, и дужину те секвенце. Ова метода се такође користи за проверу правила која имају више ставки у левој и/или десној страни, при чему се по потреби појединачне ставке у улазним аргументима замењују одговарајућим скуповима ставки и обрнуто. Детаљна објашњења и псеудокод су представљени у наставку.

Procedure 3: is_PIS_Rule procedure's pseudocode

is_PIS_Rule (occurI, occurJ, num_tids_I, lengthOfSeq)

// global variables used in method: minSup, minConf, maxStd

1. supIJ ⟵ occurI.Length / lengthOfSeq
2. IF (supIJ ≤ minSup) THEN RETURN ∅
3. confIJ ⟵ occurI.Length / num_tids_I
4. IF confIJ ≤ minConf THEN confIJ ⟵ ∅
5. stdIJ ⟵ CALL getStd (occurI, occurJ, lengthOfSeq)
6. IF stdIJ ≥ maxStd THEN stdIJ ⟵ ∅
7. RETURN {supIJ, confIJ, stdIJ}

Прво, израчунава се подршка правила у текућој секвенци дељењем броја појављивања скупа ставки I са дужином те секвенце. Уколико је њена вредност већа од прага минималне подршке, она се додаје у низ вредности параметара. У супротном, враћа се null. Друго, поузданост правила у текућој секвенци се израчунава дељењем броја појављивања скупа ставки I који формирају правило са укупним бројем појављивања скупа ставки I. Уколико је вредност већа од прага минималне поузданости, она се додаје у низ вредности параметара. У супротном, у низ вредности параметара се додаје null. На крају, стандардна девијација правила у текућој секвенци се израчунава позивањем помоћне методе. Резултат те методе је вредност стандардне девијације, која се поново проверава да ли је мања од максимално дозвољене стандардне девијације и уколико јесте, додаје се у низ вредности параметара. Ако услов није испуњен, поново се додаје null у низ параметара.

Може се приметити да метода различито поступа када се проверавају вредности параметара у односу на прагове minSup, minConf и maxStd. Уколико подршка не испуни одговарајући праг, метода се одмах прекида и враћа null. Када праг минималне поузданости или праг максималне

стандардне девијације није задовољен, метода се не прекида, већ се у излазни низ додаје вредност null. Разлог за овакво понашање методе лежи у две чињенице. Прва је да неиспуњавање прага подршке значи да правило у датој секвенци не може бити периодично унутар-секвенцијално правило и да се не може проширивати у наредним корацима алгоритма, па даља обрада таквог кандидата нема смисла. Друга чињеница је да се на овај начин једноставно разликују секвенце у којима правило не може бити проширено од оних у којима је проширење могуће, али правило не испуњава услове да буде периодично унутар-секвенцијално правило, што се утврђује провером да ли је излаз методе null. На крају, излаз ове методе представља низ који садржи вредности параметара појављивања правила у текућој секвенци.

4.1.7. Метода getNextIC

Метода getNextIC је помоћна метода коју не позивају директно главни алгоритам нити методе ExpandLeft и ExpandRight, већ остале функције унутар система. Она као улазне параметре прима листу ставки из скупа IC, идентификатор секвенце (SID) и вредност променљиве „претходна позиција". Ова променљива одговара последњем TID-у пронађеног појављивања скупа IC у методи која је иницирала позив. За сваку ставку из скупа IC метода тражи следеће појављивање почевши од задатог TID-а. Уколико за било коју ставку не постоји појављивање у опсегу од „претходне позиције" до краја секвенце, метода враћа вредност null. Ово је неопходно јер се појављивање целог скупа IC региструје само ако су пронађене све појединачне ставке које га чине.

Метода проналази прво и последње појављивање скупа ставки IC међу појављивањима свих појединачних ставки које тај скуп чине. Она формира низ у којем први елемент представља TID првог појављивања, док је други елемент TID последњег појављивања. Тај низ представља излазни резултат ове методе. Први и последњи TID се враћају зато што они означавају почетак и крај појављивања скупа ставки IC, који ће касније представљати леву страну правила. Према томе, циљ ове методе је да пронађе следеће појављивање скупа ставки IC или да враћањем вредности null сигнализира да појављивања више нема.

4.1.8. Метода countIC

Метода countIC прима следеће улазне аргументе: листу ставки која обухвата скуп I и ставку C обједињене у јединствену листу под називом скуп ставки IC, као и листу заједничких секвенци скупа I и ставке C, односно секвенци које садрже скуп IC. За сваку појединачну секвенцу, ова метода израчунава број појављивања скупа ставки IC. Она позива методу getNextIC како би за сваку секвенцу добила низ који садржи почетак и крај појављивања скупа IC. Све док метода getNextIC не врати вредност null, што би значило да су пронађена сва појављивања скупа IC у тренутној секвенци, алгоритам наставља са њеним позивањем и сваки пут увећава бројач појављивања скупа ставки IC.

Након тога, метода проверава да ли скуп ставки IC има довољан број појављивања у тој секвенци како би задовољио праг минималне подршке. Овај корак се спроводи како би се елиминисале секвенце у којима ставка C не може да прошири правило, и то пре самог израчунавања појављивања тог проширеног правила. Такође, ово омогућава главним деловима алгоритма да одбаце ставку C као кандидата за проширење правила IC⇒J на основу броја секвенци које имају жељену подршку. Уколико је подршка довољна, број појављивања у

тренутној секвенци се, заједно са одговарајућим идентификатором (SID), додаје у мапу. Када се обраде све секвенце, ова мапа ће садржати број појављивања скупа IC за сваку тражену секвенцу појединачно и биће враћена као излазни резултат методе countIC.

4.1.9. Метода getRuleOccur

Метода getRuleOccur прима следеће улазне параметре: листу ставки у скупу IC, листу ставки у скупу J, као и листу заједничких секвенци за скуп I и ставку C. За сваку појединачну секвенцу, ова метода проналази валидна појављивања правила IC⇒J кроз низ корака који су у наставку описани текстом и псеудокодом.

Procedure 4: getRuleOccur procedure's pseudocode

```
getRuleOccur (itemsetIC, itemsetJ, sidsIC)
// list of global variables used in method: seqLengths, minSup
1.  occurIJ ⟵ ∅
2.  FOR each s in sidsIC
3.    previousPos ⟵ -1
4.    cond ⟵ true
5.    firstOccI ⟵ ∅
6.    lastOccI ⟵ ∅
7.    firstOccJ ⟵ ∅
8.    lastOccJ ⟵ ∅
9.    WHILE (cond)
10.     firstI ⟵ -1
11.     lastI ⟵ -1
12.     firstLastI ⟵ CALL getNextIC (itemsetIC, s, previousPos)
13.     IF firstLastI == ∅ THEN
14.       cond ⟵ false
15.       BREAK (line 9)
16.     ELSE
17.       firstI ⟵ firstLastI.Get(0)
18.       lastI ⟵ firstLastI.Get(1)
19.     positionI ⟵ lastI
20.     firstLastJ ⟵ CALL getNextIC (itemsetJ, s, positionI)
21.     IF firstLastJ == ∅ THEN
22.       cond ⟵ false
23.       BREAK (line 9)
24.     ELSE
25.       firstJ ⟵ firstLastJ.Get(0)
26.       lastJ ⟵ firstLastJ.Get(1)
27.     firstOccI.Add(firstI)
28.     lastOccI.Add(lastI)
29.     firstOccJ.Add(firstJ)
30.     lastOccJ.Add(lastJ)
31.     previousPos ⟵ lastJ
32.   END WHILE (line 9)
33.   IF (firstOccI.Length < 2) THEN CONTINUE (line 2)
34.   IF (firstOccI.Length / seqLengths.Get(s) < minSup) THEN CONTINUE (line 2)
```

35. occurI ⟵ {firstOccI, lastOccI}
36. occurJ ⟵ {firstOccJ, lastOccJ}
37. occurrences ⟵ {occurI, occurJ}
38. occurIJ.Add(s).Add(occurrences)
39. END FOR (line 2)
40. RETURN occurIJ

Први корак обухвата дефинисање променљиве која чува вредност последњег TID-а претходно пронађеног појављивања правила IC⇒J. Означавајући крај последњег појављивања, ова променљива, под називом „претходна позиција", користи се за спречавање проналаска дуплираних или невалидних правила. На тај начин се, на пример, занемарује правило чија се лева страна налази пре десне стране већ пронађеног правила. Без ове променљиве, појављивање правила би било урачунато чак и у случајевима када је забрањено бројање испреплетаних појављивања.

У другом кораку, метода тражи следеће појављивање сваке појединачне ставке из скупа IC, почевши од вредности TID-а коју чува променљива „претходна позиција", и то позивањем методе getNextIC. Ова метода обезбеђује следеће појављивање леве стране правила. Трећи корак представља проверу да ли је низ добијен у другом кораку различит од null. Уколико је вредност null, то значи да у секвенци више нема појављивања скупа ставки IC, те се даља претрага обуставља.

Четврти корак подразумева проналажење најмањег и највећег TID-а унутар добијеног низа, који се редом означавају као први и последњи TID појављивања скупа ставки IC. Вредност последњег TID-а се додељује променљивој „позиција I", која означава крај леве стране правила. Ова променљива се користи у наредном кораку како би се осигурало да се десна страна правила појављује након леве.

У петом кораку се поново позива метода getNextIC ради тражења следећег појављивања сваке ставке из скупа J, почевши од вредности TID-а у променљивој „позиција I". Као и у трећем кораку, уколико резултат ове методе није null, поступак се враћа на други корак. Шести корак се извршава након прекида претраге у трећем или петом кораку. Ако је подршка правила за тренутну секвенцу већа од прага минималне подршке, за њу се креира низ појављивања правила IC⇒J и смешта у мапу, при чему кључ представља одговарајући SID. Кораци се понављају све док се не обраде све секвенце са листе заједничких секвенци. Када се листа секвенци исцрпи, мапа појављивања правила се враћа као излазни резултат.

У овом одељку описани су главни делови предложеног алгоритма и пружена су детаљна објашњења припадајућих метода. Посебна пажња посвећена је процедурама, концептима и условима који се значајно разликују или су потпуно другачији од оних у оквиру RuleGrowth алгоритма. На крају, настојали смо да пружимо бољи увид у структуру предложеног алгоритма, као и у стратегије коришћене током његовог развоја како би се осигурала већа ефективност и ефикасност у раду.

4.2. MEHAUPM: Модификације и обрада мешовитих вредности

Како би се на прави начин одговорило на изазов тачне обраде динамичких промена вредности корисности (код ставки са мешовитом корисношћу), у изворни EHAUPM алгоритам уведено је неколико кључних измена. Иако ове преправке нису обимне, оне доводе до значајне разлике у крајњим исходима. Ове измене су биле пресудне за омогућавање прецизног прорачуна корисности и стварање скупова од ставки чији јединични профит варира између позитивних и

негативних вредности. Појединости ових преправки приказане су у псеудокоду, док су разлози за њихово увођење детаљније образложени у наставку текста.

Procedure 5: MEHAUPM algorithm and structures pseudocode

Abbreviations

---

auub (Average Upper Utility Bound)

MAUList (Modified Average Utility List)

MAUEntry: a record containing (tid, utility, rmu, remu) for an itemset.

EAUCM (Estimated Average Utility Co-occurrence Matrix)

tmu (transaction-maximum utility)

minUtility: The minimum average utility threshold.

remu (Remaining Estimated Maximum Utility).

rmu (Remaining Maximum Utility)

sumReal: The actual sum of utilities for an itemset.

sumUtils: The sum of utilities for an itemset without negative values.

sumOfRemu: The sum of `remu` values for an itemset.

MAULIST data structure (MAUEntry):

Fields:

item, sumOfRemu, sumOfRmu, sumUtils

//MODIFICATION 4//: new field sumReal

addElement function steps:

1. adds the corresponding field remu value of the entry to sumOfRemu
2. adds the corresponding field rmu value of the entry to sumOfRmu
3. //MODIFIED//: add corresponding field utility value of entry to sumUtils only if positive.
4. //MODIFIED//: add corresponding field utility value of MAUEntry.sumReal without condition

---

MEHAUPM (Input_file_path, output_file_path, minUtility)

MAIN Algorithm Steps:

1. Calculate the initial auub for each item:

   For each transaction, compute its maximum utility (tmu).

   //MODIFICATION 1//: If tmu is negative, set tmu to 0.

   Add tmu to the auub of each item in the transaction.

2. Filter and sort 1-itemsets:

   Keep only items whose auub is greater than or equal to `minUtility`.

   //MODIFICATION 2//: Sort these items in descending order of their auub. If auub's are equal, sort by item ID in descending order.

3. Construct MAULists and EAUCM:

   For each transaction:

      *Create a revised transaction with only relevant items, sorted by the `compareItems` logic.

      *For each item in the revised transaction:

         **Calculate `rmu` and `remu`.

         ** //MODIFICATION 3//: Add the item's utility to `remu` only if the utility is positive.

         **Store `rmu`, `remu`, and other values in the item's MAUList.

      **//MODIFICATION 4 reference//: When the new element is added to MAUList (MAUEntry), the modified structure of MAUList is used. See modification 4 in the MAUList structure for more information.

      *For each pair of items in the revised transaction:

         **Update the EAUCM structure with the transaction's maximum utility.

4. Recursively search for High Average Utility Itemsets:

   For each itemset `X` being considered:

      *//MODIFICATION 5//: If `X`'s calculated average utility (based on `sumReal`) divided by itemset length meets `minUtility`, then `X` is a High Average Utility Itemset. Record it.

      *//MODIFICATION 6//: Apply a pruning rule: If `X`'s upper bound for average utility (based on `sumutils` and `sumOfRemu`) is less than `minUtility`, then prune this branch.

      * If `X`'s `sumOfRmu` (another upper bound) meets `minUtility`:

         ** Consider extending `X` with subsequent items `Y`.

         ** For each such `Y`:

*** Check the pre-calculated auub of the 2-itemset `(X, Y)` from EAUCM. Prune if too low.

*** Construct the MAUList for `XY` by merging `X`'s and `Y`'s MAULists.

*** During the merge, if a tid from `X`'s MAUList is processed but not found in `Y`'s:

****//MODIFICATION 7//: When updating `sumOfRemu` for pruning, subtract the item's utility only if it is positive.

****//MODIFICATION 8//: If the maximum value of sumOfRemu or sumOfRmu is below the minimum utility, stop trying to expand this itemset.

*** If `XY`'s MAUList can be formed (i.e., not pruned), recursively call the search with `XY`.

Прва измена осигурава да максимална корисност ставке у трансакцији никада не буде негативна. Ово прилагођавање је пресудно јер збир ових максималних вредности кроз све трансакције у којима се ставка појављује одређује њен auub праг. Тај праг се затим користи за рано одбацивање неперспективних ставки, спречавајући стварање већих скупова од k ставки. Ограничавањем било које негативне вредности максималне корисности на нулу, алгоритам одржава оптимистичну горњу границу за сваку ставку. Без ове границе, ставке или мешовити скупови са негативним максималним вредностима корисности могли би бити прерано искључени, иако би могли допринети позитивној корисности у одређеним трансакцијама у којима се појављују одговарајући скупови од k ставки. Будући да је наш циљ да у коначне резултате укључимо и негативне и мешовите скупове ставки, под условом да остварују довољно високу корисност у релевантним трансакцијама, примењујемо ову стратегију свођења на нулу како бисмо осигурали задржавање свих потенцијално вредних комбинација.

Друга преправка односи се на обртање редоследа којим се појединачне ставке разматрају приликом проширења. Конкретно, метода compareItems ставке сортира опадајуће, уместо претходног подразумеваног растућег поретка. Будући да су остале измене довеле до нешто блажих граница за одбацивање скупова ставки, ова промена је безбедно уведена како би се те границе пооштриле, а да се притом не ризикује прекомерно одбацивање или пропуштање било ког скупа ставки са високо-просечном корисношћу (HAUI). Захваљујући томе, овако рано одбацивање не нарушава тачност резултата.

Трећа измена у алгоритму редефинише начин израчунавања преостале корисности уводећи збир искључиво ненегативних вредности корисности уместо текућег максимума. У измењеној верзији, преостала корисност се израчунава сабирањем само позитивних вредности корисности ставки које унутар трансакције долазе након тренутне, уместо укључивања и негативних вредности. Изостављање негативних вредности осигурава да преостала корисност остане стабилна и представља оптимистичну горњу границу.

Уколико би алгоритам користио стандардну дефиницију преостале корисности из области откривања скупова ставки високе корисности, која подразумева додавање и негативних вредности, могло би доћи до прераног одбацивања грана у стаблу претраге. То би се дешавало у ситуацијама када укупна корисност преосталих ставки задовољава праг, док је ниједна појединачна ставка не премашује, чиме би се занемарили валидни скупови ставки високо-просечне корисности (HAUI). Својство антимонотоности, према којем скупови који нису HAUI не могу произвести HAUI скупове кроз проширења, не важи за негативне и мешовите вредности корисности.

Наша редефинисана bound мера такође прецизније представља укупну остварив корисност код проширења са више ставки, у поређењу са процењеном преосталом корисности која се заснива на максимуму. С обзиром на присуство негативних вредности у скуповима података, bound мера преостале корисности заснована на максимуму би потценила стварну преосталу корисност. Поред тога, њена вредност би показивала недоследна осциловања (повремено би расла, а повремено опадала) уместо да се доследно смањује са даљим проширењима скупова ставки. Како бисмо предупредили ове потешкоће, усвојили смо овај прикладнији начин израчунавања преостале корисности.

У трећем кораку алгоритма, четврта преправка обухвата реструктурирање MAUList (Modified Average Utility List) увођењем новог поља, sumReal. Метода addElement је ажурирана како би динамички израчунавала sumReal и прецизирала израчунавање поља sumUtils. Атрибут sumReal бројчано исказује стварну корисност скупа ставки, изричито узимајући у обзир негативне вредности, док sumUtils представља корисност без укључивања негативних вредности. Ове две засебне мере корисности имају допунске улоге у алгоритму: sumUtils се користи како би се избегло прерано одбацивање перспективних скупова ставки под важећим условима, док се sumReal примењује током завршне фазе провере како би се утврдило да ли скуп ставки испуњава услове да постане скуп ставки високо-просечне корисности (HAUI). У псеудокоду, на месту означеном са „reference modification 4”, ова измена је примењена приликом додавања новог поља у ову листу.

Пета преправка, која се налази у четвртом кораку, обухвата исписивање скупова ставки који су идентификовани као HAUI на основу њихове sumReal вредности подељене са дужином скупа ставки. У овом кораку се користи тачан прорачун корисности, јер он служи као коначна валидација за HAUI, а не као механизам за одбацивање (pruning). Због тога не постоји ризик од превелике или премале процене стварне вредности корисности.

Шеста преправка, примењена у четвртом кораку алгоритма, ревидира услов за одбацивање (pruning). Претходно је алгоритам користио сувише благу bound меру, израчунату дељењем збира корисности ставки са потенцијалном дужином скупа, чему је додавана преостала корисност без дељења. Ово замењујемо строжом bound мером, која најпре сабира само позитивне корисности ставки и позитивне преостале корисности, а затим цео тај збир дели са дужином образаца. Изостављање негативних вредности из оба збира омогућава нам да директно делимо са дужином образаца, што је стандардна пракса, уместо да мешамо просечне и укупне вредности.

Будући да су обе компоненте наше нове bound мере прецењене вредности (overestimates), не постоји ризик од одбацивања било ког перспективног скупа ставки. Насупрот томе, према старом услову за одбацивање, многи скупови који садрже негативне вредности били би прерано одбачени, упркос томе што је тадашња мера била преблага за чисто позитивне случајеве. Резултат је мањи број рекурзивних позива и мања потрошња меморије, пошто више грана задовољава овај строжи критеријум и раније долази до прекида.

Седма преправка у четвртом кораку односи се на условно одузимање корисности ставке од вредности преостале корисности скупа ставки, само ако је корисност те ставке позитивна. Ово одузимање се врши током обраде скупа ставки како би се он припремио за потенцијална проширења. Будући да се вредност преостале корисности сада састоји искључиво од позитивних вредности, ово условно одузимање спречава губитак валидних проширења скупова ставки, чиме се по самој природи алгоритма онемогућава одузимање негативних корисности.

У четвртом кораку, осма преправка обухвата измену услова за заустављање даљих покушаја проширења скупа ставки. Раније се услов за заустављање активирао ако би минимум збира преостале корисности или збира преостале максималне корисности пао испод прага

минималне корисности. Ова измена уводи блажи услов, захтевајући да максимум било ког од ова два збира буде испод прага како би се скуп ставки одбацио. Постављањем услова за одбацивање на приступу заснованом на максимуму уместо на минимуму, осигуравамо да ниједан валидан HAUI не буде одбачен. Овај приступ је конзервативнији, али гарантује тачност, нарочито зато што негативне вредности у овим збировима сводимо на нулу. Тиме се спречава прерано одбацивање грана у којима би комбинована корисност више нижих вредности могла да премаши праг минималне корисности, чак и ако ниједна појединачна вредност то не чини. Ово одржава независност граница, посебно када је једна од тих граница под утицајем негативних корисности (или нултих корисности). Приступ истражује сваку грану у којој би или једна будућа ставка или комбинована корисност преосталих ставки могла да задовољи праг. На тај начин се избегава прерано прекидање потенцијалних добитака од више ставки, док се и даље ефикасно одбацују гране које немају никакав потенцијал.

Временска сложеност алгоритма може се изразити као збир трошкова обраде појединачних скупа ставки. За сваки разматрани скуп ставки X, најзахтевнији део израчунавања односи се на његово проширивање кандидатима који задовољавају услове претраге. При томе, за сваког кандидата врши се провера његовог заједничког појављивања са скупом ставки X у релевантним секвенцама. Због тога је трошак обраде једног скупа ставки пропорционалан производу броја кандидата за проширење и броја секвенци у којима се посматрани скуп ставки копојављује. Укупна временска сложеност алгоритма добија се сабирањем ових трошкова за све скупове ставки који се заиста разматрају током извршавања алгоритма и може се формално записати као:

$$O = \sum_{X \in S} (C_x \; X \;\; T_x)$$

где је:

- S скуп свих скупова ставки које алгоритам посети у стаблу претраге
- C број кандидата за проширење скупа X
- T број секвенци у којима се X појављује.

Меморијска сложеност MAUList структуре одређује се збиром броја појављивања сваке ставке у трансакцијама базе података. Наиме, за сваку ставку i из скупа I конструише се MAUList, која садржи један запис по свакој трансакцији у којој се i налази. Сваки такав запис укључује поља као што су tid, utility, rmu и remu. Према томе, укупна меморија потребна за све MAUList-ове пропорционална је:

$$O = \sum_{i \in I} T_i$$

где је:

- I скуп ставки које задовољавају праг услова
- T број трансакција у којима се ставка i појављује.

У пракси, скуп I се значајно смањује пресецањем на основу горње границе auub, што доводи до оптималног меморијског трошка чак и за обимне базе података. Овај приступ осигурава да се меморија користи искључиво за складиштење неопходних података, омогућавајући ефикасно извршавање алгоритма.

У овом потпоглављу смо представили основне дефиниције, формално изложили проблем којим се бави овај рад и детаљно описали измене алгоритма неопходне за обраду мешовитих и

негативних вредности корисности уз потпуну тачност резултата. Претходна поглавља су испитала ограничења постојећих приступа за откривање скупова ставки високе корисности при управљању негативним вредностима. Предложени измењени алгоритам гарантује потпуну тачност при обради мешовитих и негативних вредности корисности, ефикасно решавајући све граничне случајеве који су представљали изазов за претходне методе. У наредном поглављу ћемо извршити евалуацију овог алгоритма на стварним скуповима података који садрже негативне и мешовите вредности корисности, како бисмо демонстрирали предности нашег приступа.

Фокус четвртог поглавља је представљање алгоритама PISRuleGrowth и MEHAUPM, који чине језгро предложеног истраживачког оквира. Оба алгоритма су формално описана псеудокодовима, уз детаљна објашњења кључних корака, структура података и стратегија проширења и одбацивања. PISRuleGrowth представља унапређени унутар-секвенцијални алгоритам који интегрише елементе RuleGrowth приступа са иновативним механизмима за идентификацију периодичних правила заједничких за више секвенци. За разлику од традиционалних метода, овај приступ осигурава потпуну тачност анализом свих појављивања правила у оквиру сваке посматране секвенце. Са друге стране, MEHAUPM представља модификацију EHAUPM алгоритма за прецизно руковање мешовитим вредностима корисности кроз осам кључних измена. Оне омогућавају тачно израчунавање корисности, чак и када су присутне негативне вредности, уз задржавање ефикасности кроз побољшане технике пресецања грана и одбацивања скупова ставки. Ова два алгоритма, иако су намењена различитим проблемима, демонстрирају заједничке принципе: употребу специјализованих структура података за брз приступ, агресивно пресецање за смањење простора претраге и баланс између тачности и скалабилности, чиме се осигурава њихова применљивост на реалне, обимне скупове података.

5. Експериментална валидација и анализа резултата

У наставку рада извршена је експериментална анализа предложених алгоритама, чији су концепти и имплементациони детаљи представљени у претходном поглављу. Експерименти су спроведени на реалним скуповима података како би се практично оцениле перформансе, скалабилност и ефикасност алгоритама PISRuleGrowth и MEHAUPM у условима који одговарају реалним сценаријима употребе. Анализа је организована у два одвојена дела: први део бави се евалуацијом алгоритма за откривање периодичних унутар-секвенцијалних правила (PISRuleGrowth), док други део тестира модификовани алгоритам за откривање скупова ставки високо-просечне корисности у присуству мешовитих вредности (MEHAUPM). Циљ експеримената није само потврда исправности и изводљивости предложених решења, већ и систематско испитивање утицаја различитих параметара на резултате, као и поређење са релевантним постојећим методама, тамо где је то било могуће. Сви експерименти су изведени у контролисаним условима, уз детаљно бележење времена извршавања, заузећа меморије и карактеристика добијених образаца, што омогућава квалитативно и квантитативно разумевање доприноса предложених алгоритама. Експериментална анализа у овом поглављу истовремено представља основу за проверу постављених хипотеза. У оквиру потпоглавља 5.1 разматра се потврђивање хипотеза H0.1 и H0.2, док се у потпоглављу 5.2 проверавају хипотезе H0.3 и H0.4.

5.1. Експериментална евалуација PISRuleGrowth алгоритма

Спроведен је низ експеримената с циљем процене перформанси предложеног алгоритма PISRuleGrowth. Алгоритам је имплементиран у програмском језику Java, а мерења потрошње

времена и меморије извршена су коришћењем стандардног Java API-а. Сви експерименти су реализовани на рачунару опремљеном AMD Ryzen 3 процесором радног такта 2.6 GHz са 8 GB RAM меморије, под оперативним системом Windows 10. Време извршавања је мерено у секундама (s), а употреба меморије у мегабајтима (MB). Приказане вредности представљају просек пет извршавања алгоритма за унапред дефинисане вредности параметара. Експерименти приказани у овом одељку служе за проверу хипотезе H0.1, која се односи на могућност откривања периодичних унутар-секвенцијалних правила, као и хипотезе H0.2, која се односи на ефикасност алгоритамске реализације тог приступа.

Карактеристике три базе података са реалним подацима коришћене у овим експериментима, детаљно су приказане у Табели II.

Табела II

Карактеристике база података

	MSNBC	WeeklySales	OnlineRetail
# Број секвенци	31 790	811	1496
# Број различитих ставки	17	68	3570
# Просечна вредност (различитих) бројева ставки по секвенци	5.33	10.96	115.65
# Просечна вредност броја ставки по трансакцији (величина трансакције)	1	1	21.35
# Просечна вредност броја трансакција по секвенци (величина секвенце)	13	52	9
# Тип базе података	Dense	Dense	Dense
# Врста података	click-stream from web site	sales data	sales data

База података MSNBC преузета је са веб-сајта SPMF библиотеке (SPMF, n.d.) и претходно је коришћена за откривање секвенцијалних правила. Свака секвенца садржи информације о активностима различитих корисника на информативном веб-сајту током једног дана. Свака трансакција представља кориснички захтев за једном страницом, при чему су странице означене бројевима уместо URL адресама. Ова база увек садржи само једну ставку по трансакцији. Међутим, просечан број ставки по секвенци и укупан број трансакција се разликују, јер потоњи број обухвата понављања истих ставки у различитим трансакцијама. У односу на остале базе података, ова поседује далеко највећи број секвенци.

WeeklySales дели карактеристику једне ставке по трансакцији са претходном базом, али има највећи укупан број трансакција међу свим коришћеним базама података. Добијена је са UCI веб-сајта (UCI Machine Learning Repository, n.d.) и садржи информације о недељној продаји за више од 800 производа. Свака секвенца се састоји од трансакција које представљају количине недељне продаје истог производа. Ова база података је за потребе истраживања трансформисана у SPMF формат.

OnlineRetail је такође преузета из UCI репозиторијума (UCI Machine Learning Repository, n.d.), али је модификована како би одговарала сврси истраживања. Оригинални скуп података је садржао секвенце са само једном куповином по купцу. Процесом агрегације, све трансакције истог купца су хронолошки поређане у једну секвенцу. Такође, секвенце са малим бројем трансакција су уклоњене, јер теоријски не могу да садрже два појављивања било ког правила.

Модификована база садржи убедљиво највећи број различитих ставки, као и највећи број различитих ставки по секвенци и по трансакцији. Након ових измена, тип базе се променио из ретке (sparse) у густу (dense) базу података.

Главни мотив за избор ових база података лежи у чињеници да су оне стандардни скупови података који се користе за бројне задатке у оквиру откривања образаца у подацима, укључујући и секвенцијалну анализу. Други разлог је тај што свака од њих представља различит тип базе података са којима се често сусрећемо у реалним ситуацијама. MSNBC је типичан представник база са изузетно великим бројем секвенци, WeeklySales се истиче великим бројем трансакција, док OnlineRetail садржи веома велики број ставки унутар сваке трансакције.

Неке од наведених карактеристика ових база података изразито одступају у односу на остале њихове особине, као и у поређењу са другим коришћеним базама. На пример, база MSNBC има изузетно велики број секвенци у односу на број различитих ставки, али и у односу на остале базе података. Посматрањем извршавања алгоритма над базама које имају овакве екстремне карактеристике стиче се посредан увид у његово понашање и на базама са сличним или умеренијим карактеристикама. Додатни мотив за избор ових скупова података јесте и приказ практичне примене алгоритма, посебно у анализи потрошачких корпи, кроз његове перформансе на последње две базе података.

Почетне вредности параметара у експериментима изабране су тако да представљају најрестриктивније прагове који ипак омогућавају добијање троцифреног броја периодичних унутар-секвенцијалних правила на све три базе података. Ознака PISRuleGrowth(a, b, c, d) користи се за означавање подешавања алгоритма, при чему слова редом представљају вредности минималне подршке, минималне поузданости, максималне стандардне девијације и минималног односа периодичних секвенцијалних правила. Као основна поставка коришћена је конфигурација PISRuleGrowth(0.05, 0.2, 5, 0.005).

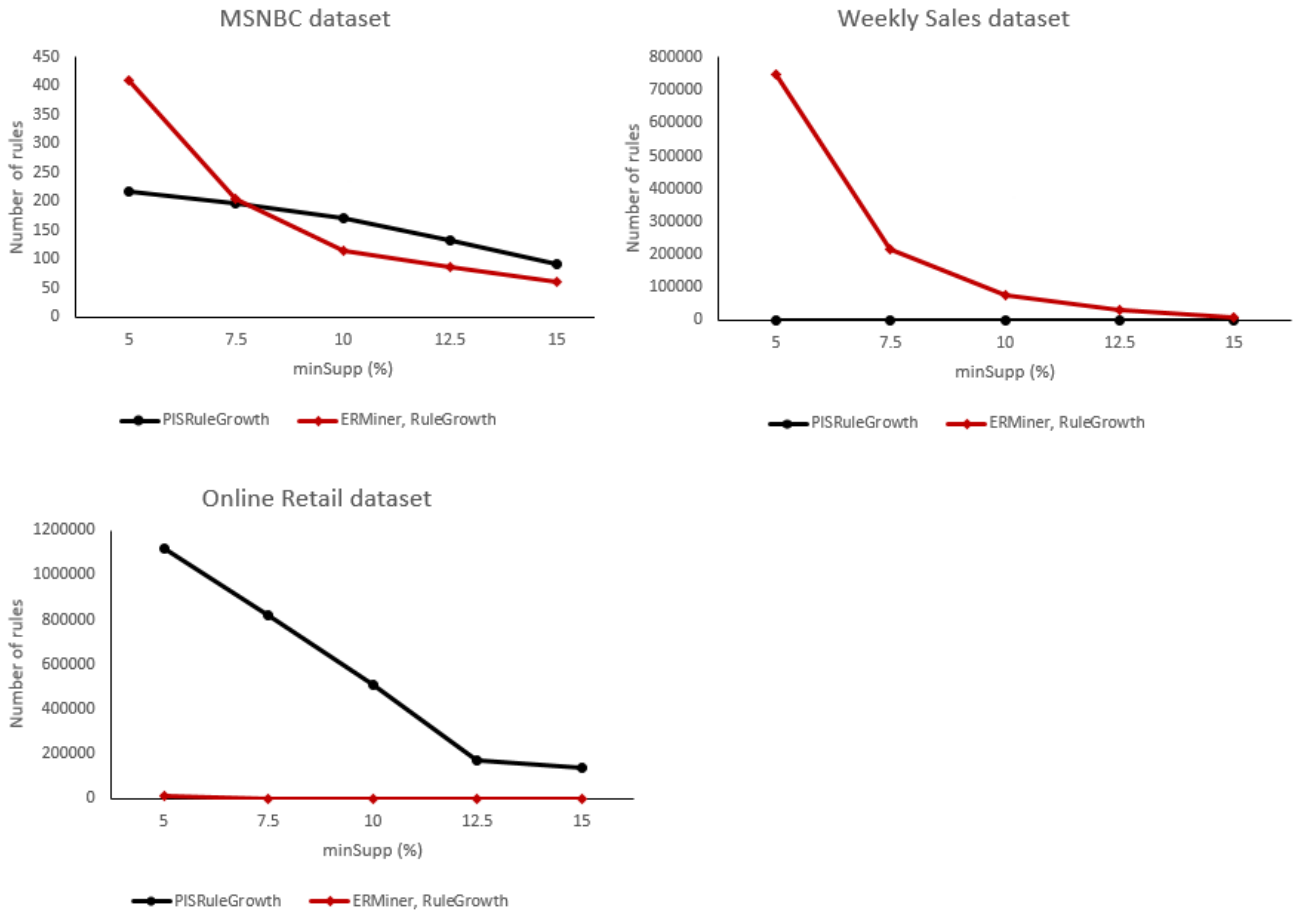
Изабрани прагови дефинишу минималне услове које једно правило мора да испуни како би се сматрало релевантним и стабилним у посматраним подацима. То значи да правило мора да се појави у најмање 5% трансакција у оквиру секвенце. Десна страна правила мора да се јави након појављивања леве стране у најмање 20% свих појављивања леве стране у датој секвенци. Поред тога, правило мора бити присутно у најмање 0,5% укупног броја секвенци у бази података. Вредност стандардне девијације за сваку секвенцу мора бити мања од 5, при чему је ово једини параметар који се задаје као апсолутна вредност, а не као процентуални праг који се накнадно преводи у релативну меру, као што је случај са осталим параметрима.

5.1.1. Поређење перформанси алгоритма PISRuleGrowth са другим алгоритмима за генерисање секвенцијалних правила при истим вредностима параметара

Први задатак је био да се перформансе нашег алгоритма упореде са перформансама других алгоритама за генерисање секвенцијалних правила. Наш алгоритам се значајно разликује од осталих алгоритама из исте групе јер многе његове карактеристике (нпр. проналажење унутар-секвенцијалних уместо међу-секвенцијалних правила) чине овај задатак веома захтевним. Ипак, изабрали смо опште алгоритме за генерисање секвенцијалних правила који су најсличнији нашем, ERMiner (Fournier-Viger et al., 2014a) и RuleGrowth (Fournier-Viger et al., 2011).

Експерименти су изведени како би се упоредио утицај промене параметра подршке у претходно поменутим алгоритмима и PISRuleGrowth алгоритму. Резултати приказани на Слици 1 указују на значајан диспаритет у броју генерисаних правила између постојећих алгоритама и нашег

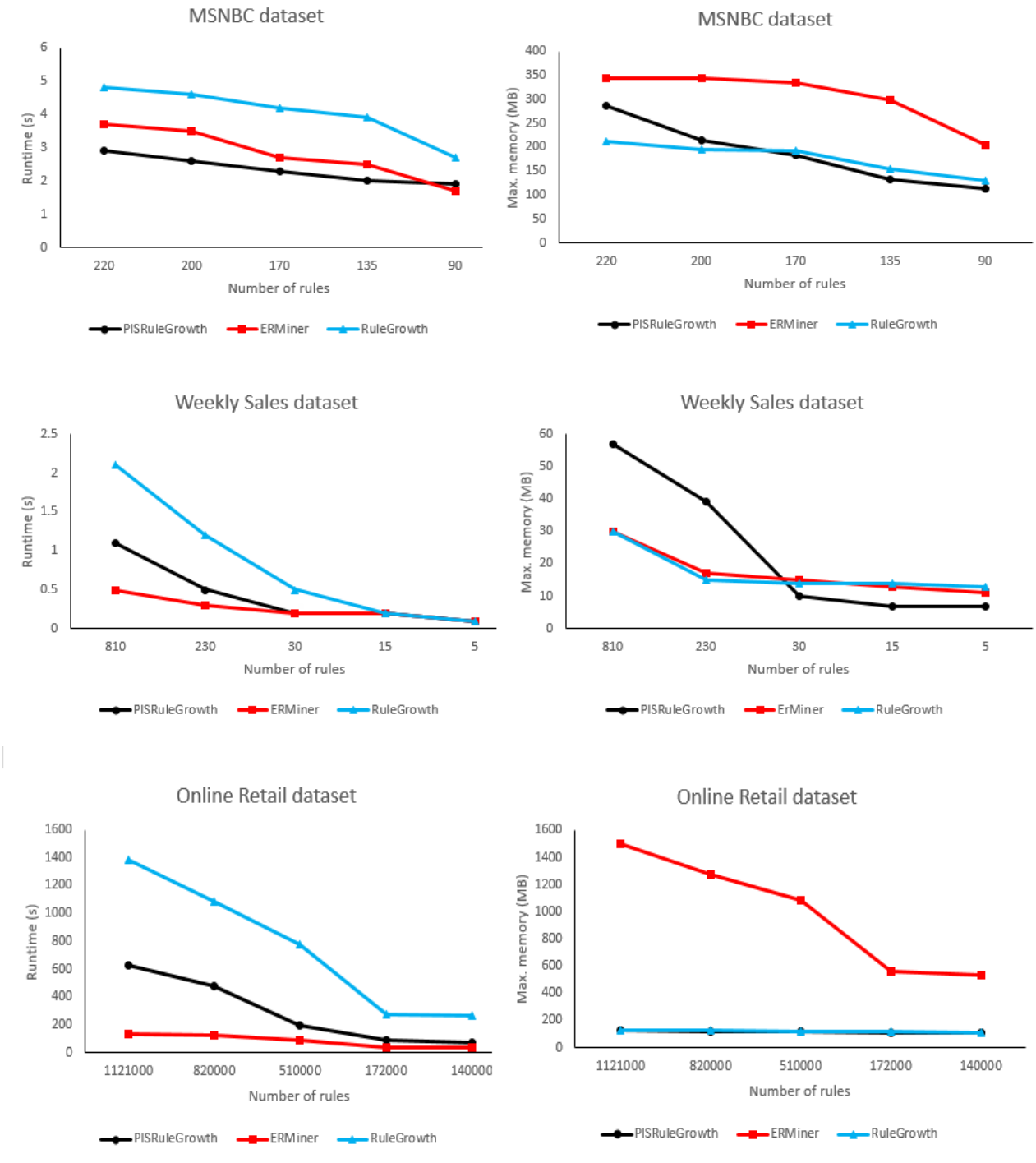
предложеног решења. Разлика између нашег алгоритма и остала два алгоритма кретала се од 2400 пута мањег до 140.000 пута већег броја генерисаних правила. Упоређивање њиховог времена извршавања и потрошње меморије у овим условима било би, у најмању руку, неинформативно. Експерименти са променом параметра поузданости имали су исти проблем.



Слика 1. Број генерисаних секвенцијалних правила (ERMiner/RuleGrowth) у односу на периодична унутар-секвенцијална правила (PISRuleGrowth) за MSNBC, WeeklySales и OnlineRetail базе података.

5.1.2. Поређење перформанси PISRuleGrowth и других алгоритама за генерисање секвенцијалних правила при генерисању истог броја правила

Узимајући у обзир претходне резултате, одлучили смо да упоредимо перформансе ових алгоритама без обзира на вредности њихових параметара. Касније у овом поглављу извршићемо исте анализе користећи неоптимизовану верзију нашег алгоритма. Стога су експерименти који упоређују ове популарне алгоритме за генерисање секвенцијалних правила са нашим алгоритмом изведени користећи вредности за које они генеришу исти или приближно исти број правила. Изабрана вредност броја правила добијена је варирањем прага подршке у оквиру PISRuleGrowth алгоритма. Резултати ових експеримената приказани су на Слици 2.



Слика 2. Поређење перформанси алгоритама PISRuleGrowth, ERMiner и RuleGrowth на базама података MSNBC, WeeklySales и OnlineRetail.

Очекивања пре извођења ових експеримената била су да ће алгоритам PISRuleGrowth имати најслабије перформансе у поређењу са преостала два алгоритма, због времена и меморије потребних за проналажење сваког појављивања сваког правила у свакој секвенци. Насупрот томе, друга два алгоритма морала су да пронађу само прво појављивање правила у свакој секвенци. Иако је ова претпоставка била разумна, у спроведеним експериментима она се није у потпуности потврдила.

Најслабије перформансе алгоритма PISRuleGrowth забележене су на скупу података WeeklySales, који има највећи број трансакција по секвенци. Разлог за то лежи у чињеници да

алгоритам тражи правила унутар сваке појединачне секвенце и обрађује сваку трансакцију. Алгоритам ERMiner је био бржи и до два пута када је број правила већи од 50, док су испод тог прага имали слично време извршавања. Такође, ERMiner је у тим условима био ефикаснији у погледу потрошње меморије и до 4,7 пута, док је испод тог прага PISRuleGrowth трошио од 1,5 до 1,9 пута мање меморије.

Алгоритам RuleGrowth је захтевао мање меморије, од 1,9 до 2,6 пута, када је број правила био већи од 190, док је за мање од 20 правила наш алгоритам био ефикаснији, од 1,9 до 2 пута. Са друге стране, PISRuleGrowth је био бржи од RuleGrowth алгоритма од 1,9 до 2,5 пута када је број правила већи од 20, док су испод тог прага имали приближно исто време извршавања.

Укупно посматрано, резултати на овом скупу података показују да је ERMiner остварио боље перформансе од PISRuleGrowth алгоритма, док је PISRuleGrowth у целини био успешнији од алгоритма RuleGrowth. Ипак, у случајевима када је број правила мали, наш алгоритам је постигао исте или чак боље перформансе у односу на ERMiner.

На скупу података OnlineRetail, алгоритам ERMiner је био бржи од PISRuleGrowth-а од 2 до 4,7 пута, али је при томе имао знатно већу потрошњу меморије, од 4,9 до чак 12 пута. Ово се може посматрати као одређени компромис: ERMiner постиже боље време извршавања, али по цену значајно веће потрошње меморије у односу на PISRuleGrowth.

С друге стране, наш алгоритам је био бржи од RuleGrowth-а од 2,2 до 3,9 пута, уз приближно исту потрошњу меморије, чиме је на овом скупу података јасно надмашио конкурентски алгоритам.

Најбоље перформансе је наш алгоритам постигао на скупу података MSNBC, који има највећи број секвенци. Алгоритам ERMiner је имао сличну брзину извршавања, али је PISRuleGrowth трошио око 2,3 пута мање меморије. Алгоритам RuleGrowth је имао приближно исту потрошњу меморије, али је био спорији од нашег алгоритма за 1,4 до 1,9 пута. Ови резултати указују на то да је PISRuleGrowth на овом скупу података надмашио оба конкурентска алгоритма.

Може се закључити да, у поређењу са другим алгоритмима за генерисање општих секвенцијалних правила, наш алгоритам остварује слабије резултате на скуповима података са великим бројем трансакција по секвенци. Међутим, његове перформансе се побољшавају са смањењем броја генерисаних правила на истом типу података. Код осталих типова скупова података може се очекивати да наш алгоритам покаже одређени компромис (tradeoff) између брзине извршавања и потрошње меморије, или да оствари боље перформансе у односу на друге алгоритме из ове групе.

Фокус у наставку овог поглавља биће на процени утицаја сваког параметра на перформансе алгоритама, мењањем њихових вредности и посматрањем промена у брзини извршавања и максимално заузетој меморији. Закључци о скалабилности алгоритама такође ће бити изведени током ових експеримената. Додатно, тестираћемо перформансе алгоритама за исти број правила на различитим типовима база података.

5.1.3. Поређење перформанси PISRuleGrowth алгоритма и његове неоптимизоване верзије при варирању параметра minSup

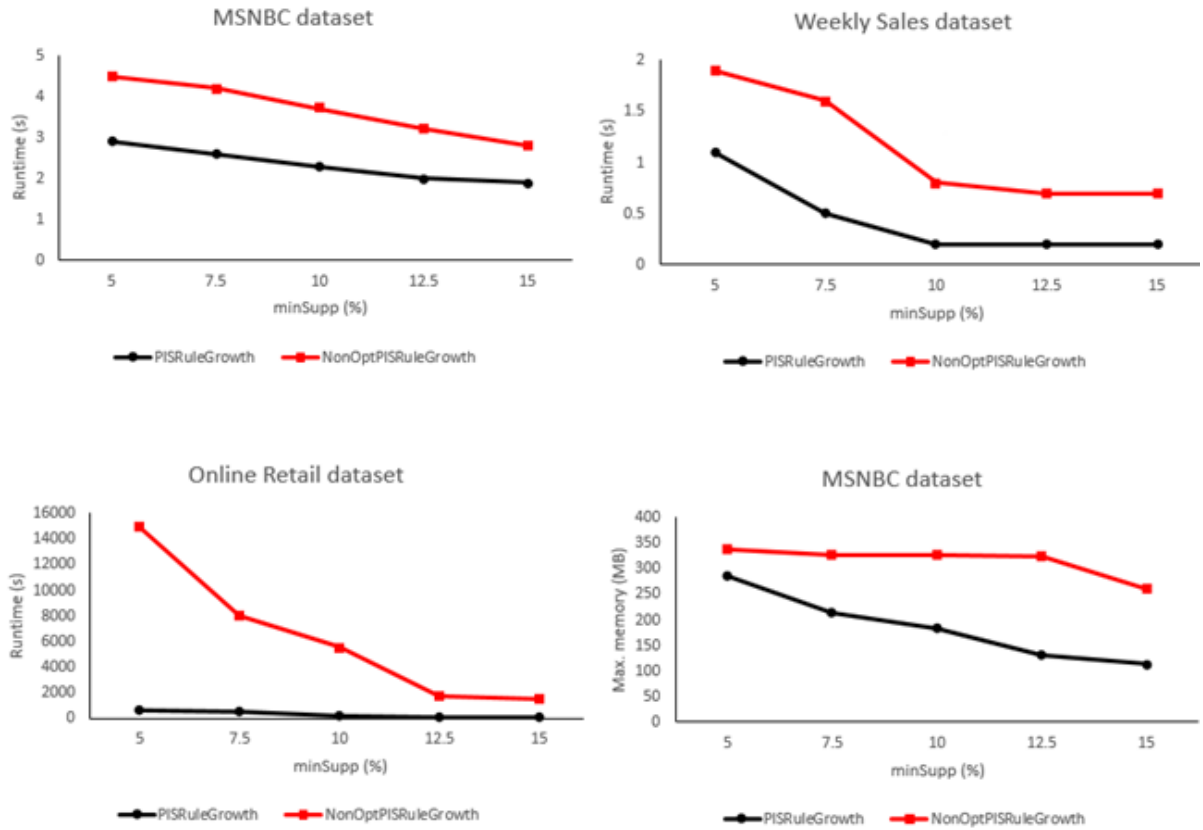
С обзиром на то да је ово први алгоритам за откривање периодичних унутар-секвенцијалних правила, његове перформансе при варирању вредности параметара не можемо поредити ни са једним постојећим алгоритмом. У ту сврху креирали смо NonOptPISRuleGrowth, неоптимизовану верзију нашег алгоритма.

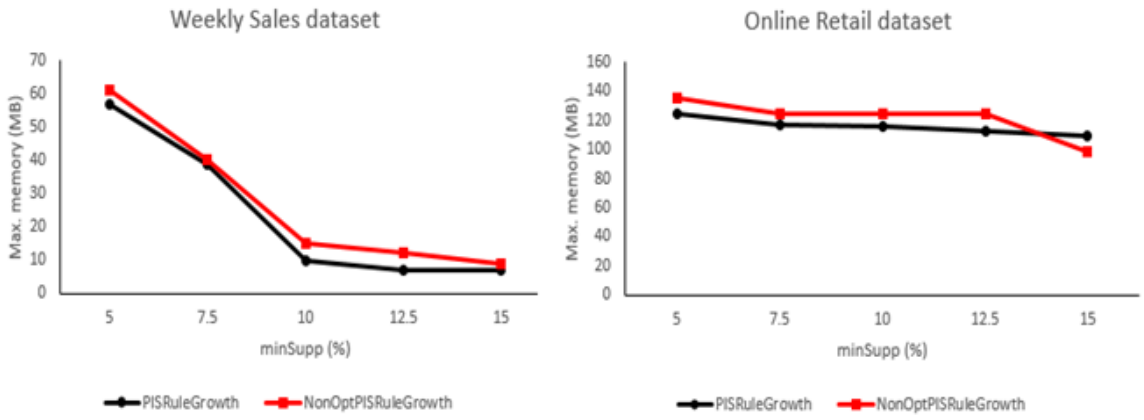
Овај алгоритам је сличнији алгоритму RuleGrowth: у оба методе проширивања пролазе кроз секвенце из базе података (укључујући израчунавање појављивања и периодичности), уместо коришћења вредности из генерисане мапе појављивања ставки. Сходно томе, његова мапа је мања јер не садржи вредности појављивања ставки у свакој секвенци.

Метод скенирања базе података је такође измењен и елиминише само ставке које се не појављују у довољном броју секвенци у бази, и то и из базе и из мапе, као код алгоритма RuleGrowth. У нашем оригиналном алгоритму те ставке се не уклањају из базе, већ се уопште не уносе у мапу појављивања ставки, заједно са ставкама које се не појављују довољан број пута унутар сваке секвенце. Такође, сваки услов који може убрзати елиминацију потенцијалног правила постављен је на крај, уместо да се примени што је раније могуће.

Најпре смо проценили утицај варирања минималног прага подршке на перформансе алгоритама и анализирали добијене резултате. Резултати су приказани на Слици 4. Експерименти су спроведени ради поређења времена извршавања два алгоритма при различитим вредностима параметра minSup (од 5% до 15%).

Са Слике 4 може се уочити да се време извршавања смањује са повећањем вредности minSup за све скупове података. Најмање смањење времена извршавања забележено је код MSNBC скупа података, што је у складу са најмањим смањењем броја генерисаних периодичних унутар-секвенцијалних правила, док је код осталих скупова података ово смањење израженије.





Слика 4. Утицај параметра minSup на базе података MSNBC, WeeklySales и OnlineRetail

Време извршавања се код оба алгоритма смањује приближно истом стопом, изузев код WeeklySales скупа података, али се њихове вредности значајно разликују. Код WeeklySales скупа наш алгоритам је смањио време извршавања за 19% више у односу на неоптимизовану верзију. Алгоритам PISRuleGrowth био је приближно 1,5 пута бржи од NonOptPISRuleGrowth за MSNBC, а 3 до 3,5 пута бржи за WeeklySales скуп података.

Највећа разлика у брзини забележена је за OnlineRetail скуп података, где је наш алгоритам био 16 до 28 пута бржи у односу на конкурентску верзију. Највећа разлика у брзини постигнута је управо на бази са највећим бројем периодичних унутар-секвенцијалних правила (1.120.854, у поређењу са 813 и 229 у осталим базама) за задате параметре, што указује на његову скалабилност.

Максимална потрошња меморије такође се смањивала са повећањем вредности minSup за све скупове података. Највеће смањење забележено је код WeeklySales скупа података, што је у складу са највећим смањењем броја генерисаних периодичних унутар-секвенцијалних правила. Количина максимално заузете меморије смањивала се код оба алгоритма приближно истом стопом, изузев код MSNBC скупа података, где је наш алгоритам остварио 38% веће смањење у односу на конкурентску верзију.

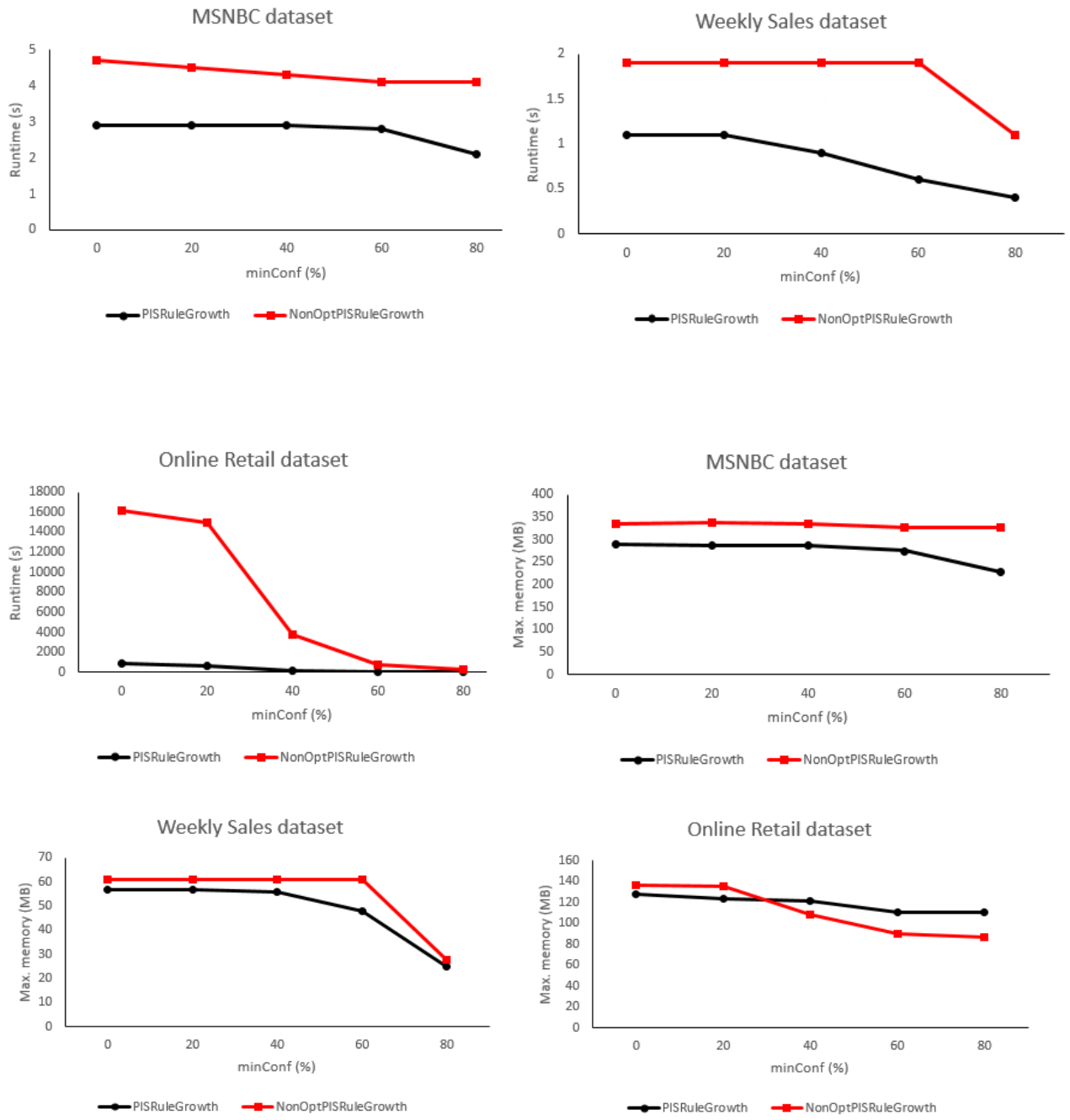
Разлика у максималној потрошњи меморије при варирању minSup између ова два алгоритма уочена је само код MSNBC скупа података, где је наш алгоритам трошио до 2,4 пута мање меморије. Наш алгоритам је показао разлику у потрошњи меморије управо на скупу података са најмањим смањењем броја генерисаних периодичних унутар-секвенцијалних правила што представља додатну предност у погледу његове скалабилности.

5.1.4. Поређење перформанси PISRuleGrowth алгоритма и његове неоптимизоване верзије при варирању параметра minConf

Проценили смо утицај варирања минималног прага поверења (minConf), а резултати су приказани на Слици 5. У овим експериментима, minConf је имао вредности од 0% до 80%. Време извршавања смањивало се са повећањем вредности minConf за све скупове података. Највеће смањење времена извршавања забележено је за OnlineRetail скуп података, а најмање, и то значајно, за MSNBC скуп података.

Опет, стопа смањења времена извршавања у нашем алгоритму за WeeklySales скуп података била је већа за 22%. Наш алгоритам је приближно 1,5 пута бржи од конкурентске верзије за

MSNBC и 1,7 до 3,2 пута бржи за WeeklySales скуп података. Најбољи резултат постигнут је за OnlineRetail скуп података, где је алгоритам био 14 до 26 пута бржи.



Слика 5. Утицај параметра minConf на скупове података MSNBC, WeeklySales и OnlineRetail

Највеће смањење максималне потрошње меморије при повећању minConf забележено је код WeeklySales скупа података, који је имао нешто мање смањење броја генерисаних периодичних унутар-секвенцијалних правила него OnlineRetail скуп у овом примеру. Ово се може објаснити чињеницом да OnlineRetail скуп података садржи далеко више различитих ставки које се чувају у мапи појављивања ставки у нашем алгоритму, као и да је у целини већи скуп података складиштен у меморији у неоптимизованој верзији, што објашњава разлику у потрошњи меморије без обзира на веће смањење броја генерисаних правила.

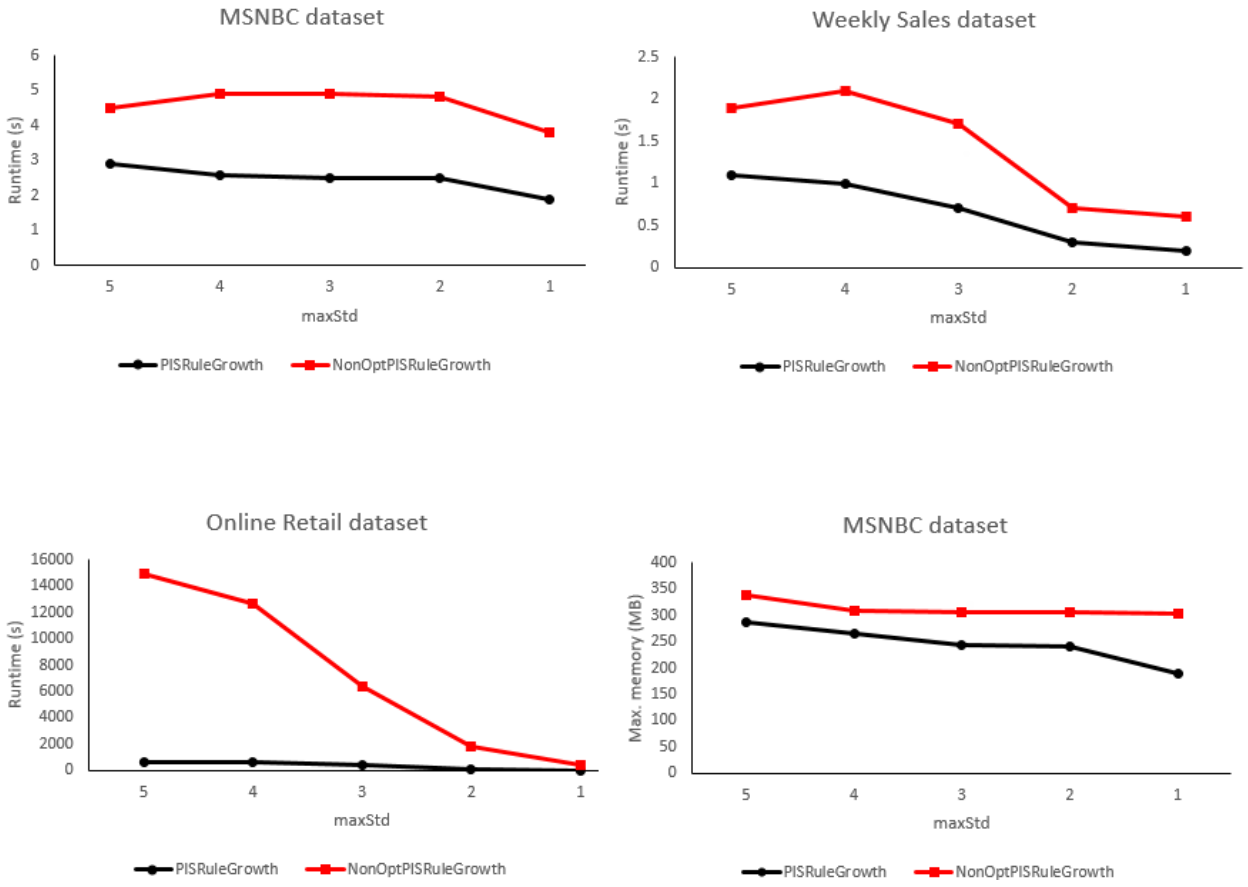
Највећа разлика у смањењу потрошње меморије забележена је за OnlineRetail скуп података, где је конкурентска верзија остварила 23% веће смањење. Количина максимално коришћене меморије је слична код оба алгоритма.

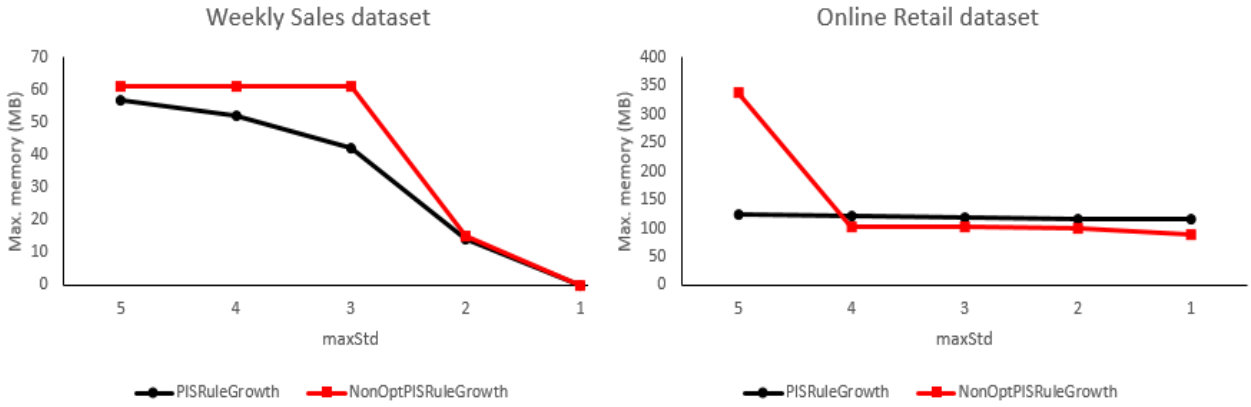
5.1.5. Поређење перформанси PISRuleGrowth алгоритма и његове неоптимизоване верзије при варирању параметра maxStd

Наредни експерименти су спроведени како би се испитао утицај максималне стандардне девијације (maxStd) на перформансе алгоритама. Резултати су приказани на слици 6. Параметар maxStd је имао вредности од 5 до 1.

Највеће смањење броја генерисаних периодичних унутар-секвенцијалних правила поново је забележено код базе података OnlineRetail, овог пута слично као код базе података WeeklySales, али је смањење времена извршавања било знатно мање. Ово се може објаснити већом величином базе података OnlineRetail, односно много већим бројем секвенци за обраду.

Време извршавања се смањило смањењем вредности maxStd. Смањење је било приближно исто код оба алгоритма, с тим што је наш алгоритам забележио нешто краће време извршавања за базе података MSNBC и WeeklySales (односно 13% и 11% брже од конкурента). Алгоритам PISRuleGrowth је 1,5 до 2 пута бржи за базу података MSNBC, 1,7 до 3 пута бржи за базу података WeeklySales, а 16 до 25 пута бржи за базу података OnlineRetail у односу на конкурентску верзију.





Слика 6. Утицај параметра maxStd на скупове података MSNBC, WeeklySales и OnlineRetail

Максимална потрошња меморије смањивала се са смањењем вредности maxStd у овом експерименту. Највеће смањење забележено је код базе података WeeklySales, која је имала слично смањење броја генерисаних периодичних унутар-секвенцијалних правила као база података OnlineRetail, али нешто мању стопу повећања брзине извршавања, коју надокнађује мањом потрошњом меморије.

Укупна количина максимално коришћене меморије је слична код оба алгоритма, осим код базе података MSNBC, где је наш алгоритам трошио до 1,6 пута мање меморије, и код базе података OnlineRetail, где је трошио нешто више. Понекад величина мапе појављивања ставки може бити већа од величине саме базе података изражене у MB, што може довести до веће потрошње меморије нашег алгоритма, посебно код мањих база података.

5.1.6. Поређење перформанси PISRuleGrowth алгоритма и његове неоптимизоване верзије при варирању параметра minRa

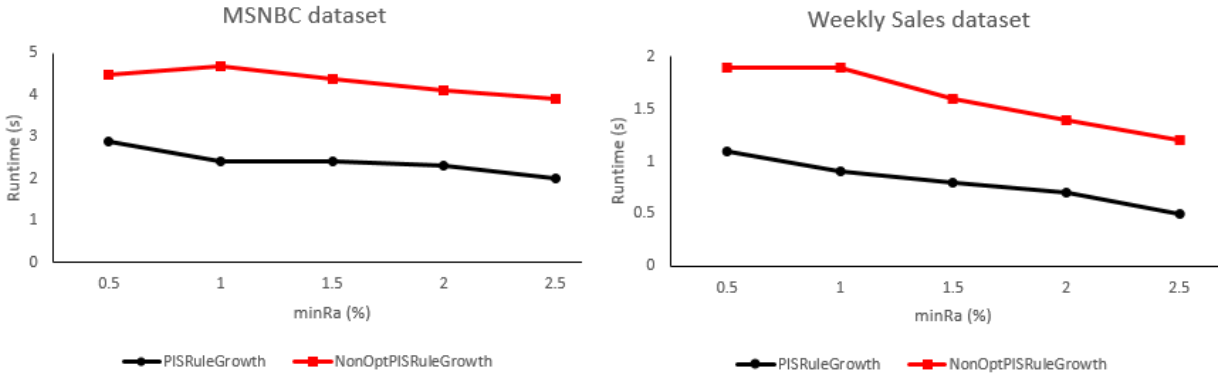
Испитан је утицај варирања минималног прага односа (minRa) на брзину извршавања и потрошњу меморије алгоритама, а резултати су приказани на сликама 7 и 8. Вредност параметра minRa кретала се од 0,5% до 2,5%.

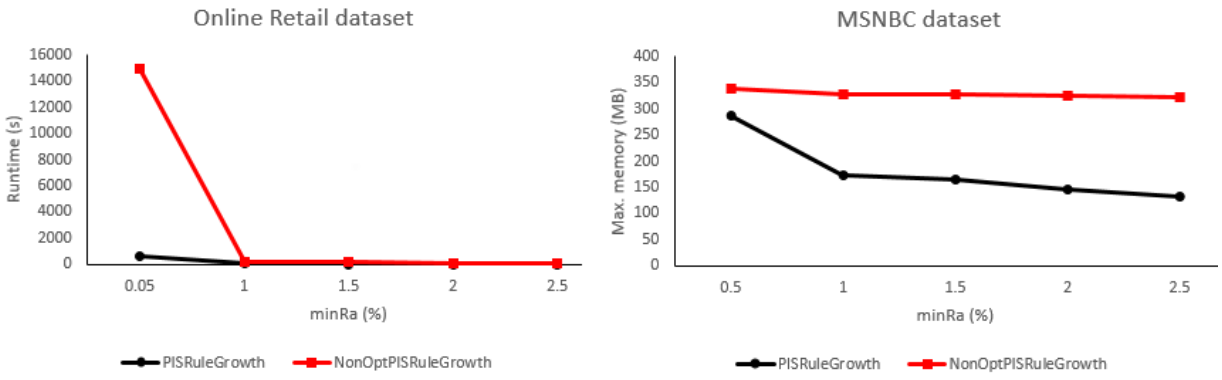
Слика 7 показује да се време извршавања смањивало са повећањем вредности minRa. Највеће смањење времена извршавања забележено је код базе података OnlineRetail, што је у складу са највећим смањењем броја генерисаних периодичних унутар-секвенцијалних правила. Време извршавања смањивано је код оба алгоритма приближно истом стопом, осим код базе података WeeklySales, где је наше решење остварило 18% веће смањење. Наш алгоритам је био од 1,5 до 1,9 пута бржи за базу података MSNBC и од 1 до 2,4 пута бржи за базу података WeeklySales у односу на конкурентску верзију. Најбољи резултат у поређењу постигнут је код базе података OnlineRetail, где је алгоритам био бржи од 7 до 32 пута.

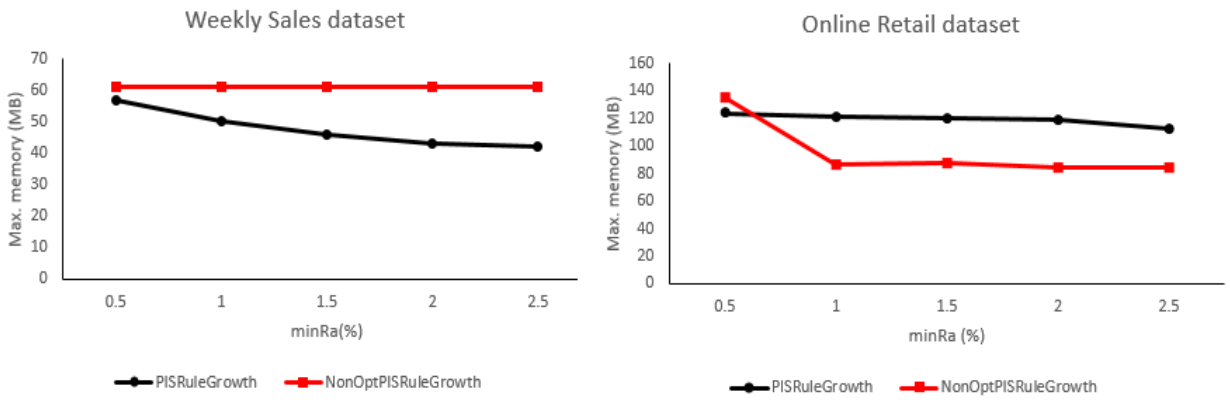
Максимална потрошња меморије смањивала се са повећањем вредности minRa. Највеће смањење забележено је код базе података MSNBC, која је имала најмање смањење времена извршавања. Укупна количина максимално коришћене меморије смањивала се код оба алгоритма приближно истом стопом, осим код базе података MSNBC, где је наш алгоритам остварио 52% веће смањење. При поређењу максималне потрошње меморије за ову базу података, наш алгоритам користио је до 2,4 пута мање меморије. За базу података WeeklySales, наш алгоритам трошио је нешто мање, а за OnlineRetail нешто више меморије.

Након овога можемо извући још неке закључке из претходно представљених резултата спроведених тестова. Најмање смањење броја генерисаних периодичних унутар-секвенцијалних правила услед промене вредности параметра подршке забележено је код базе података OnlineRetail. Ово је закључено поређењем са бројем правила у овој бази података при промени других параметара и узимањем у обзир већег процента смањења броја правила за овај параметар у осталим базама података. Разлог је највећи број ставки по секвенци ове базе података (више од 2000), од којих многе имају већу подршку од задатог прага параметра.

Ова база података је забележила највеће смањење броја генерисаних правила при повећању параметра minRa, јер елиминише ставке које се не појављују у дефинисаном броју секвенци. Ово доводи до израженијег смањења скупа кандидата у овој бази података, с обзиром на то да у просеку садржи око 115 различитих ставки по секвенци, од којих се многе не појављују у великом броју секвенци. Као последица тога, остале две базе података имале су најмање смањење броја генерисаних периодичних унутар-секвенцијалних правила за овај параметар услед мањег броја ставки.







Слика 7. Утицај параметра minRa на скупове података MSNBC, WeeklySales и OnlineRetail

Велико смањење броја генерисаних периодичних унутар-секвенцијалних правила у овој бази података при варирању параметра maxStd последица је већег броја трансакција по секвенци. Смањењем вредности овог параметра може доћи до занемаривања правила која имају већи размак између трансакција које садрже та правила.

Наш алгоритам је остварио највећу стопу смањења времена извршавања и највећу разлику у брзини (до 32 пута) у односу на конкурентски алгоритам за све тестове на бази података OnlineRetail. Његове перформансе су нарочито очигледне када се ради са великим бројем ставки и великим бројем ставки по трансакцији. Оне се обрађују веома брзо захваљујући структури мапе појављивања ставки и брзој елиминацији непотпуних периодичних унутар-секвенцијалних правила.

Највећа разлика у стопи смањења времена извршавања између два алгоритма забележена је код базе података WeeklySales за све извршене тестове. Ова база садржи највећи број трансакција по секвенци, што успорава конкурентски алгоритам јер мора да обради велике секвенце директно из базе података. Код ове базе података такође је забележена највећа стопа смањења коришћене меморије, осим при варирању параметра minRa, где је база података MSNBC имала бољу стопу смањења. Разлог је у томе што базе података WeeklySales и MSNBC имају најмањи, односно највећи број секвенци. Из тога следи да је повећање параметра minRa најкорисније за базе са већим бројем секвенци.

Наш алгоритам је имао нижу потрошњу меморије (до 2 пута) само код базе података MSNBC, која има најмањи број ставки и, сходно томе, најмању мапу појављивања ставки. Због тога је код неких вредности параметара за базу података OnlineRetail наш алгоритам трошио мало више меморије од конкурентске верзије, пошто она има највећи број ставки и највећу мапу појављивања ставки.

У закључку, наш алгоритам је имао најбољу брзину на великим базама података. Као што се и очекивало, велики број различитих ставки и трансакција по секвенци није имао значајан негативан утицај на перформансе алгоритма.

5.1.7. Поређење перформанси PISRuleGrowth алгоритма и његове неоптимизоване верзије при генерисању истог броја правила

Да бисмо утврдили утицај типа базе података на перформансе нашег алгоритма, спровели смо још један експеримент. Тестирали смо брзину и потрошњу меморије алгоритма за исти број правила у све три базе података. Да бисмо искључили утицај варирања појединачног параметра

на одређени тип базе података, спровели смо четири теста, при чему смо параметре повећавали један по један.

Параметар је у свакој бази података повећаван све док није дао жељени број генерисаних периодичних унутар-секвенцијалних правила, односно исти број као у осталим базама података. Ради сажетости, приказујемо само резиме добијених резултата.

Време извршавања алгоритма било је увек најкраће у бази података WeeklySales, а најдуже у бази података OnlineRetail. У бази података WeeklySales време извршавања било је 5 до 6 пута краће него у бази података MSNBC, где је алгоритам био од 1,5 до 2 пута бржи него на бази података OnlineRetail.

Што се тиче потрошње меморије, алгоритам је остварио најбољи резултат у бази података WeeklySales, где је трошио од 7 до 10 пута мање меморије у односу на други најбољи резултат. Базе података MSNBC и OnlineRetail смењивале су се на другом месту по потрошњи меморије, трошећи од 1,5 до 2 пута мање меморије у односу на ону трећу.

Имајући у виду ове резултате, можемо потврдити да највећи утицај на брзину извршавања алгоритма и потрошњу меморије има број секвенци, док број различитих ставки има мањи утицај. Међутим, брзина ће се смањити, а потрошња меморије повећати за вредности параметарских прагова које доводе до генерисања знатно већег броја периодичних унутар-секвенцијалних правила.

5.2. Експериментална евалуација MEHAUPM алгоритма

У овом одељку спроведена је серија експеримената ради процене перформанси предложеног MEHAUPM алгоритма (модификовани EHAUPM алгоритам). Алгоритам је имплементиран у програмском језику Java. Време извршавања и потрошња меморије мерени су коришћењем стандардног Java API-ја.

Сви експерименти су изведени на рачунару опремљеном процесором AMD Ryzen 3 такта 2,6 GHz и са 8 GB RAM меморије, под оперативним системом Windows 10. Време извршавања приказано је у секундама (s), а потрошња меморије у мегабајтима (MB). Наведене вредности представљају просек пет покретања алгоритма са фиксним вредностима параметара. Анализа представљена у овом одељку усмерена је на проверу хипотезе H0.3, која се односи на тачније откривање скупова ставки у присуству мешовите корисности, и хипотезе H0.4, која се односи на предности примене просечне корисности у погледу квалитета добијених образаца.

Карактеристике три реалне базе података коришћене у овим експериментима приказане су у Табели III.

База података DiscountSales преузета је са листе задатака такмичења Data Mining Cup 2016 (Data Mining Cup, n.d.). Реч је о реалној бази података са анонимизованим трансакцијама током двогодишњег периода, која садржи продају различитих модних производа из америчке интернет продавнице. Свака трансакција садржи поруџбину, производ, тренутну цену, препоручену малопродајну цену и друге детаље. Током посматраног периода продавница је у различитим временским интервалима одобравала попусте на бројне производе, што је довело до цена нижих од препоручених. Колона профит израчуната је као разлика између ових цена. Ова база података има убедљиво највећи број трансакција и највећу укупну корисност.

База података Superstore преузета је са платформе Kaggle (Alimo, 2023), али се може пронаћи и у репозиторијуму података платформе Tableau (Tableau, n.d.). Користили смо верзију која садржи историјске податке о трансакцијама у периоду од 2011. до 2014. године. Свака

трансакција садржи податке о фактури, производу, купцу, продаји и профиту. Профит може имати негативне вредности за поједине трансакције у којима су артикли купљени по сниженој цени. Ова база података има најмање вредности за све карактеристике наведене у Табели III, али има највећу густину података у односу на остале посматране базе.

База података RetailNegative садржи историјске податке о трансакцијама из белгијске малопродаје. Верзија коришћена у овом раду садржи вештачки додате негативне вредности корисности и првобитно је креирана за експерименте спроведене у раду у ком је представљен FHN алгоритам (Fournier-Viger, 2014). Све остале вредности, осим корисности, остале су непромењене. Ова база података има највећи број различитих ставки, као и највећу просечну и максималну дужину трансакције. Као што је приказано у Табели III, она такође показује највећу варијабилност у дужинама трансакција.

Табела III

Карактеристике база података

	DiscountSales	Superstore	RetailNegative
# Број трансакција	738,380	2,471	88,162
# Број различитих ставки	3,821	1,817	16,470
# Просечна вредност дужина трансакција	2.56	3.01	10.3
# Највећа дужина трансакције	35	14	76
# Укупна корисност (Total utility)	12,722,881	343,219	8,371,105
# Тип базе података	Sparse	Moderately Sparse	Sparse
# Врста података	Sales	Sales	Sales

Разлози за избор ових база података за експерименталну анализу су троструки. Прво, оне показују неопходност правилног руковања мешовитим и негативним вредностима корисности у реалним условима, где се профитабилност производа мења услед промотивних активности (нпр. попусти, ваучери) или отписа (нпр. истек рока трајања). Наглашавамо да фокусирање искључиво на негативне вредности корисности није довољно, јер цене производа у пракси ретко имају искључиво негативне вредности.

Друго, ове базе података су широко прихваћене у истраживачкој заједници, што омогућава директно поређење образаца откривених нашим алгоритмом са претходним и будућим резултатима других метода.

Треће, свака база података представља различите карактеристике реалних система: DiscountSales представља базе са великим бројем трансакција, RetailNegative сценарије са дугим трансакцијама и разноврсним асортиманом производа, док Superstore представља мале, густе базе са кратким трансакцијама.

У наредним пододељцима спроводимо серију експеримената како бисмо истакли предности откривања образаца са мешовитом и негативном високом просечном корисношћу (MEHAUPM).

Најпре анализирамо перформансе у погледу времена извршавања и потрошње меморије, где наш модификовани алгоритам показује супериорне резултате у односу на друге алгоритме високе корисности који подржавају негативне вредности.

Затим наглашавамо предности откривања образаца високо-просечне корисности у односу на класичне методе високе корисности, чак и када обе групе алгоритама подржавају негативне вредности корисности.

На крају, илуструјемо предности MEHAUPM приступа (који омогућава обраду мешовитих и негативних вредности корисности) у односу на алгоритме који занемарују такве вредности.

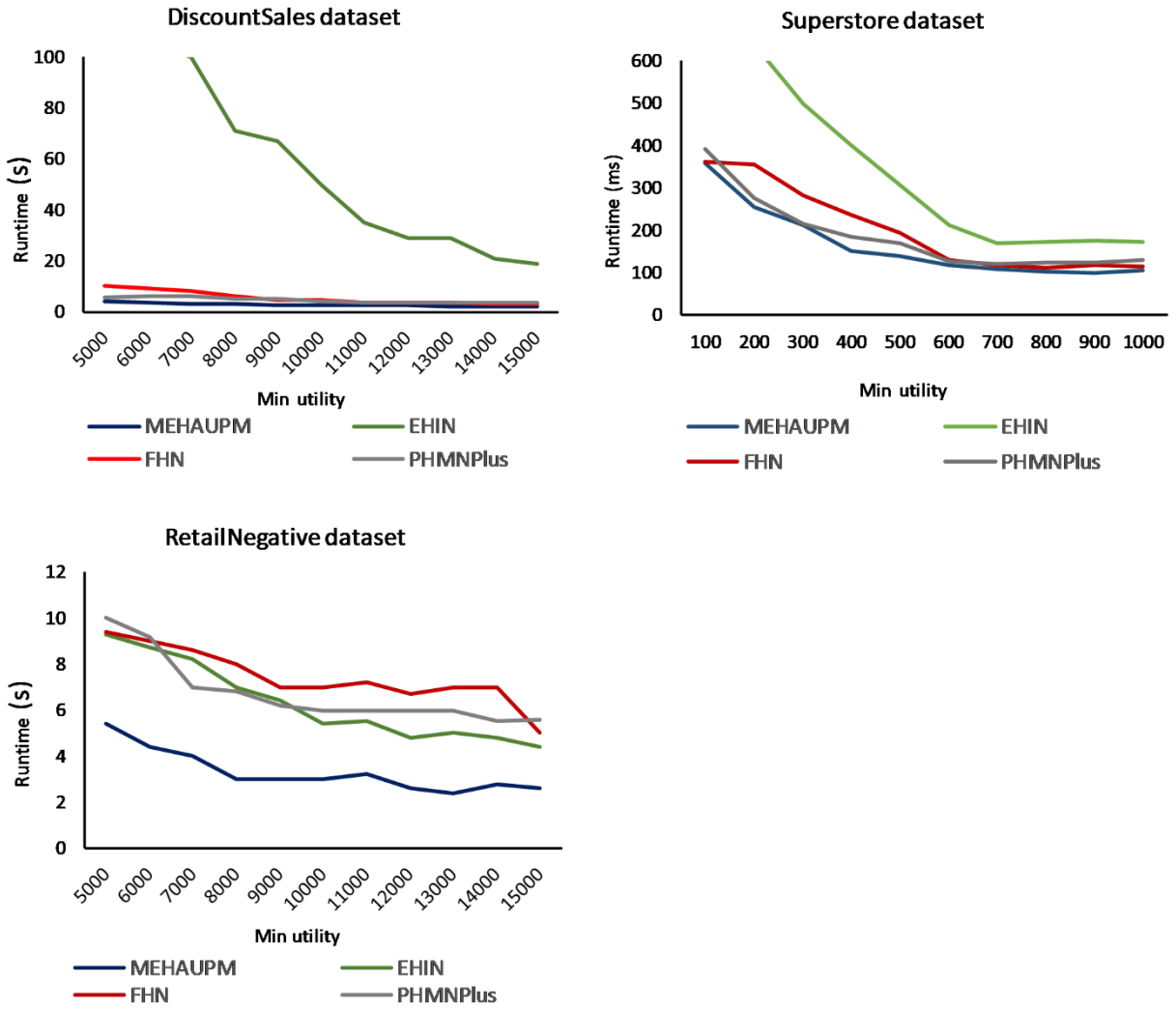
Минимални праг корисности за MEHAUPM изабран је појединачно за сваку базу података како би се на оптималан начин истакли значајни обрасци. Ради обезбеђивања фер поређења, параметар периодичности код PHMinerPlus алгоритма подешаван је само у случајевима када је број генерисаних образаца значајно одступао од резултата MEHAUPM алгоритма, чиме је спречено изражено заостајање почетног (базног) алгоритма.

5.2.1. Поређење перформанси алгоритма MEHAUPM и других алгоритама високе корисности који обрађују негативне вредности корисности при истим вредностима параметара корисности

Први задатак био је поређење перформанси нашег алгоритма са постојећим алгоритмима за откривање скупова ставки високе корисности који обрађују негативне вредности. Због ограниченог броја таквих алгоритама, одабране су три репрезентативне методе: FHN, референтни алгоритам за откривање скупова ставки високе корисности са негативним профитом; EHIN, новији меморијски ефикасан алгоритам; и PHMN, алгоритам за откривање скупова ставки високе корисности са ставкама које имају периодични карактер и могу имати негативне вредности.

Експерименти су спроведени ради упоређивања утицаја варирања прага минималне корисности на рад ових алгоритама и MEHAUPM. Вредности минималне корисности су одабране тако да истакну разлике између алгоритама, уз обезбеђивање најширег могућег опсега генерисаних скупова ставки. Времена извршавања и потрошња меморије приказани су на Сликама 8 и 9.

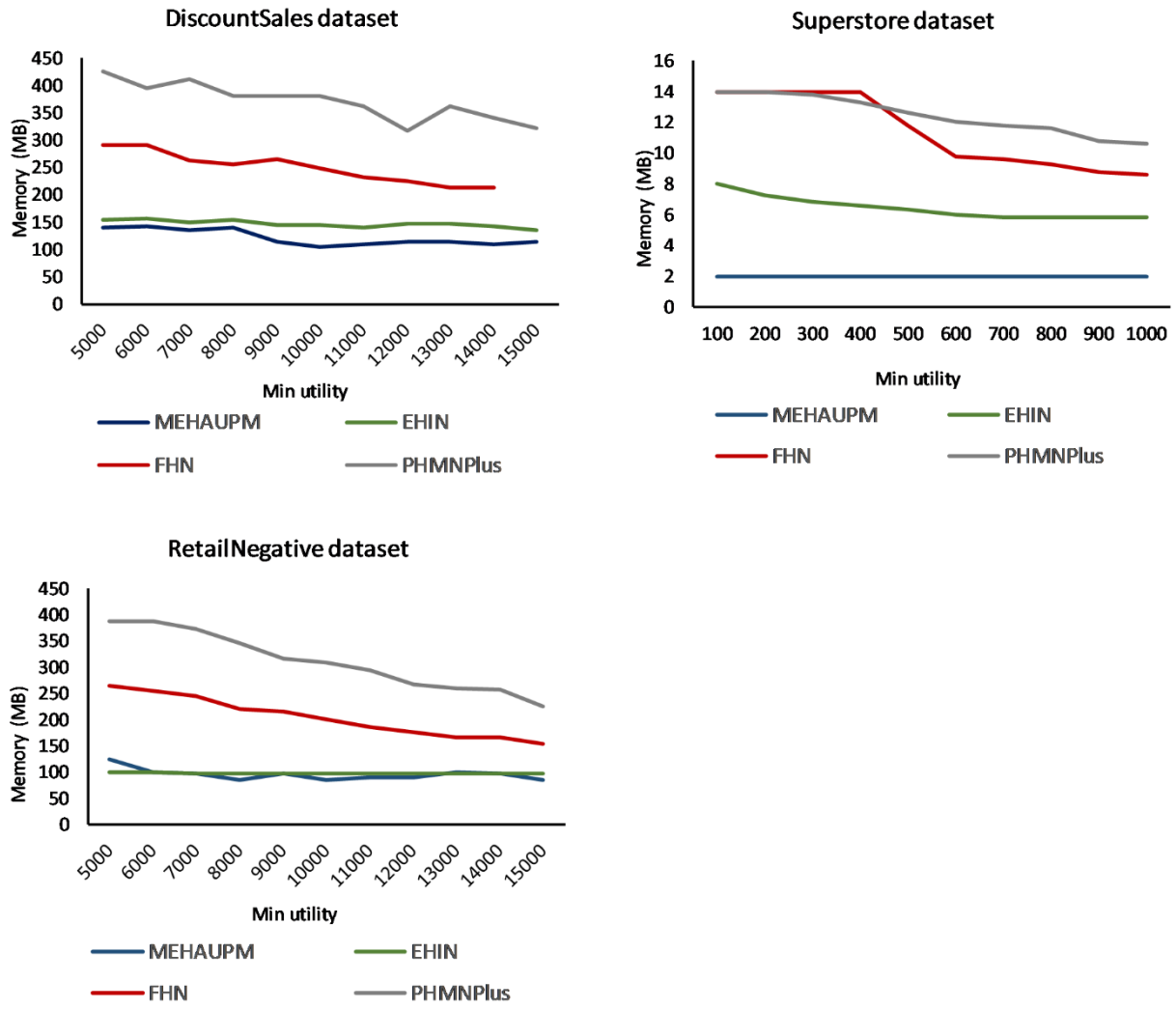
На бази података DiscountSales, наш алгоритам је постигао најкраће време извршавања за све прагове минималне корисности. Највећа разлика у брзини забележена је у поређењу са EHIN алгоритмом, који је био 9 пута спорији за највеће вредности корисности и до 32 пута спорији за ниже вредности корисности. Сходно томе, EHIN показује најгоре перформансе на базама података где прагови корисности генеришу највећи број скупова ставки.



Слика 8. Поређење перформанси (време извршавања) алгоритама MEHAUPM, EHIN, FHN и PHMNPlus на базама података DiscountSales, Superstore и RetailNegative.

FHN алгоритам је био 1,4 до 2,5 пута спорији од нашег, при чему му је брзина значајно опадала са повећањем броја скупова. PHMNPlus алгоритам је био 1,3 до 1,9 пута спорији, показујући најгоре перформансе при средњим праговима корисности. Док је време извршавања PHMNPlus алгоритма било најближе нашем за најниже вредности корисности, његова брзина није опадала истом стопом као код нашег алгоритма при повећавању прага корисности.

Ови резултати показују да наш алгоритам постиже супериорну брзину на базама са великим обимом трансакција, као што је DiscountSales.



Слика 9. Поређење перформанси (потрошња меморије) алгоритама MEHAUPM, EHIN, FHN и PHMNPlus на базама података DiscountSales, Superstore и RetailNegative.

На бази података Superstore забележена је најмања разлика у брзини између нашег алгоритма и осталих алгоритама. Ипак, наш алгоритам је постигао супериорну брзину за све прагове минималне корисности. EHIN алгоритам је био 1,6 до 2,6 пута спорији од нашег, при чему је највећа разлика забележена за средње вредности корисности.

FHN алгоритам је био до 1,5 пута спорији при средњим праговима корисности, али је показивао најближе перформансе нашем алгоритму (незнатно спорије) за ниже вредности корисности. PHMNPlus алгоритам је имао најупоредивије перформансе, с тим што је био незнатно спорији од нашег у скоро половини случајева, при чему је његова најлошија перформанса била 1,25 пута спорија.

Дакле, ови резултати показују да наш алгоритам надмашује постојеће методе на базама података са мањим бројем различитих ставки и мањим обимом трансакција.

Способност доследног постизања већих брзина, која се додатно унапређивала при већим стопама повећања брзине, најочигледније је показана на бази података RetailNegative. Сви базни алгоритми били су најмање 1,7 пута спорији од нашег алгоритма за све прагове минималне корисности.

EHIN алгоритам је био до 2,3 пута спорији од нашег при средњим праговима корисности, чинећи га најбржим међу базним алгоритмима. Најспорији алгоритам на бази RetailNegative био је FHN, са временом извршавања до 2,9 пута споријим од нашег, при чему су најлошији резултати забележени за веће вредности корисности. PHMNPlus алгоритам је био до 2,5 пута спорији, показујући највећу разлику у перформансама при вишим вредностима корисности.

Иако су сви базни алгоритми показали најмање разлике у брзини при нижим праговима корисности, њихова времена извршавања нису се повећавала тако ефикасно као код нашег алгоритма са растом вредности корисности. Док су FHN и PHMNPlus надмашили EHIN на две базе, само су EHIN и наш алгоритам показали супериорне перформансе на бази RetailNegative, која се карактерише дугим трансакцијама и великим бројем различитих скупова ставки.

На свим базама података, MEHAUPM је доследно показивао најкраћа времена извршавања, демонстрирајући јасну предност у перформансама у односу на остале алгоритме. Наши резултати показују да је брзина MEHAUPM остала супериорна без обзира на карактеристике базе података, укључујући величину, дужину скупова ставки и број трансакција.

Најмања разлика у перформансама забележена је на бази Superstore, која садржи најмањи број трансакција и највећу густину међу оцењеним базама. EHIN је показао најгоре перформансе на базама DiscountSales и Superstore, али је генерално надмашивао остале базне алгоритме на бази RetailNegative. Сходно томе, време извршавања EHIN алгоритма било је најбоље за базу са највећим бројем и дужином трансакција.

Након MEHAUPM-а, PHMNPlus је показивао следеће најбрже перформансе, постижући најбоље резултате посебно на најгушћој бази Superstore.

Са Слике 9 може се уочити да се потрошња меморије повећава са смањењем вредности минималног прага корисности, што је у складу са повећањем броја генерисаних образаца. Наш алгоритам је у већини случајева остварио најмању потрошњу меморије у односу на остале алгоритме. Највеће разлике у потрошњи меморије уочене су код базе података DiscountSales, где конкуренти показују значајно веће заузеће меморије, посебно при нижим вредностима прага. Код Superstore базе разлике су мање изражене, што се може објаснити мањим бројем трансакција и већом густином података. Најстабилније предности наш алгоритам показује код RetailNegative базе, где задржава мању потрошњу меморије кроз све вредности параметра, што указује на бољу скалабилност у условима већег броја различитих ставки и дужих трансакција.

5.2.2. Предности алгоритма MEHAUPM у односу на друге алгоритме високе корисности који обрађују негативне вредности корисности при истим вредностима параметара корисности

Циљ наредних пасуса је да се истакну предности скупова ставки које генеришу алгоритми способни да обрађују негативну корисност, али који користе различите приступе израчунавању корисности. Да бисмо нагласили предности приступа високо-просечне корисности у односу на класични приступ високе корисности у овом контексту, спровешћемо серију експеримената.

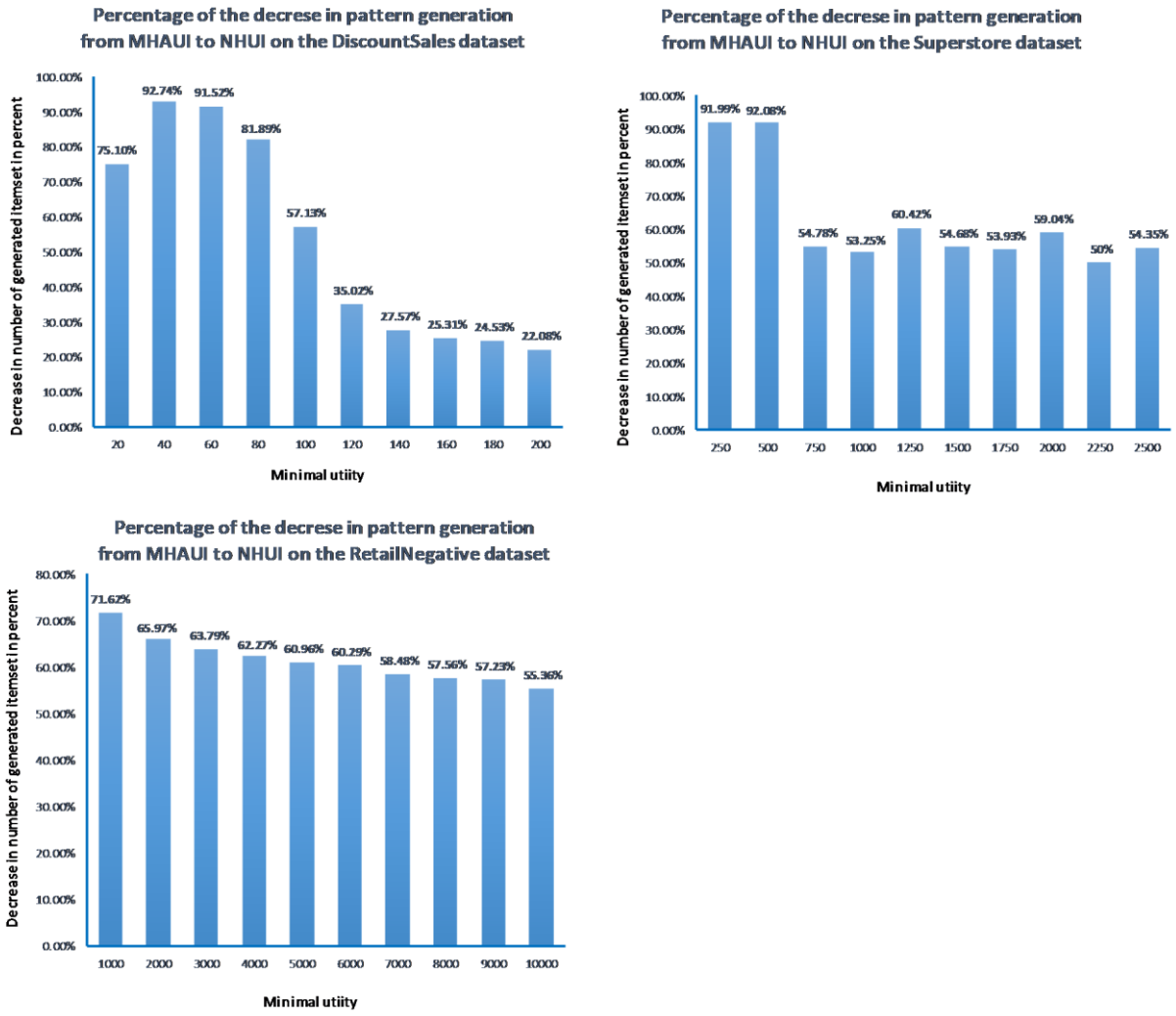
Циљ сваке методе откривања скупова ставки јесте идентификација скупова са високом употребном вредношћу и пожељним карактеристикама. Експерименти анализирају резултате ових метода, односно скупове ставки које оне генеришу. Конкретно, упоредићемо генерисане скупове ставки у погледу њиховог броја за задату минималну вредност корисности, просечне дужине и просечне корисности по ставци.

Алгоритам коришћен за генерисање скупова ставки високе корисности са подршком за негативне вредности јесте FHM. Ради сажетости, у појединим деловима текста изоставићемо

експлицитно навођење да ови алгоритми подржавају негативну корисност или да генерисани скупови могу садржати ставке са негативним вредностима корисности, међутим, то се подразумева у читавом овом пододељку.

Циљ сваке методе откривања скупова ставки, као и сваке друге методе откривања образаца у подацима, јесте да произведе мањи број скупова ставки који су информативнији. Вођени тим циљем, спровели смо експерименте како бисмо показали да откривање скупова ставки високо-просечне корисности генерише мањи број скупова у односу на откривање скупова високе корисности, када оба приступа обрађују ставке са негативном корисношћу. Наредни експерименти ће додатно показати да су преостали скупови ставки уједно и најзначајнији.

Прва разлика у броју генерисаних скупова ставки за базу података DiscountSales јавља се при минималној вредности корисности од 30.000. Скуп ставки који је изостављен методом високо-просечне корисности био је „articlei1003279 articlei1002178“, са укупном корисношћу од 33.800, али просечном корисношћу од само 16.900. С обзиром на овако висок минимални праг корисности, овај двочлани скуп ставки може се сматрати мање значајним у односу на једночлани скуп са истом укупном корисношћу.



Слика 10. Проценат смањења броја генерисаних образаца при примени мешовите високо-просечне корисности у односу на примену високе корисности, када оба приступа обрађују негативну профитну корисност, на базама података DiscountSales, Superstore и RetailNegative.

Анализа резултата приказаних на Слици 10. показује да за ову базу података откривање скупова високе корисности генерише до 92,74% више скупова ставки. Овај вишак се пре свега састоји од веома дугих и потенцијално мање вредних скупова ставки, као и бројних репетитивних образаца.

Иако се разлика у броју генерисаних скупова смањује са повећањем минималне вредности корисности и укупним смањењем броја скупова, најзначајнија разлика се јавља управо онда када је број генерисаних скупова већ велики, што представља значајну предност.

На пример, при минималној вредности корисности од 40, откривање скупова високо-просечне корисности генерише 85.294 скупа ставки, док метода високе корисности генерише 1.174.995 скупова ставки.

Код базе података Superstore, прва разлика у броју генерисаних скупова ставки јавља се при минималној вредности корисности од 15.000, при чему откривање скупова високо-просечне корисности генерише 50% мање скупова.

Изостављени двочлани скуп ставки био је „Canon imageCLASS 2200 Advanced Copier; Acco Perma 4000 Stacking Storage Drawers“, са укупном корисношћу од 15.130. Међутим, његова просечна корисност од само 7.565 указује на то да је мање значајан.

Процентуална разлика у броју генерисаних скупова ставки између откривања скупова високо-просечне корисности и високе корисности (при обради мешовитих и негативних вредности корисности) за базу података Superstore достиже и до 92,08%. Као и код базе DiscountSales, највећа разлика у броју генерисаних скупова јавља се при најнижим вредностима минималне корисности, што потврђује претходно описане предности.

На пример, при минималној вредности корисности од 250, откривање скупова високо-просечне корисности генерисало је 941 скуп ставки, у поређењу са 11.743 скупа добијеним методом високе корисности. Разлика достиже врхунац при минималној корисности од 500, а затим опада на 54,78%.

У поређењу са базом DiscountSales, ова разлика се даље не смањује, већ осцилира око те вредности за преостале вредности минималне корисности. Чак и при повећању минималне корисности до 10.000, разлика не пада испод 54,78%, што ову базу чини оном са највећом укупном разликом у броју генерисаних скупова ставки међу посматраним базама.

При откривању образаца у RetailNegative бази, прва разлика у броју генерисаних скупова ставки јавља се при минималној вредности корисности од 510.000, при чему откривање скупова високо-просечне корисности генерише 33% мање скупова.

Изостављени скуп ставки био је „38 39“, са укупном корисношћу од 510.635, али просечном корисношћу од 255.317. Пошто се обе појединачне ставке („38“ и „39“) већ појављују као учестали скупови ставки, овај пример јасно илуструје предност нашег приступа у изостављању само редундантних и понављајућих скупова ставки за задати праг корисности.

Анализа графика на Слици 10. показује да, за разлику од осталих база података, разлика у броју генерисаних скупова ставки не прелази 90%, већ достиже максимум од 71,62%. Иако је ова разлика мања у односу на друге базе, она је и даље значајна.

Поред тога, смањење ове разлике са променом минималне вредности корисности је постепено и остаје изнад 47% у широком опсегу вредности минималне корисности (од 35.000 до 1.000). Ово указује на то да код базе података RetailNegative метода високе корисности производи значајан вишак скупова ставки за већину вредности минималне корисности које дају више од једног резултата.

Разлика у исходима између ова два приступа откривања образаца је најконзистентнија управо код базе података RetailNegative кроз различите нивое минималне корисности.

Поред броја генерисаних скупова ставки, у откривању скупова ставки постоји тенденција ка генерисању скупова који су краћи и садржајнији. Ова карактеристика смањује редундантност унутар скупова и олакшава анализу кориснику, јер се приказује мањи број ставки за разматрање.

Сходно томе, други експеримент имао је за циљ да прикаже разлику у просечној дужини генерисаних скупова ставки између приступа високо-просечне корисности и високе корисности при обради негативне корисности. Наши резултати, приказани на Слици 11., показују да откривање скупова високо-просечне корисности доследно генерише краће скупове ставки за све посматране вредности минималне корисности.

Код базе података DiscountSales, просечна дужина скупова ставки генерисаних методом високо-просечне корисности креће се од 1 до 2,59, док се просечна дужина скупова генерисаних методом високе корисности креће од 1,91 до 5. Дакле, откривање скупова високо-просечне корисности производи скупове до 3,8 пута краће од оних које генерише метода откривања скупова високе корисности.

Као што се могло очекивати на основу резултата са Слике 8., најзначајнија разлика у просечној дужини јавља се при нижим вредностима минималне корисности, које истовремено дају и највећи број скупова. База DiscountSales издваја се као једина у којој просечне дужине скупова добијених коришћењем два различита приступа постају практично исте на одређеним нивоима минималне корисности.

На овој бази, откривање скупова високо-просечне корисности дало је и најкраћу просечну дужину скупова од 1, као и најдужу просечну дужину од 2,59 у поређењу са резултатима на осталим базама података.

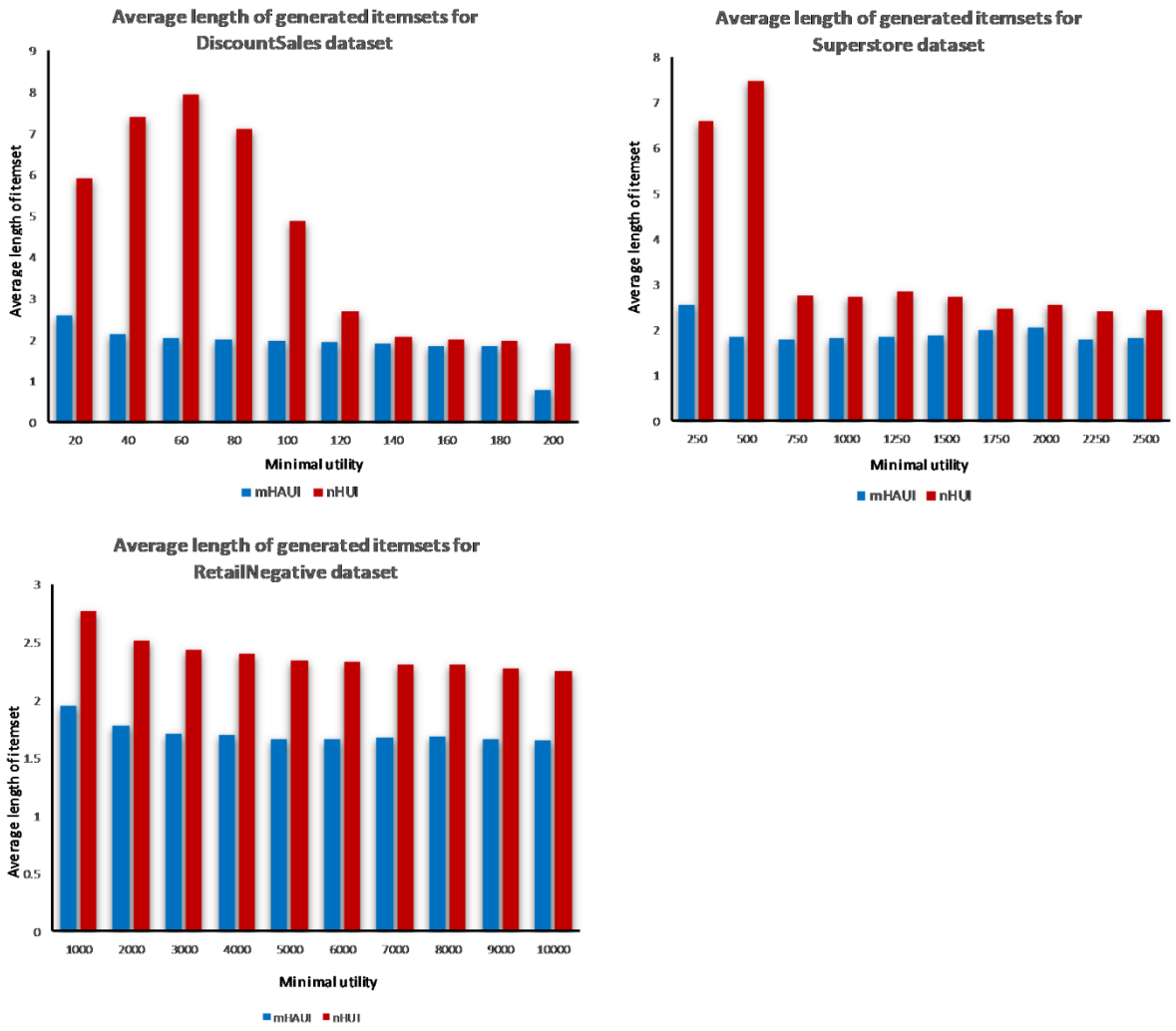
Просечна дужина генерисаних скупова ставки за базу података Superstore при примени откривања скупова ставки високо-просечне корисности креће се од 1,81 до 2,53, док се при откривању скупова ставки високе корисности креће од 2,39 до 7,45. Као и код базе DiscountSales, најзначајнија разлика у просечној дужини скупова јавља се при нижим вредностима минималне корисности.

За различите нивое минималне корисности, просечна дужина скупова високо-просечне корисности остаје релативно конзистентна, без значајних одступања. Скупови генерисани методом високо-просечне корисности имају просечну дужину која је 1,23 до 4,02 пута краћа у односу на скупове добијене методом високе корисности, када оба приступа обрађују негативне вредности профита по ставци.

Битно је истаћи да ова база података показује највећу разлику у просечној дужини скупова између приступа високо-просечне корисности и високе корисности у поређењу са осталим анализираним базама.

Код базе података RetailNegative, просечна дужина скупова ставки генерисаних методом високо-просечне корисности кретала се од 1,65 до 1,95. Просечна дужина скупова генерисаних методом високе корисности кретала се од 2,25 до 2,77.

Ова разлика у дужини је била стабилна за све тестиране вредности минималне корисности. Скупови високо-просечне корисности били су 1,36 до 1,42 пута краћи од оних добијених методом високе корисности. Минимална разлика није падала испод фактора 1,36, што уједно представља највећу минималну разлику у односу на остале посматране базе података.



Слика 11. Просечна дужина скупова ставки при примени откривања скупова ставки мешовите високо-просечне корисности у односу на откривање скупова ставки високе корисности, када оба приступа обрађују негативну корисност, на базама података DiscountSales, Superstore и RetailNegative.

Поред тога, база RetailNegative показује најнижу горњу границу дужине скупова високо-просечне корисности, која никада није премашила 1,96. Стога ова база има најшире распоређену разлику у дужини скупова за посматране типове скупова.

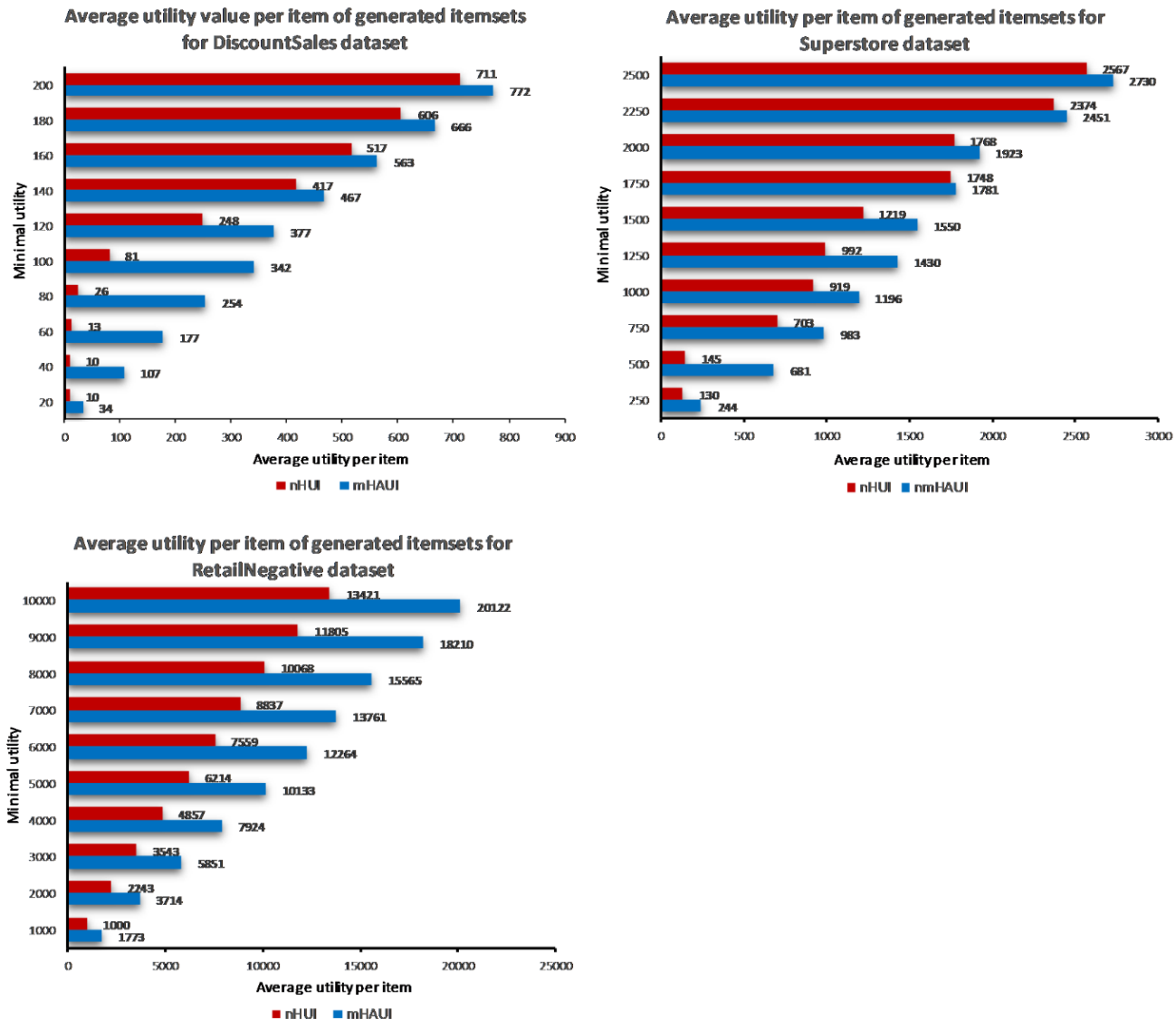
Један од главних циљева свих алгоритама је идентификација скупова ставки или образаца са што већом корисношћу. Претходни експерименти су показали да откривање скупова високо-просечне корисности генерише мањи број и краће скупове у односу на откривање скупова високе корисности.

Да бисмо показали да се смањен број и краћа дужина скупова добијених методом високо-просечне корисности поклапају са већим значајем, израчунали смо просечну корисност по ставци за откривене скупове. Резултати су приказани на Слици 12.

Код базе података DiscountSales, просечна корисност по ставци скупова генерисаних методом високо-просечне корисности кретала се од 34 до 772, док се код скупова генерисаних методом високе корисности кретала од 10 до 711.

Важно је истаћи да су скупови високо-просечне корисности имали просечну корисност по ставци и до 10,7 пута већу у односу на скупове добијене методом високе корисности, што представља највећу забележену разлику у просечној корисности између посматраних база.

Као што се могло очекивати, ова разлика је најизраженија при нижим вредностима минималне корисности, када оба алгоритма генеришу већи број скупова. На пример, при минималном прагу корисности од 60, 47.985 скупова генерисаних методом високо-просечне корисности имало је просечну корисност по ставци више од 10 пута већу у односу на 565.644 скупова генерисаних методом високе корисности.



Слика 12. Просечна корисност по ставци генерисаних скупова ставки при примени откривања скупова мешовите високо-просечне корисности у односу на откривање скупова високе корисности, када оба приступа обрађују негативну профитну корисност, на базама података DiscountSales, Superstore и RetailNegative.

Ова разлика у просечној корисности по ставци постајала је мање значајна са повећањем минималне вредности корисности.

Најмање изражене разлике у просечној корисности по ставци забележене су код базе података Superstore, где се просечна корисност по ставци креће од 244 до 2.730 за скупове генерисане методом високо-просечне корисности, а од 130 до 2.567 за скупове добијене методом високе корисности.

Ипак, скупови високо-просечне корисности и даље су имали просечну корисност по ставци и до 4,7 пута већу у односу на скупове високе корисности. Побољшање резултата било је нарочито значајно за нижу половину посматраних вредности минималне корисности.

Међутим, како је праг минималне корисности повећаван, посебно од вредности 1.750 и више, разлика се смањивала. Упркос овом смањењу при вишим праговима, можемо закључити да скупови високо-просечне корисности генерално поседују већу просечну корисност по ставци за ову базу података.

Код базе података RetailNegative, скупови генерисани методом високо-просечне корисности имали су просечне вредности корисности између 1.773 и 20.122, док се код скупова високе корисности оне кретале од 1.000 до 13.421.

На овој бази забележена је највећа просечна корисност по ставци за наш алгоритам у односу на све анализиране базе података. Просечна корисност по ставци била је 1,5 до 1,77 пута већа за скупове високо-просечне корисности у односу на скупове високе корисности.

Резултати на овој бази показали су најзначајнију и најстабилнију релативну предност скупова високо-просечне корисности, при чему њихова просечна корисност по ставци никада није била мања од 1,5 пута у односу на скупове високе корисности.

У резимеу овог експеримента, могу се извући следећи закључци у вези са резултатима на различитим базама података. При обради базе DiscountSales, највећа разлика у броју генерисаних скупова ставки забележена је између нашег алгоритма и његових конкурената. Ово додатно потврђује предности методе високо-просечне корисности у односу на методу високе корисности, јер је највеће смањење броја скупова настало на бази са највећим бројем трансакција.

База DiscountSales такође је показала највећу разлику у просечној корисности између скупова високо-просечне корисности и високе корисности, што је очекивано с обзиром на то да ова база има највећу укупну корисност.

Може се закључити да су скупови високо-просечне корисности, који обрађују мешовите и негативне вредности корисности, боља опција приликом обраде великих база података.

Најмање значајни резултати забележени су код базе података Superstore, осим највеће разлике у дужини скупова ставки. Скупови генерисани методом високо-просечне корисности показали су највећу разлику у дужини у односу на скупове високе корисности на бази са најмањим бројем различитих ставки.

Мањи број различитих ставки повећава вероватноћу генерисања скупова који садрже сличне комбинације ставки, што није пожељно. Ово наглашава ефикасност откривања скупова високо-просечне корисности, јер је успешно смањило број понављајућих скупова који су били присутни у резултатима методе високе корисности за ову малу базу.

На крају, чак и уз мању разлику у овом конкретном случају, алгоритам за откривање скупова високо-просечне корисности надмашио је алгоритам за откривање скупова високе корисности у овим експериментима.

Најзначајније предности откривања скупова ставки високо-просечне корисности, при обради мешовитих вредности профита, забележене су код базе података RetailNegative. На овој бази,

метода високо-просечне корисности значајно је надмашила методу високе корисности на свим спроведеним експериментима.

Метода високо-просечне корисности доследно је генерисала мањи број скупова ставки, краће скупове и већу просечну корисност по ставци, при чему су разлике биле приметне за све посматране вредности минималне корисности у поређењу са скуповима генерисаним методом високе корисности.

Сходно томе, скупови високо-просечне корисности са мешовитом корисношћу дали су боље резултате у односу на скупове високе корисности када оба приступа обрађују негативне вредности профита. Ова предност је посебно изражена код базе са најдужим трансакцијама и највећим бројем различитих ставки, што указује на њихову погодност за анализу најзахтевнијих типова база података.

5.2.3. Предности алгоритма MEHAUPM у односу на стандардне алгоритме за откривање скупова ставки високо-просечне корисности

У овој подсекцији истражићемо предности приступа откривања скупова ставки са високо-просечном корисношћу које обрађује мешовите и негативне вредности јединичног профита у поређењу са стандардним откривањем скупова ставки високо-просечне корисности. С обзиром на то да ће се једино разликовати скупови ставки који садрже ставке са мешовитом и негативном корисношћу, велике разлике се не очекују. Ипак, циљ нам је да нагласимо да коришћење методе високо-просечне корисности која обрађује негативне и мешовите вредности јединичног профита нема недостатака и пружа благе предности у односу на метод који ове вредности не обрађује. Поред тога, резултат тачније представља базу података која садржи ове негативне вредности.

Ови експерименти прате сличну методологију као у претходној подсекцији. Почећемо поређењем различитих скупова ставки које генеришу обе методе. За разлику од претходне подсекције, нећемо поредити дужину генерисаних скупова, јер не постоји значајна разлика у дужини. Вредности просечне дужине скупова ставки ове две методе на нашим базама података не разликују се за више од 0,01, што се не сматра значајним.

Уместо тога, извршићемо два теста за поређење корисности. Поредићемо корисност по скупу ставки, уместо по ставци као у претходној подсекцији. Ова метрика је релевантнија јер се разлика налази у ставкама које скупови садрже, а не у дужини самих скупова. Такође, поредићемо укупну корисност како бисмо приказали укупан добитак корисности приликом обраде мешовитих и негативних вредности јединичног профита ставки.

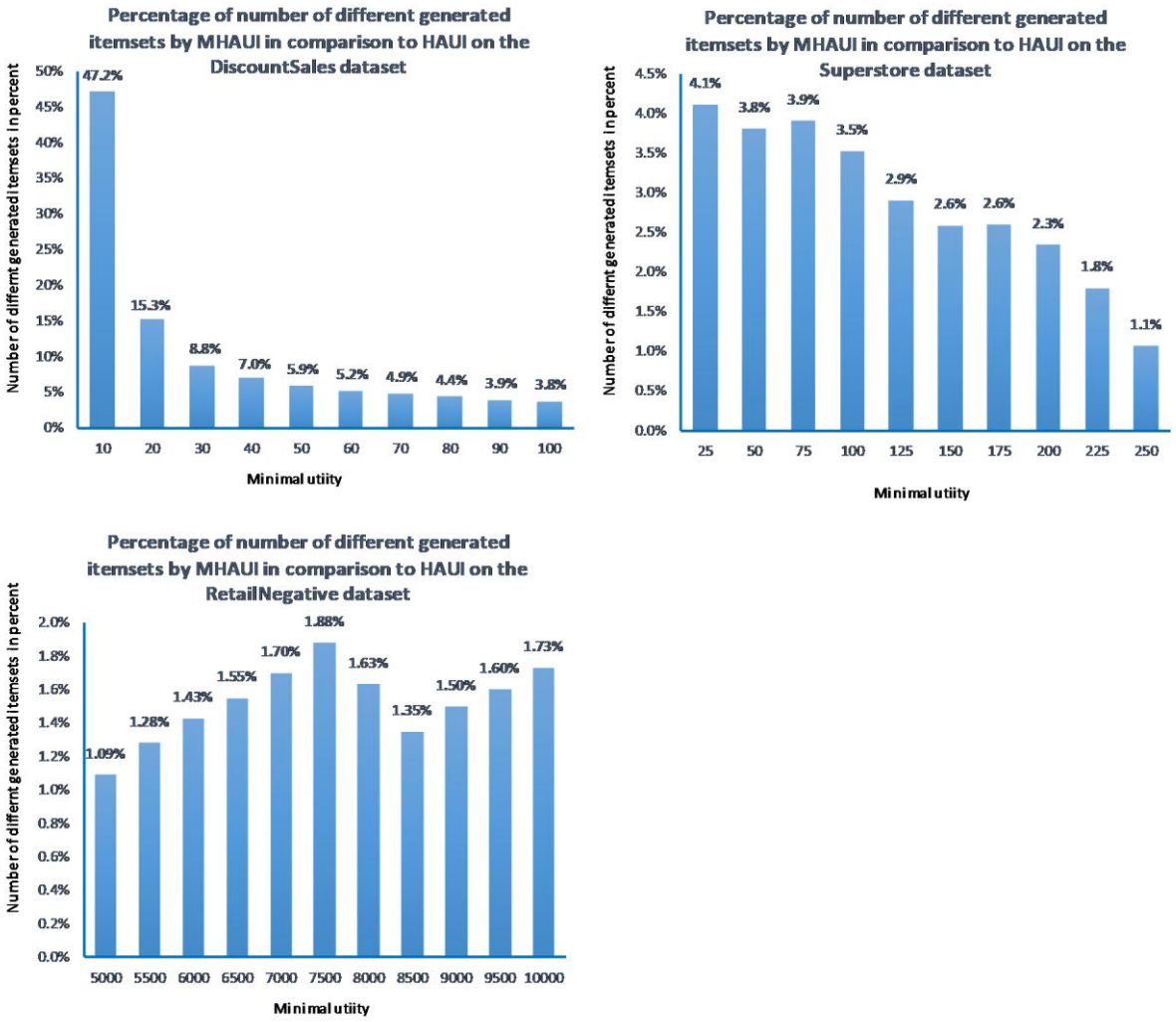
Горња граница прага корисности која показује варијације у генерисаним скуповима ставки између две методе откривања скупова ставки високо-просечне корисности на бази DiscountSales била је 800. Међутим, прва разлика већа од 1% јавила се при прагу корисности од 300. Примери изостављених скупова ставки, који оба садрже ставке са мешовитом корисношћу, укључују „Article i1002029 i1002049“ и „Article i1003981 i1002519“, са вредностима корисности 802 и 810, редом.

Иако су ови скупови имали довољну корисност да буду укључени у резултате при прагу од 800, стандардне методе високо-просечне корисности их игноришу и нетачно израчунавају због повременог присуства негативних вредности јединичног профита.

Анализа резултата представљених на Слици 13 показује да откривање скупова ставки високо-просечне корисности са и без обраде мешовитих и негативних вредности јединичног профита на овој бази података може дати различите резултате у распону од 3,8% до 47,2%. Иако се

разлика у генерисаним скуповима ставки смањује са повећањем прага корисности, она остаје значајна чак и при већим вредностима. Смањење ове разлике постаје постепеније за вредности корисности изнад 100. На пример, разлика се даље значајно не смањује и износи још 2,97% при прагу корисности од 150.

Битно је истаћи да ова база података показује најзначајнију разлику у броју генерисаних скупова ставки у поређењу са свим осталим испитиваним базама.



Слика 13. Проценат различитих скупова ставки који се генеришу при откривању скупова ставки са мешовитом и високо-просечном корисношћу уз обраду негативних вредности профита у односу на стандардно откривање скупова ставки високо-просечне корисности на базама података DiscountSales, Superstore и RetailNegative.

На бази Superstore, прва разлика у генерисаним скуповима ставки примећена је при прагу корисности од 750 када се смањивала минимална вредност корисности. Ова разлика је превазишла 1% при прагу од 700. Јасан пример изостављеног скупа ставки чак и при вишим вредностима корисности, као што је 750, је „Lg g3 phone; cisco 9971 ip video phone“, који има корисност од 794.

Неспособност стандардних метода високо-просечне корисности да обрађују негативне и мешовите вредности јединичног профита доводи до овог изостављања, што је потврђено чињеницом да једна ставка у скупу има мешовиту корисност која је неправилно третирана као негативна.

На овој бази података, разлика у резултатима између алгоритама за откривање скупова ставки високо-просечне корисности који обрађују и који не обрађују негативне и мешовите вредности корисности кретала се од 1,1% до 4,1%. Ова разлика се постепено смањује са повећањем минималног прага корисности. Напомињемо да ова база података показује најшире распоређену разлику у резултатима за већину прагова корисности који генеришу скупове ставки.

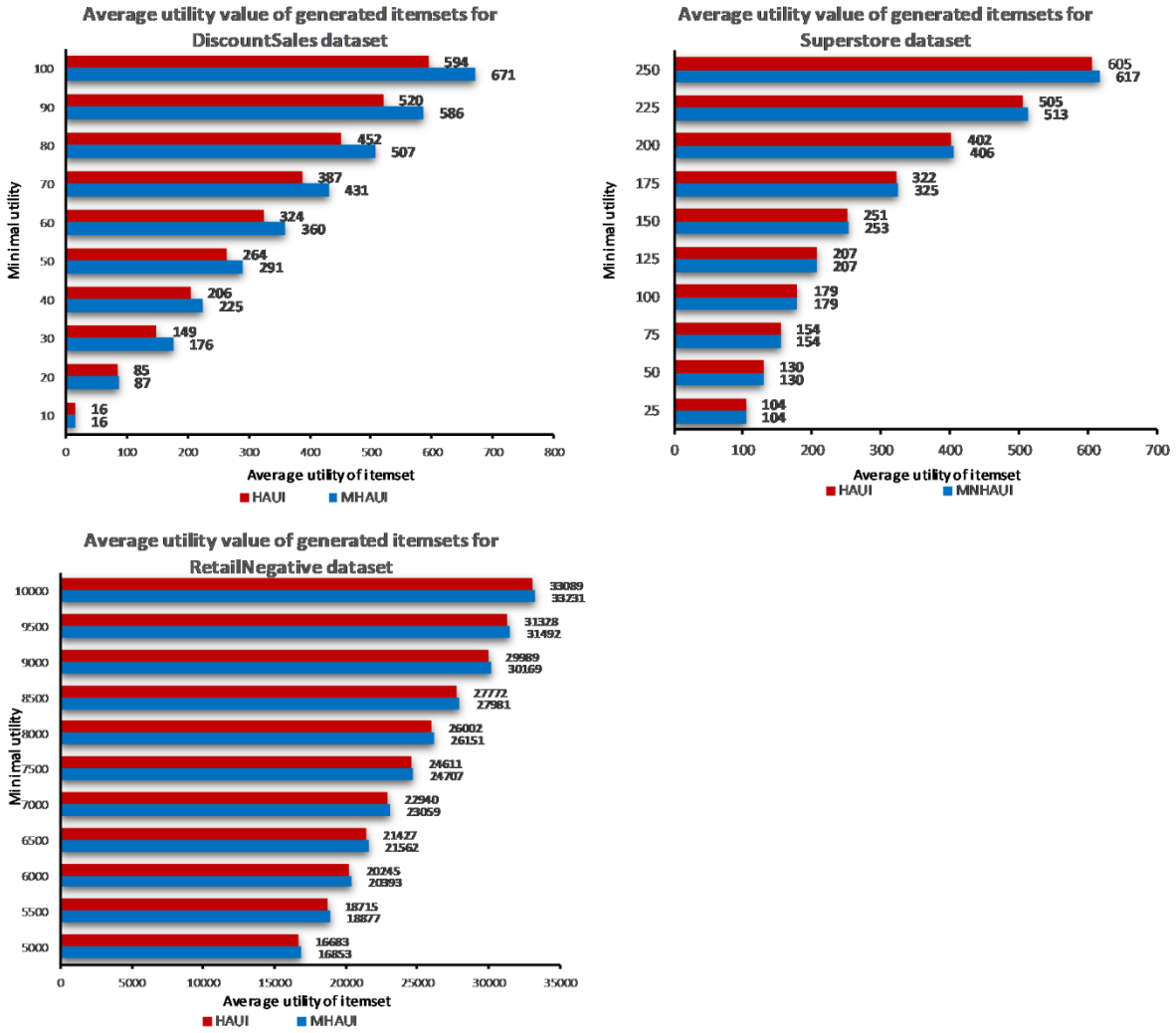
Најмања разлика је примећена на бази података RetailNegative. Највиши праг корисности који је довео до различитих скупова ставки био је 31.000. При прагу корисности од 27.000, разлика је прешла 1%. За минималне прагове корисности од 31.000 и 27.000, скупови ставки „38 41“ и „38 39 41“ су изостављени, редом. Ови скупови имали су вредности корисности од 31.880 и 27.905, али их стандардни алгоритам за откривање скупова ставки високо-просечне корисности није обрадио због негативне вредности корисности ставке 41.

База података RetailNegative показала је разлику у генерисаним скуповима ставки до 1,88% у посматраним минималним праговима корисности. Ова разлика се није константно смањивала, већ је благо осцилирала између 1% и 2%, а поново је почела да расте од прага корисности од 19.000, достижући максималну разлику од 5% при прагу од 27.000.

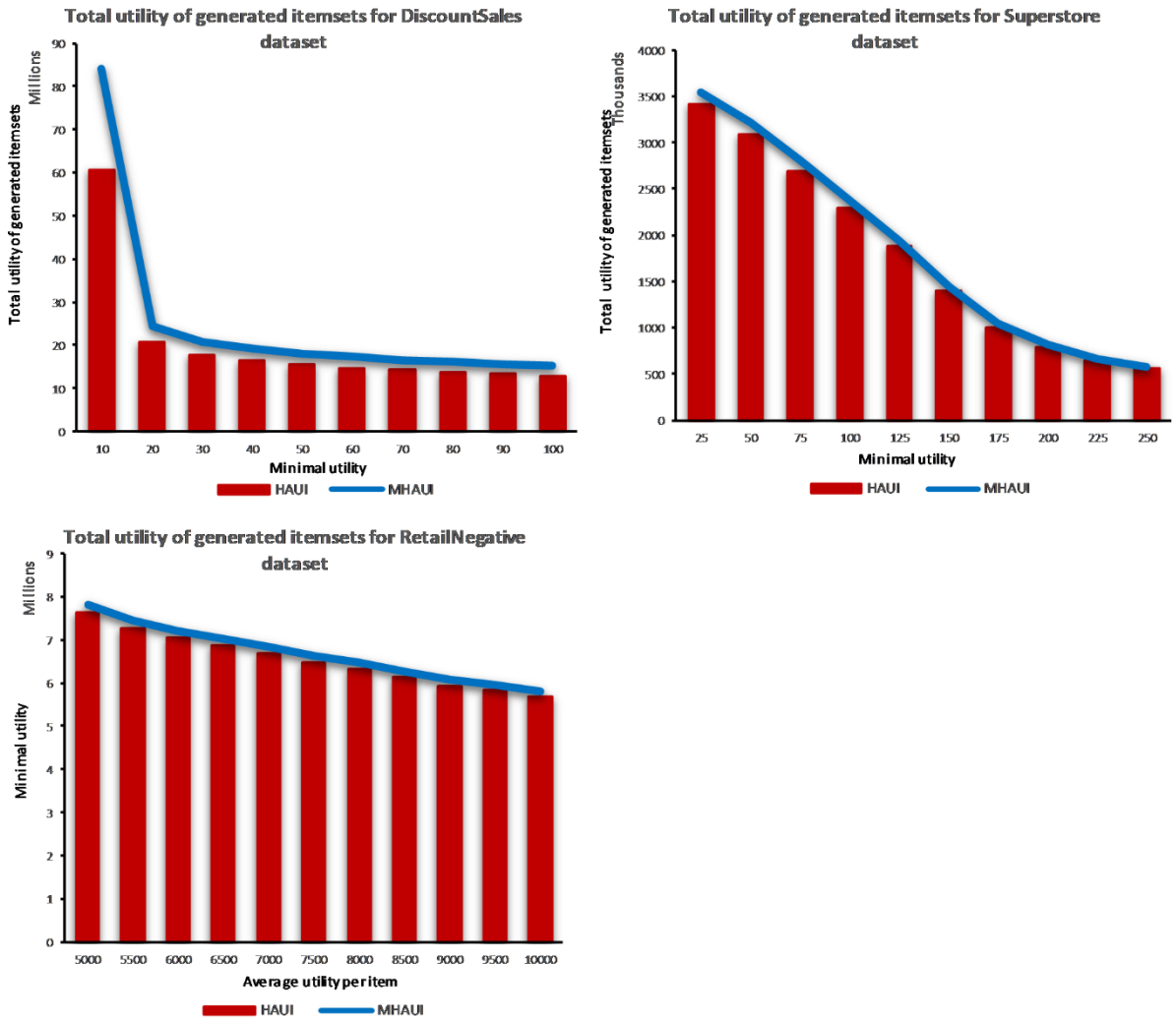
Циљ следећих експеримената је испитати однос између обраде ставки са мешовитим и негативним јединичним профитом и повећане корисности генерисаних скупова ставки, како просечно по скупу тако и укупно. Као што је раније поменуто, оба алгоритма за откривање скупова ставки високо-просечне корисности генеришу скупове без значајне разлике у дужини, те је поређење просечне корисности по скупу ставки прикладно и фер. Ови експерименти су потврдили да обрада ових вредности корисности доводи до већих вредности корисности, како просечно по скупу ставки, тако и укупно. Резултати су илустровани на Сликама 14 и 15.

Највећа разлика у просечној корисности по скупу ставки и у укупној корисности, међу посматраним базама података, примећена је на бази DiscountSales. Просечна корисност по скупу ставки била је идентична само при најнижем прагу корисности, након чега је постала већа него код стандардног алгоритма. Просечна корисност по скупу ставки коришћењем нашег алгоритма била је до 12% већа од оне код стандардног алгоритма. Вредност укупне корисности добијене нашим алгоритмом на овој бази кретала се од 15,6% до 32,4% више у односу на стандардни алгоритам. Укупна корисност излазних података значајно се разликовала између ова два алгоритма за све посматране прагове корисности.

Најмање значајно побољшање у просечној корисности по скупу ставки примећено је на бази Superstore. Просечна корисност по скупу ставки била је идентична за доњу половину посматраних минималних прагова корисности. Међутим, за горњу половину, просечна корисност по скупу ставки била је већа коришћењем нашег алгоритма, са повећањем до 1,96%. Како је праг корисности растао и алгоритми генерисали мање скупова ставки, разлика у просечној корисности постала је приметнија. Укупне вредности корисности биле су доследно најмање 3% веће када су обрађиване мешовите и негативне вредности корисности.



Слика 14. Просечна корисност по скупу ставки генерисаних помоћу откривања скупова ставки са мешовитом и високо-просечном корисношћу уз обраду негативних вредности профита у односу на стандардно откривање скупова ставки високо-просечне корисности на базама података DiscountSales, Superstore и RetailNegative.



Слика 15. Укупна корисност генерисаних скупова ставки помоћу откривања скупова ставки са мешовитом и високо-просечном корисношћу уз обраду негативних вредности профита у односу на стандардно откривање скупова ставки високо-просечне корисности на базама података DiscountSales, Superstore и RetailNegative.

На бази RetailNegative, просечна корисност по скупу ставки била је доследно различита када су обрађиване у односу на необрађене негативне и мешовите вредности корисности. Разлика је била мања, али конзистентна широм базе, при чему је наш алгоритам давао 0,43% до 1% веће просечне вредности корисности по скупу ставки. Укупна корисност скупова ставки високо-просечне корисности са обрађеним негативним и мешовитим вредностима била је између 2,1% и 2,3% већа него код стандардних скупова ставки високо-просечне корисности. Ова база је показала најшире распрострањене и доследно боље резултате у посматраним минималним праговима корисности када се обрађују негативне и мешовите јединичне вредности корисности.

У закључку овог подпоглавља, анализираћемо резултате добијене на посматраним базама података. Највећа разлика у укупној корисности и просечној корисности по скупу ставки између скупова ставки високо-просечне корисности са и без обрађених негативних и мешовитих вредности примећена је на бази DiscountSales. Наш алгоритам дао је највећу разлику у корисности за базу са највећом укупном корисношћу и бројем трансакција. Разлог за то може бити што са великим бројем трансакција постоји и више комбинација ставки са уравнотеженим негативним и позитивним вредностима корисности, које конкуренти алгоритам пропушта или неправилно израчунава.

На бази Superstore, разлике у генерисаним скуповима ставки и у укупној корисности су имале најшире распоне. С обзиром на то да је ова база била најмање разнолика, доследне разлике у вредностима нису изненађујуће. Међутим, можемо закључити да је на базама са краћим и дужим трансакцијама откривање скупова ставки високо-просечне корисности уз обраду мешовитих и негативних вредности дало боље резултате у односу на оно које их не обрађује.

На бази RetailNegative примећена је највећа разлика у генерисаним резултатима и најдоследнија разлика у просечној корисности по скупу ставки. Ова база има најдужи низ трансакција и највећи број различитих ставки. Са разноврснијим скуповима ставки, постоји више ставки са мешовитим и негативним јединичним вредностима корисности, па је највећа разлика у генерисаним скуповима ставки очекивана. Најдоследнија разлика у просечној корисности по скупу ставки представља позитивну особину, показујући да наш алгоритам даје боље резултате при раду са базама података са великим бројем различитих ставки. У целини, откривање скупова ставки високо-просечне корисности уз обраду негативних и мешовитих вредности корисности показало се бољим у односу на стандардно откривање скупова ставки високо-просечне корисности.

У овом подпоглављу извршили смо више експеримената како бисмо демонстрирали перформансе алгоритма за откривање скупова ставки високо-просечне корисности уз обраду мешовитих и негативних вредности корисности. У првој групи експеримената потврдили смо да наш алгоритам надмашује алгоритме за откривање скупова ставки високе корисности по питању и брзине и потрошње меморије када оба алгоритма обрађују негативне јединичне вредности корисности. Друга група експеримената показала је да откривање скупова ставки високо-просечне корисности даје боље резултате од алгоритма за откривање високе корисности када се обрађују мешовите и негативне вредности добити у свим посматраним базама података. Овај напредак огледа се у краћим генерисаним скуповима ставки и већој просечној корисности по ставци. У последњој групи тестова упоређивали смо алгоритме за откривање високо-просечне корисности који обрађују и који не обрађују негативне вредности. Експерименти су показали да постоји значајна разлика у резултатима када база садржи негативне и мешовите јединичне вредности добити које нису адекватно обрађене. Стога, ако се ове вредности правилно обраде, може се очекивати повећање просечне корисности по скупу ставки и укупне корисности генерисаних скупова ставки. Поред тога, наш алгоритам је надмашио конкуренцију на свим посматраним базама података.

Спроведена експериментална анализа потврдила је практичну применљивост, ефикасност и робусност предложених алгоритама за откривање образаца у сложеним сценаријима података. Евалуација PISRuleGrowth алгоритма показала је његову способност да ефикасно открива периодична унутар-секвенцијална правила, при чему је остварио значајне предности у погледу брзине извршавања на базама са великим бројем секвенци и ставки, захваљујући оптимизованој структури мапе појављивања. Иако је на скуповима са великим бројем трансакција по секвенци заостајао за ERMiner алгоритмом, PISRuleGrowth је доследно надмашио RuleGrowth алгоритам и показао повољан компромис између времена извршавања и потрошње меморије, нарочито при мањем броју генерисаних правила. Са друге стране, MEHAUPM алгоритам се истакао као супериорно решење за откривање скупова ставки високо-просечне корисности у присуству мешовитих и негативних вредности. Поред тога што је остварио најкраћа времена извршавања у поређењу са конкурентским алгоритмима (FHN, EHIN, PHMNPlus) на свим тестираним базама, MEHAUPM је генерисао квалитетније резултате – краће скупове ставки са већом просечном корисношћу по ставци, чиме је смањена редундантност и повећана информативност резултујућих образаца. Посебно је значајно што је обрада мешовитих и негативних вредности корисности довела до веће укупне корисности и просечне корисности по скупу у односу на стандардне приступе који ове вредности занемарују. Укупно гледано, резултати експеримената недвосмислено показују да предложени алгоритми представљају вредне доприносе у областима откривања секвенцијалних правила и откривања

корисних образаца, пружајући истраживачима и практичарима моћне алате за анализу сложених реалних података уз истовремено обезбеђивање високих перформанси и релевантности добијених резултата.

6. Закључак

У закључном поглављу овог рада обједињени су доприноси из две различите, али концептуално повезане истраживачке целине: откривања периодичних унутар-секвенцијалних правила заједничких за више секвенци и откривања скупова ставки високо-просечне корисности уз обраду мешовитих и негативних вредности јединичне добити.

У првом делу рада уведен је проблем откривања периодичних унутар-секвенцијалних правила заједничких за више секвенци. Концепт претраживања периодичних образаца унутар секвенци, карактеристичан за област откривања стандардних периодичних образаца (periodic pattern mining), адаптиран је у домену откривања секвенцијалних правила. Предложен је нови алгоритам, PISRuleGrowth, који користи приступ раста образаца (pattern-growth) за генерисање правила и структуру мапе података ради ефикаснијег управљања информацијама о периодичности. Експериментална евалуација на реалним скуповима података показала је да алгоритам постиже супериорне или упоредиве резултате у односу на постојеће приступе, уз повољан однос времена извршавања и потрошње меморије, осим у случајевима изузетно великих база са великим бројем трансакција. У поређењу са неоптимизованом верзијом, унапређени алгоритам је постигао значајна побољшања у брзини и, у већини случајева, у меморијској ефикасности, посебно када се генерише велики број периодичних унутар-секвенцијалних правила. Резултати су потврдили да предложени приступ омогућава ефикасно и скалабилно откривање периодичних правила у разноврсним базама података.

Поред постигнутих резултата, анализирана су и ограничења, као и могућности за даљи развој. У контексту откривања периодичних образаца, значајан потенцијал лежи у проширењу алгоритма на флексибилне периодичне обрасце, где се мање релевантни догађаји могу изоставити ради откривања значајнијих регуларности. Интеграција техника као што су дискретизација вредности и ефикасније структуре података могла би додатно унапредити перформансе. Такође, с обзиром на динамичку природу савремених база података, увођење инкременталног откривања или откривања правила у реалном времену представља значајан правац будућег истраживања. Додатно, примена концепта топ-к стабилних периодичних образаца могла би повећати употребљивост алгоритма, смањити зависност од ручног подешавања параметара и омогућити откривање стабилнијих и предвидљивијих образаца. Потенцијалне примене обухватају анализу продаје, предвиђање одустајања клијената, предвиђање одржавања система и интеграцију у системе за препоруке, као и проширење на откривање ретких периодичних унутар-секвенцијалних правила.

Посебно занимљив правац представља могућност коришћења откривених периодичних унутар-секвенцијалних правила као структурисаног извора знања у савременим ВИ системима. Таква правила би могла бити организована у репозиторијум понављајућих временских образаца који би се претраживао приликом анализе секвенцијалних података, на пример у системима заснованим на претраживању релевантног контекста или у агентским системима који комбинују генеративне моделе са специјализованим аналитичким компонентама. На тај начин, резултати алгоритма не би служили само за самосталну анализу секвенцијалних база, већ и као проверљив и интерпретабилан слој знања за системе који кориснику треба да објасне уочене обрасце понашања или да их искористе у процесу доношења препорука и одлука.

Други део рада бавио се критичним изазовом у откривању скупова ставки високе и високо-просечне корисности, односно адекватном обрадом ставки чије се јединичне вредности добити динамички мењају између позитивних и негативних. Формално је дефинисан појам мешовитих ставки и истакнут њихов значај у реалним пословним сценаријима, нарочито у условима промоција, попуста и стратегија продаје са губитком ради дугорочног позиционирања на тржишту. Интеграцијом концепта мешовите корисности са ограничењима високо-просечне корисности и модификацијом постојећег ефикасног алгоритма развијен је MEHAUPM, који гарантује исправно и потпуно генерисање скупова ставки високо-просечне корисности у окружењима са динамичким променама предзнака добити.

Емпиријска анализа на реалним скуповима података обогаћеним сценаријима попуста и промоција показала је да MEHAUPM надмашује постојеће алгоритме како у погледу времена извршавања и потрошње меморије, тако и у квалитативним карактеристикама добијених резултата. Алгоритам је генерисао краће скупове ставки са већом просечном корисношћу по ставци у поређењу са приступима високе корисности, као и скупове ставки са већом просечном корисношћу по скупу и већом укупном корисношћу у односу на стандардне алгоритме високо-просечне корисности који не обрађују мешовите вредности. Конзистентне разлике у резултатима потврђују да адекватна обрада мешовитих и негативних вредности има суштински утицај на квалитет добијених образаца и пословних увида.

Као и у првом делу рада, идентификовани су правци будућег истраживања. Посебно је значајно проширење алгоритма на инкрементално окружење и окружење у реалном времену, што би омогућило праћење промена профитабилности у динамичким базама података. Додатно, принципи обраде мешовите корисности могу се применити на друге варијанте откривања образаца, као што су затворени, максимални или минимални скупови високе и високо-просечне корисности.

Важан правац даљег развоја представља и повезивање оваквих алгоритама са савременим ВИ системима за подршку одлучивању. У хибридним архитектурама заснованим на великим језичким моделима и агентском начину рада, MEHAUPM би могао да функционише као специјализовани аналитички модул који се активира приликом процене профитабилности комбинација производа, анализе ефеката попуста, планирања промотивних стратегија или објашњавања добијених препорука. Уместо да генеративни модел самостално процењује економску вредност образаца на основу неструктурираних података, резултати добијени овим алгоритмом могли би да представљају проверљиву аналитичку основу за формулисање прецизнијих и транспарентнијих препорука.

Посматрани заједно, предложени приступи отварају простор за њихово укључивање у савремене хибридне и неуро-симболичке архитектуре ВИ, у којима се статистички и генеративни модели допуњују експлицитно откривеним, формално дефинисаним и интерпретабилним обрасцима. Периодична унутар-секвенцијална правила могу обезбедити структуру за разумевање понављајућих временских односа, док скупови ставки високо-просечне корисности са мешовитим вредностима могу обезбедити поуздану основу за процену значаја и профитабилности сложених трансакционих образаца. На тај начин, алгоритми развијени у овом раду могли би у будућности да имају улогу не само самосталних метода откривања образаца, већ и специјализованих компоненти у ширим интелигентним системима који комбинују претрагу знања, аналитичку прецизност, објашњивост и генеративне могућности савремених модела.

На основу формализације истраживачких проблема, развоја предложених алгоритамских решења и спроведене експерименталне евалуације, може се закључити да су постављене хипотезе потврђене. Хипотеза H0.1 потврђена је дефинисањем и реализацијом модела периодичних унутар-секвенцијалних правила, којим се омогућава откривање понављајућих временских односа који нису обухваћени традиционалним приступима откривању

секвенцијалних правила. Хипотеза H0.2 потврђена је резултатима експеримената који показују да предложена алгоритамска реализација омогућава ефикасно генерисање ове класе правила, уз конкурентне временске и меморијске перформансе у односу на референтне алгоритме. Хипотеза H0.3 потврђена је анализом тачности откривања скупова ставки у присуству мешовитих вредности корисности, при чему је показано да предложени приступ избегава преурањено елиминисање валидних образаца, што је проблем који се јавља код постојећих решења. Хипотеза H0.4 потврђена је резултатима поређења са приступима заснованим на укупној корисности, који показују да примена просечне корисности у условима мешовите корисности доводи до издвајања концизнијих скупова ставки са већом аналитичком вредношћу.

Рад даје допринос у две комплементарне области откривања законитости у подацима: проширује домен откривања секвенцијалних образаца увођењем периодичности у оквиру више секвенци и унапређује откривање корисних скупова омогућавајући коректну обраду мешовитих и негативних вредности добити. Заједнички именитељ оба доприноса јесте прилагођавање алгоритама реалним, динамичким и комплексним подацима, чиме се повећава њихова практична вредност и применљивост у савременим аналитичким системима.

Референце

Agrawal R, Srikant R (1995) Mining sequential patterns. In: Proceedings of the 11th international con ference on data engineering. IEEE, pp, 3–14. https://doi.org/10.1109/ICDE.1995.380415

Ahmed, C. F., Tanbeer, S. K., & Jeong, B. S. (2010, June). Mining high utility web access sequences in dynamic web log data. In 2010 11th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (pp. 76-81). IEEE. https://doi.org/10.1109/SNPD.2010.21

Ahmed, C. F., Tanbeer, S. K., & Jeong, B. S. (2011). A framework for mining high utility web access sequences. IETE technical review, 28(1), 3-16. https://doi.org/10.4103/0256-4602.74506

Ahmed, C. F., Tanbeer, S. K., Jeong, B. S., & Lee, Y. K. (2009, February). Efficient mining of utility-based web path traversal patterns. In 2009 11th International Conference on Advanced Communication Technology (Vol. 3, pp. 2215-2218). IEEE. https://doi.org/10.5555/1701655.1701797.

Alimo, A. (2023). Market Sales [Dataset]. Kaggle. https://www.kaggle.com/datasets/abdelrahmanalimo/sales-ecommerce

Altuntas, F., & Gök, M. Ş. (2021). Analysis of patent documents with utility mining: a case study of wind energy technology. Kybernetes, 50(9), 2548-2582. https://doi.org/10.1108/K-06-2020-0365

Amirat H, Benslimane A, Fournier-Viger P, Lagraa N (2018). Locrec: rule-based successive location recommendation in lbsn. In: 2018 IEEE international conference on communications (ICC), IEEE, pp 1–6 https://doi.org/10.1109/ICC.2018.8422183

Ashraf, M., Abdelkader, T., Rady, S., & Gharib, T. F. (2022). TKN: an efficient approach for discovering top-k high utility itemsets with positive or negative profits. Information Sciences, 587, 654-678. https://doi.org/10.1016/j.ins.2021.12.024

Bapna, C., Reddy, P. K., & Mondal, A. (2020, October). Improving product placement in retail with generalized high-utility itemsets. In 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA) (pp. 60-69). IEEE. https://doi.org/10.1109/DSAA49011.2020.00018

Cesario E (2023) Big data analytics and smart cities: applications, challenges, and opportunities. Front Big Data 6:1149402 https://doi.org/10.3389/fdata.2023.1149402

Cesario E, Folino F, Manco G, PontieriL (2005) An incremental clustering scheme for duplicate detection in large databases. In: 9th International database engineering & application symposium (IDEAS'05), IEEE, pp 89–95 https://doi.org/10.1109/IDEAS.2005.10

Chu, C. J., Tseng, V. S., & Liang, T. (2009). An efficient algorithm for mining high utility itemsets with negative item values in large databases. Applied Mathematics and Computation, 215(2), 767-778. https://doi.org/10.1016/j.amc.2009.05.066

Data Mining Cup. (n.d.). Data Mining Cup. Retrieved July 12, 2025, from https://www.data-mining-cup.com/

Demir, S., Alkan, O., Cekinel, F., & Karagoz, P. (2019). Extracting potentially high profit product feature groups by using high utility pattern mining and aspect based sentiment analysis. High-Utility Pattern Mining: Theory, Algorithms and Applications, 233-260. https://doi.org/10.1007/978-3-030-04921-8_9

Fahed L, Lenca P, Haralambous Y, Lefort R (2020) Distant event prediction based on sequential rules. Data Sci Pattern Recogni 4(1):1–23

Fournier-Viger, P. (2014). FHN: efficient mining of high-utility itemsets with negative unit profits. In Advanced Data Mining and Applications: 10th International Conference, ADMA 2014, Guilin, China, December 19-21, 2014. Proceedings 10 (pp. 16-29). Springer International Publishing. https://doi.org/10.1007/978-3-319-14717-8_2

Fournier-Viger, P., & Zida, S. (2015, April). FOSHU: faster on-shelf high utility itemset mining--with or without negative unit profit. In Proceedings of the 30th annual ACM symposium on applied computing (pp. 857-864). https://doi.org/10.1145/2695664.2695823

Fournier-Viger, P., Chun-Wei Lin, J., Truong-Chi, T., & Nkambou, R. (2019a). A survey of high utility itemset mining. High-utility pattern mining: Theory, algorithms and applications, 1-45. https://doi.org/10.1007/978-3-030-04921-8_1

Fournier-Viger, P., Wu, C. W., Zida, S., & Tseng, V. S. (2014b). FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning. In Foundations of Intelligent Systems: 21st International Symposium, ISMIS 2014, Roskilde, Denmark, June 25-27, 2014. Proceedings 21 (pp. 83-92). Springer International Publishing. https://doi.org/10.1007/978-3-319-08326-1_9

Fournier-Viger P, Faghihi U, Nkambou R, Nguifo EM (2010) CMRULES: an efficient algorithm for mining sequential rules common to several sequences. Knowledge-Based Systems, 25(1), 63–76 https://doi.org/10.1016/j.knosys.2011.07.005

Fournier-Viger P, Gueniche T, Zida S, Tseng VS (2014a) ERMiner: sequential rule mining using equiv alence classes. In: Advances in intelligent data analysis XIII: 13th international symposium, IDA 2014, Leuven, Belgium, October 30–November 1, 2014. Proceedings 13,. Springer International Publishing, pp 108–119 https://doi.org/10.1007/978-3-319-12571-8_10

Fournier-Viger P, Li Z, Lin JCW, Kiran RU, Fujita H (2019b) Efficient algorithms to identify periodic patterns in multiple sequences. Inf Sci 489:205–226 https://doi.org/10.1016/j.ins.2019.03.050

Fournier-Viger P, Lin CW, Duong QH, Dam TL, ŠevcíkL, Uhrin D, Voznak M (2017) PFPM: discovering periodic frequent patterns with novel periodicity measures. In: Proceedings of the 2nd Czech-China scientific conference 2016, IntechOpen https://doi.org/10.5772/66780

Fournier-Viger P, Lin JCW, Duong QH, Dam TL (2016) PHM: mining periodic high-utility itemsets. In: Advances in data mining. Applications and theoretical aspects: 16th industrial conference, ICDM 2016, New York, NY, USA, July 13–17, 2016. Proceedings 16, Springer International Publishing, pp 64–79 https://doi.org/10.1007/978-3-319-41561-1_6

Fournier-Viger P, Nkambou R, Tseng VSM (2011) RuleGrowth: mining sequential rules common to several sequences by pattern-growth. In Proceedings of the 2011 ACM symposium on applied computing, pp 956–961 https://doi.org/10.1145/1982185.1982394

Fournier-Viger P, Tseng VS (2011) Mining top-k sequential rules. In: Advanced data mining and appli cations: 7th international conference, ADMA 2011, Beijing, China, December 17-19, 2011, proceedings, part II 7, Springer Berlin Heidelberg, pp 180–194 https://doi.org/10.1007/978-3-642-25856-5_14

Fournier-Viger P, Tseng VS (2013) TNS: mining top-k non-redundant sequential rules. In: Proceedings of the 28th annual ACM symposium on applied computing, pp 164–166 https://doi.org/10.1145/2480362.2480395

Fournier-Viger P, Wu C. W., Tseng, V. S., & Nkambou, R. (2012). Mining sequential rules common to several sequences with the window size constraint. In: Advances in artificial intelligence: 25th Cana dian conference on artificial intelligence, Canadian AI 2012, Toronto, ON, Canada, May 28–30, 2012. Proceedings 25, Springer Berlin Heidelberg, pp 299–304 https://doi.org/10.1007/978-3-642-30353-1_27

Fournier-Viger P, Wu CW, Tseng VS (2013) Mining maximal sequential patterns without candidate maintenance. In: Advanced data mining and applications: 9th international conference, ADMA 2013, Hangzhou, China, December 14-16, 2013, proceedings, part I 9. Springer Berlin Heidelberg, pp 169–180 https://doi.org/10.1007/978-3-642-53914-5_15

Fournier-Viger P, Yang P, Kiran RU, Ventura S, Luna JM (2021) Mining local periodic patterns in a discrete sequence. Inf Sci 544:519–548 https://doi.org/10.1016/j.ins.2020.09.044

Fournier-Viger P, Yang P, Li Z, Lin JCW, Kiran RU (2020) Discovering rare correlated periodic patterns in multiple sequences. Data Knowl Eng 126:101733 https://doi.org/10.1016/j.datak.2019.101733

Fournier-Viger P, Yang P, Lin JCW, Kiran RU (2019c) Discovering stable periodic-frequent patterns in transactional data. In: Advances and trends in artificial intelligence. From theory to practice: 32nd international conference on industrial, engineering and other applications of applied intelligent systems, IEA/AIE 2019, Graz, Austria, July 9–11, 2019, Proceedings 32, Springer International Publishing, pp 230–244 https://doi.org/10.1007/978-3-030-22999-3_21

Gan, W., Chen, G., Yin, H., Fournier-Viger, P., Chen, C. M., & Yu, P. S. (2022). Towards revenue maximization with popular and profitable products. ACM/IMS Transactions on Data Science (TDS), 2(4), 1-21. https://doi.org/10.1145/3488058

Gan, W., Lin, J. C. W., Fournier-Viger, P., Chao, H. C., & Fujita, H. (2018b). Extracting non-redundant correlated purchase behaviors by utility measure. Knowledge-Based Systems, 143, 30-41. https://doi.org/10.1016/j.knosys.2017.12.003

Gan, W., Lin, J. C. W., Fournier-Viger, P., Chao, H. C., Hong, T. P., & Fujita, H. (2018a). A survey of incremental high-utility itemset mining. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(2), e1242. https://doi.org/10.1002/widm.1242

Gan, W., Wan, S., Chen, J., Chen, C. M., & Qiu, L. (2020, December). TopHUI: Top-k high-utility itemset mining with negative utility. In 2020 IEEE International Conference on Big Data (Big Data) (pp. 5350-5359). IEEE. https://doi.org/10.1109/BigData50022.2020.9378288

Garofalakis M, Rastogi R, Shim K(2002) Mining sequential patterns with regular expression constraints. IEEE Trans Knowl Eng 14(3):530–552 https://doi.org/10.1109/TKDE.2002.1000341

Han, J., Cheng, H., Xin, D., & Yan, X. (2007). Frequent pattern mining: current status and future directions. Data mining and knowledge discovery, 15(1), 55-86. https://doi.org/10.1007/s10618-006-0059-1

Hu, J., & Mojsilovic, A. (2007). High-utility pattern mining: A method for discovery of high-utility item sets. Pattern Recognition, 40(11), 3317-3324. https://doi.org/10.1016/j.patcog.2007.02.003

Ismail, W., Hassan, M. M., & Fortino, G. (2017, May). Productive-associated periodic high-utility itemsets mining. In 2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC) (pp. 637-642). IEEE. https://doi.org/10.1109/ICNSC.2017.8000165

Jabbar, M. A., Deekshatulu, B. L., & Chandra, P. (2015, December). A novel algorithm for utility-frequent itemset mining in market basket analysis. In Innovations in Bio-Inspired Computing and Applications: Proceedings of the 6th International Conference on Innovations in Bio-Inspired Computing and Applications (IBICA 2015) held in Kochi, India during December 16-18, 2015 (pp. 337-345). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-28031-8_29

Kavitha, V., & Geetha, B. G. (2016). High Utility Itemset Mining with Influential Cross Selling Items from Transactional Database. Int J AdvEngg Tech/Vol. VII/Issue II/April-June, 820, 826.

Kryszkiewicz M (1998) Representative association rules and minimum condition maximum consequence association rules. In: Principles of data mining and knowledge discovery: 2nd European symposium, PKDD'98 Nantes, France, September 23–26,1998 proceedings 2, Springer Berlin Heidelberg, pp361–369 https://doi.org/10.1007/BFb0094839

Lai, F., Zhang, X., Chen, G., & Gan, W. (2023). Mining periodic high-utility itemsets with both positive and negative utilities. Engineering Applications of Artificial Intelligence, 123, 106182 https://doi.org/10.1016/j.engappai.2023.106182

Lan, G. C., Hong, T. P., & Tseng, V. S. (2011). Discovery of high utility itemsets from on-shelf time periods of products. Expert Systems with Applications, 38(5), 5851-5857. https://doi.org/10.1016/j.eswa.2010.11.040

Liang, S., Liu, Y., Jian, L., Gao, Y., & Lin, Z. (2011, August). A utility-based recommendation approach for academic literatures. In 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (Vol. 3, pp. 229-232). IEEE. https://doi.org/10.1109/WI-IAT.2011.110

Lin, C. W., Hong, T. P., & Lu, W. H. (2010). Efficiently mining high average utility itemsets with a tree structure. In Intelligent Information and Database Systems: Second International Conference, ACIIDS, Hue City, Vietnam, March 24-26, 2010. Proceedings, Part I 2 (pp. 131-139). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-12145-6_14

Lin, J. C. W., Li, T., Fournier-Viger, P., Hong, T. P., Zhan, J., & Voznak, M. (2016). An efficient algorithm to mine high average-utility itemsets. Advanced Engineering Informatics, 30(2), 233-243. https://doi.org/10.1016/j.aei.2016.04.002

Lin, J. C. W., Ren, S., Fournier-Viger, P., & Hong, T. P. (2017a). EHAUPM: Efficient high average-utility pattern mining with tighter upper bounds. IEEE Access, 5, 12927-12940. https://doi.org/10.1109/ACCESS.2017.2717438

Lin, Y. F., Huang, C. F., & Tseng, V. S. (2017b). A novel methodology for stock investment using high utility episode mining and genetic algorithm. Applied Soft Computing, 59, 303-315. https://doi.org/10.1016/j.asoc.2017.05.032

Liu, Y., Liao, W. K., & Choudhary, A. (2005). A two-phase algorithm for fast discovery of high utility itemsets. In Advances in Knowledge Discovery and Data Mining: 9th Pacific-Asia Conference, PAKDD 2005, Hanoi, Vietnam, May 18-20, 2005. Proceedings 9 (pp. 689-695). Springer Berlin Heidelberg. https://doi.org/10.1007/11430919_79

Liu X, Sang X, Chang J, Zheng Y, Han Y (2021) The water supply association analysis method in Shenzhen based on kmeans clustering discretization and apriori algorithm. PLoS ONE 16(8):e0255684 https://doi.org/10.1371/journal.pone.0255684

Lo D, Khoo SC, Wong L (2009) Non-redundant sequential rules—theory and algorithm. Inf Syst 34(4–5):438–453 https://doi.org/10.1016/j.is.2009.01.002

Manike, C., & Om, H. (2014). High-Utility Patterns Discovery in Data Mining: A Case Study. Case Studies in Intelligent Computing: Achievements and Trends, 247. https://doi.org/10.1201/b17333-12

Mannila H, Toivonen H, Verkamo AI (1997) Discovery of frequent episodes in event sequences. Data Min Knowl Disc 1:259–289 https://doi.org/10.1023/A:1009748302351

Mondal, A., Mittal, R., Khandelwal, V., Chaudhary, P., & Reddy, P. K. (2021, October). PEAR: a product expiry-aware and revenue-conscious itemset placement scheme. In 2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA) (pp. 1-10). IEEE. https://doi.org/10.1109/DSAA53316.2021.9564189

Nguyen HQ, Pham TT, Vo V, Vo B, Quan TT (2018)The predictive modeling for learning student results based on sequential rules. Int J Innov Comput Inf Control 14(6):2129–2140 https://doi.org/10.24507/ijicic.14.06.2129

Patil, S. K., & Bojewar, S. (2017, May). Application of high utility mining for pattern prediction. In 2017 International Conference on Trends in Electronics and Informatics (ICEI) (pp. 1154-1158). IEEE. https://doi.org/10.1109/ICOEI.2017.8300895

Pei J, Han J, Mortazavi-Asl B, Wang J, Pinto H, Chen Q, Dayal U, Hsu MC (2004) Mining sequential patterns by pattern-growth: the prefixspan approach. IEEE Trans Knowl Data Eng 16(11):1424–1440 https://doi.org/10.1109/TKDE.2004.77

Sagare SC, Shirgave SK, Kodavade DV (2020) A system for predictive data analytics using sequential rule mining. Int J Softw Innov (IJSI) 8(4):78–94 https://doi.org/10.4018/IJSI.2020100107

Shie, B. E., Hsiao, H. F., Tseng, V. S., & Yu, P. S. (2011). Mining high utility mobile sequential patterns in mobile commerce environments. In Database Systems for Advanced Applications: 16th International Conference, DASFAA 2011, Hong Kong, China, April 22-25, 2011, Proceedings, Part I 16 (pp. 224-238). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-20149-3_18

Singh, K., Shakya, H. K., Singh, A., & Biswas, B. (2018). Mining of high-utility itemsets with negative utility. Expert Systems, 35(6), e12296. https://doi.org/10.1111/exsy.12296

Singh H, Kaur M, Kaur P (2017) Web page recommendation system based on partially ordered sequential rules. J Intell Fuzzy Syst 32(4):3009–3015 https://doi.org/10.3233/JIFS-169244

Singh I, Manuja M, Mathur R, Goswami M (2020) Detecting intrusive transactions in databases using partially-ordered sequential rule mining and fractional-distance based anomaly detection. Int J Intell Eng Inf 8(2):138–171 https://doi.org/10.1504/IJIEI.2020.109098

SPMF. (n.d.). A Java Open-Source Data Mining Library: Datasets. http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php

Tableau. (n.d.). Global Superstore [Data file]. Retrieved July 12, 2025, from http://www.tableau.com/sites/default/files/training/global_superstore.zip

Tanbeer SK, Ahmed CF, Jeong BS, Lee YK (2009) Discovering periodic-frequent patterns in transactional databases. In: Advances in knowledge discovery and data mining: 13th Pacific-Asia conference, PAKDD 2009 Bangkok, Thailand, April 27–30, 2009 proceedings 13. Springer Berlin Heidelberg, pp 242–253 https://doi.org/10.1007/978-3-642-01307-2_24

Tzvetkov P, Yan X, Han J (2005) Tsp: Mining top-k closed sequential patterns. Knowl Inf Syst 7(4):438–457 https://doi.org/10.1007/s10115-004-0175-4

UCI Machine Learning Repository. (n.d.). Datasets. https://archive.ics.uci.edu/ml/datasets/

Weng, C. H. (2016). Discovering highly expected utility itemsets for revenue prediction. Knowledge-Based Systems, 104, 39-51. https://doi.org/10.1016/j.knosys.2016.04.009

Yan X, Han J, Afshar R (2003) Clospan: mining: closed sequential patterns in large datasets. In: Pro ceedings of the 2003 SIAM international conference on data mining. Society for industrial and applied mathematics, pp 166–177 https://doi.org/10.1137/1.9781611972733.15

Zihayat, M., Davoudi, H., & An, A. (2017). Mining significant high utility gene regulation sequential patterns. BMC systems biology, 11, 1-14. https://doi.org/10.1186/s12918-017-0475-4

Пратеће изјаве

ИЗЈАВА О АУТОРСТВУ

Име и презиме аутора _____Невена Миленковић_____

Број индекса _____5008/2017_____

Изјављујем да је докторска дисертације под насловом
“Развој алгоритама машинског учења за периодична унутар-секвенцијална правила и

динамичку корисност ставки“
_____

- резултат сопственог истраживачког рада;
- да дисертација у целини ни у деловима није била предложена за стицање друге дипломе према студијским програмима других високошколских установа
- да су резултати коректно наведени и
- да нисам кршио/ла ауторска права и користио/ла интелектуралну својину других лица.

У Београду, Потпис аутора

_____ _____

ИЗЈАВА О ИСТОВЕТНОСТИ ШТАМПАНЕ И ЕЛЕКТРОНСКЕ ВЕРЗИЈЕ ДОКТОРСКОГ РАДА

Име и презиме аутора _____ Невена Миленковић _____

Број индекса _____5008/2017_____

Студијски програм _____Информациони системи и квантитативни менаџмент___

Наслов рада _“Развој алгоритама машинског учења за периодична унутар-секвенцијална правила и динамичку корисност ставки“

Ментор _____Борис Делибашић_____

Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла ради похрањења у Дигиталном репозиторијуму Универзитета у Београду.

Дозвољавам да се објаве моји лични подаци везани за добијање академског назива доктора наука, као што су име и презиме, година и место рођења и датум објаве рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

У Београду, Потпис аутора

_____ _____

ИЗЈАВА О КОРИШЋЕЊУ

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

“Развој алгоритама машинског учења за периодична унутар-секвенцијална правила и динамичку корисност ставки“_____

која је моје ауторско дело.

Дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у Дигиталном репозиторијуму Универзитета у Београду и доступну у отвореном приступу могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио/ла.

1. Ауторство (CC BY)

2. Ауторство – некомерцијално (CC BY-NC)

3. Ауторство – некомерцијално – без прерада (CC BY-NC-ND)

4. Ауторство – некомерцијално – делити под истим условима (CC BY-NC-SA)

5. Ауторство – без прерада (CC BY-ND)

6. Ауторство – делити под истим условима (CC BY-SA)

(Молимо да заокружите само једну од шест понуђених лиценци. Кратак опис лиценци је саставни део ове изјаве).

У Београду, Потпис аутора

_____ _____

1. **Ауторство.** Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце, чак и у комерцијалне сврхе. Ово је најслободнија од свих лиценци.

2. **Ауторство – некомерцијално.** Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела.

3. **Ауторство – некомерцијално – без прерада.** Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела. У односу на све остале лиценце, овом лиценцом се ограничава највећи обим права коришћења дела.

4. **Ауторство – некомерцијално – делити под истим условима.** Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца не дозвољава комерцијалну употребу дела и прерада.

5. **Ауторство – без прерада**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца дозвољава комерцијалну употребу дела.

6. **Ауторство – делити под истим условима**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца дозвољава комерцијалну употребу дела и прерада. Слична је софтверским лиценцама, односно лиценцама отвореног кода.