شناسایی الگوهای مشابه مکرر بسته شده:

کاهش تعداد الگوهای مشابه مکرر بدون از دست دادن اطلاعات

استخراج الگوهای مکرر یک کار، کلیدی برای کشف اطلاعات مفید است. با وجود کیفیت راه حل های داده شده توسط الگوریتم های یافتن الگوهای مکرر ، اکثر آنها با چالش چگونگی کاهش تعداد الگوهای مکرر بدون از دست رفتن اطلاعات مواجه می شوند. یافتن مجموعه های مکرر این مشکل را با کشف یک مجموعه کاهش یافته از مجموعه مکرر، به نام مجموعه های مکرر بسته، که از آن می توان تمام مجموعه الگوی مکرر را بازیابی کرد، حل می کند. با این حال، برای یافتن نمونه های مکرر مشابه، که در آن تعداد الگوها حتی از مجموعه یابی مکرر بزرگتر هستند، این مشکل هنوز حل نشده است. در این مقاله، ما مفهوم یافتن الگوی مشابه مکرر بسته را برای کشف تعدادی از الگوهای مکرر مشابه بدون از دست دادن اطلاعات معرفی می کنیم. علاوه بر این،یک الگوریتم جدید یافتن الگوی مشابه مکرر بسته با نام CFSP-Miner پیشنهاد شده است. الگوریتم، الگوهای مکرر را با عبور از یک درخت که شامل تمام الگوهای مشابه شبیه بسته است، می یابد. برای انجام این کار به صورت موثر، چندین لم برای پر کردن فضای جستجو معرفی و اثبات شده است. نتایج نشان می دهد که CFSP-Miner کارآمدتر از الگوریتم های مورد استفاده یافتن الگوی مشابه مکرر است، مگر اینکه در مواردی که تعدادی از الگوهای مشابه مکرر و الگوهای مشابه مکرر بسته تقریبا برابر باشند. با این حال، CFSP-Miner قادر به پیدا کردن الگوهای مشابه بسته، تولید اندازه کاهش یافته از الگوی مشابه مکرر کشف شده بدون از دست دادن اطلاعات است. همچنین، CFSP-Miner در حین عملکرد قابل قبول در حین اجرا، مقیاس پذیری خوبی را نشان می دهد.

کلید واژه ها: داده کاوی. الگوهای مکرر. داده های ترکیبی. توابع شباهت. بسته شدن رو به پایین

1. مقدمه

یافتن الگوهای مکرر یک تکنیک است که شامل یافتن الگوهایی است (یعنی مجموعه های ویژگی با مقادیر مربوطه) که اغلب رخ می دهد (بیشتر یا برابر با حداقل آستانه تکرار) در یک مجموعه داده. این یک کار کلیدی در داده کاوی به دلیل استفاده از آن برای کشف اطلاعات مفید است مانند عوامل خطر (Li، Fu و Fahey، 2009؛ Li و همکاران، 2005؛ ناهار، امام، تلخه و چن، 2013) پروفایل های کاربر (Chiu، Yeh، Lee &، 2013)، رفتار انسان (Wen، Zhong، Wang &، 2015)، نرم افزارهای مخرب (Fan، Ye، Chen &، 2016) و غیره. علاوه بر این، یافتن الگوی مکرر را می توان به عنوان قدم قبلی یا داخلی برای سایر وظایف استخراج داده ها، مانند استخراج قواعد مرتبط (Alatas، Akin & Karci، 2008؛ Kalpana & Nadarajan، 2008؛ Lopez، Blanco، Garcia، Cano، & مارین، 2008)، طبقه بندی (هرناندز-لون، کاراسکو-اوچوا، مارتینز-ترینیداد و هرناندز-پالانکار، 2012؛ نگوین و نگوین، 2015) و خوشه بندی (Beil، Ester، Xu، 2002) استفاده کرد.

از سال 1990، اکثر الگوریتم های یافتن الگوی مکرر بر اساس تطابق دقیق ویژگی های بولین برای مقایسه و شمارش الگوها بود. این زیرمجموعه الگوریتم های یافتن الگوهای مکرر، یافتن اقلام مکرر نامیده می شود (به عنوان روش معمول برای یافتن الگوی مکرر). با این حال، اشیاء واقعی زندگی مانند اشیاء در جامعه شناسی (Ruiz-Shulcloper & Fuentes-Rodríguez، 1981)، زمین شناسی (گومز-هررا، رودریگز مورن، والاداراس آمرو و همکاران، 1994)، پزشکی (Ortiz-Posadas، Vega - Alvarado، Toni &، 2009) یا بازیابی اطلاعات (Baeza-Yates، Ribeiro-Neto و همکاران، 1999)) به ندرت مساوی هستند و یا با ویژگی های غیر بولین توصیف می شوند. به این ترتیب، توابع تشابه متفاوت از تطبیق دقیق برای مقایسه توصیف های شیء ارائه شده است که به ایجاد یک رویکرد جدید به نام یافتن الگوی مشابه مکرر می پردازد که می تواند مجموعه داده های حاوی ویژگی های غیر بولین را با استفاده از توابع شباهت کنترل کند(Danger، Ruiz-Shulcloper، Llavori &، 2004؛ رودریگز گونزالز،

مارتینز-ترینیداد، کاراسکو-اوچوا و رویز شولکلپر، 2008؛ 2011؛ 2013). این روش الگوهایی را ایجاد می کند که نمی تواند توسط آن الگوریتم ها بر اساس تطابق دقیق پیدا شود. الگوهای مکرر که با استفاده از یک تابع شباهت به دست می آیند، نامهای مرسوم مشابهی دارند (Carrasco-، Martínez Trinidad، Rodríguez-González، Ochoa، Ruiz-Shulcloper & 2013).

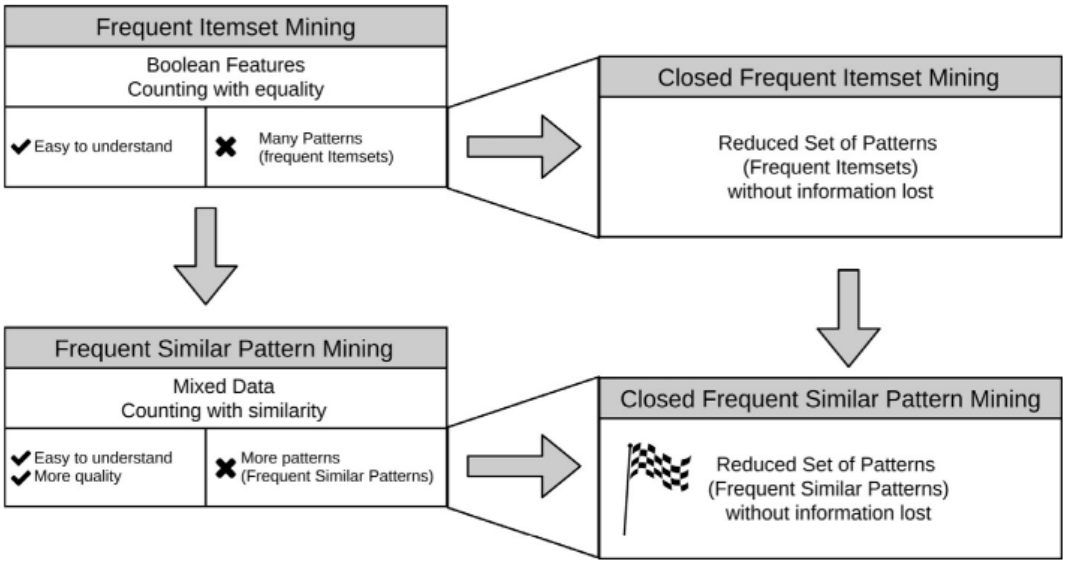
با وجود کیفیت راه حل های داده شده توسط یک الگوریتم یافتن اقلام مکرر و یا یک الگوریتم یافتن الگوهای مکرر مشابه، یک نقص مهم در هر دو روش این است که اگرچه یک مجموعه کامل از اقلام مکرر یا الگوهای مشابه مکرر می تواند یافت شود، تعداد الگوهای مکرر اغلب بیش از حد بزرگ است. (Burdick، Calimlim، و Gehrke، 2001؛ Hu، سونگ، Xiong و فو، 2008؛ Pei، هان، Mao و همکاران، 20 0 0؛ Rodríguez-González و همکاران، 2013؛ Zaki و Hsiao، 2002)

بنابراین، مفید است که مجموعه ای از تمام الگوهای مکرر را بدون از دست دادن اطلاعات بدست آورید (به عنوان مثال، از آنجا که کل مجموعه الگوی مکرر می تواند بازیابی شود). یکی از راه های انجام این کار، استفاده از یافتن اقلام مکرر بسته (Prabha، Shanmugapriya، Duraiswamy & 2013) است. الگوریتم های استخراج مکرر بسته تعریف می کنند که یک مجموعه اقلام مکرر بسته شده است، اگر هیچ الگوی فوق العاده با همان فرکانس ندارد و از این تعریف برای پیدا کردن مجموعه های مکرر بسته استفاده می شود. از چنین مجموعه های بسته، مجموعه کامل مجموعه های مکرر می تواند بدون از دست دادن اطلاعات تولید شود. الگوریتم های استخراج الگوهای مکرر به اصطلاح بسته نیز دارای کارایی کارآمدتری نسبت به الگوریتم های استخراج مکرر اقلام هستند (Pei et al.، 20 0 0؛ Uno، Asai، Uchida، Arimura & 2003؛ Zaki & Hsiao، 2002).

با این حال، مفهوم الگوهای بسته شده برای الگوهای مکرر مشابه به بهترین دانش ما مورد بهره برداری قرار نگرفته است. در این مقاله مفهوم یک الگوی مشابه مکرر مشابه و یک الگوریتم معکوس الگوریتم مشابه مکرر بسته به نام CFSP-Miner معرفی شده است که مجموعه ای از بسته های کاهش یافته از الگوهای مشابه مکرر را بدون از دست دادن اطلاعات پیدا می کند (به شکل زیر نگاه کنید محدوده کار ما) نتایج نشان می دهد که این الگوریتم پیشنهادی،

CFSP-Miner، Runtimes کارآمد تر از الگوریتم های یافتن الگوهای مشابه است که به طور مداوم در حال پیشرفت است، به جز زمانی که تعداد الگوهای مشابه مکرر و الگوهای شبیه بسته های مکرر تقریبا برابر است. از دیدگاه مقیاس پذیری، الگوریتم CFSP-Miner همچنین الگوهای شبیه نمونه های مکرر بسته را در یک زمان قابل قبول، بدون در نظر گرفتن اندازه، می یابد.

طرح کلی این مقاله به شرح زیر است: در بخش 2 کار مربوطه بررسی شده است. بخش 3 مفاهیم پایه را فراهم می کند. در بخش 4 مفاهیم متعددی معرفی و تعریف شده اند که نتیجه آن ترکیب مفهوم الگوی مشابه بسته و مفاهیم الگوی بسته است. در بخش 5 یک الگوریتم جدید برای یافتن الگوهای مشابه مکرر بسته پیشنهاد شده است. بخش 6 نتایج تجربی و بحث را ارائه می دهد و در نهایت در بخش 7 برخی نتایج و کارهای آینده مورد بحث قرار می گیرد.



شکل 1. از یافتن نمونه های مکرر تا یافتن الگوهای مشابه مکرر بسته

2. کارهای مرتبط

این مسئله استخراج الگوی مکرر موجب توجه جامعه پژوهشی داده کاوی به دلیل کاربرد بالقوه آن در بسیاری از حوزه های مختلف شده است. یک خط تحقیق بر کشف الگوهای مکرر در داده های مخلوط (مجموعه داده هایی که اشیاء آنها با ویژگی های عددی و غیر عددی توصیف می شوند) با استفاده از توابع شباهت برای مقایسه و شمارش اشیاء

متمرکز است (الگوهای مکرر مشابه) (Danger et al.، 2004؛ Rodríguez González et al . ، 2008؛ 2013). یک خط تحقیق دیگر متکی بر به دست آوردن یک مجموعه کاهش یافته از تمام الگوهای مکرر در مجموعه داده های بولی بدون از دست دادن اطلاعات است (مجموعه های مکرر بسته) (Prabha et al.، 2013؛ Uno et al.، 2003؛ Zaki & Hsiao، 2002). نتایج این خطوط تحقیق مربوط به کار فعلی در زیر بخش های زیر ارائه شده است.

2.1 استخراج الگوی مشابه مکرر

در پیشینه دو الگوریتم برای استخراج ذکر شده وجود دارد: ObjectMiner (Danger et al.، 2004) و STreeDCMiner (Rodríguez-González et al.، 2013). هر دو الگوریتم کل مجموعه ای از الگوهای مشابه مکرر را با استفاده از توابع شباهت بولین پیدا می کنند. مجموعه ای از الگوهای ردیابی معمولا شامل الگوهای پنهان به رویکرد سنتی است.

ObjectMiner (Danger et al.، 2004) اولین الگوریتمی برای استخراج الگوهای مشابه مکرر بود که از تابع شباهت متفاوت با برابری استفاده می کرد و از الگوریتم Apriori الهام گرفته شده بود (Agrawal et al.، 1994). ObjectMiner با پیروی از یک استراتژی جستجوی اولیه در عرض چند ثانیه کار می کند. با توجه به مجموعه داده D، یک تابع شباهت و حداقل آستانه فرکانس، ObjectMiner الگوهای مشابهی را در D فقط با یک ویژگی پیدا می کند. فرکانس الگوها با اضافه کردن رخداد خود و رخداد الگوهای مشابه آن محاسبه می شود. هر الگوی با فرکانس بیشتر از حداقل آستانه فرکانس، الگوی مشابهی را در نظر می گیرد. در تکرار k (شروع با k = 2) ObjectMiner الگوهای مکرر مشابهی را در D با ویژگی k پیدا می کند. این کار با ادغام الگوهای مشابه مکرر با ویژگی های k-1 انجام می شود. این فرآیند پس از آنکه هیچ الگوهای مکرر دیگری پیدا نشد، پایان می یابد.

ضعف اصلی ObjectMiner، این است که شباهت بین یک الگوی و تکرار آن در هر تکرار الگوریتم محاسبه می شود، که باعث محاسبه اضافی و غیر ضروری می شود. ObjectMiner همچنین مجموعه ای از تمام زیر توصیف های

مشابه (از جمله تکرار آن ها) را از هر یک از زیر فهرست های مکرر ذخیره می کند، که کارایی را کند می کند همانطور که در Rodríguez-González و همکاران نشان داده شده است. (2013).

StreeDC-Miner (Rodríguez-González، 2013(یک الگوریتم دیگر برای استخراج نمونه های مکرر است. STreeDC-Miner با پیروی از یک استراتژی جستجوی اولیه عمق ، با استفاده از یک ترتیب کامل بر روی ویژگی که در D تنظیم شده است، کار می کند. STreeDC-Miner شروع به تجزیه و تحلیل هر مجموعه با تنها یک ویژگی A می کند و یک ویژگی جدید به این مجموعه اضافه می کند که بعد از آخرین ویژگی در مجموعه به ترتیب مشخص شده است. مورد پایه در بازگشت که الگوهای مشابه مکرر بیشتری برای مجموعه فعلی ویژگی های تحت تجزیه و تحلیل کشف نشده است کامل می شود. به منظور افزایش کارایی فرکانس الگوهای محاسبه، STreeDC-Miner از ساختار درختی به نام STree استفاده می کند. هر برگ در STree تکرار الگوی تحت این شاخه را ذخیره می کند و همچنین شباهت بین این الگو و الگوهای مشابه آن را ذخیره می کند. شباهت بین دو الگو در STree تنها اگر زیرالگوها در STree شبیه باشند و اگر یکی از آنها یک الگوی مشابه مکرر است، محاسبه می شود. به این ترتیب STreeDC-Miner تلاش محاسباتی را برای محاسبه فرکانس هر الگو با کاهش ترسیم تابع شباهت کاهش می دهد. با این حال، STreeDC-Miner مانند ObjectMiner همان نقص یافتن بسیاری از الگوهای مکرر مشابه را دارد.

2.2 استخراج آیتم های مکرر بسته

استخراج آیتم های مکرر بسته مکرر تنها یافتن آن مجموعه های مکرر است که در آن مجموعه های دیگری وجود دارد که حاوی آن با همان فرکانس باشد (Prabha et al.، 2013). مزیت محاسبه مجموعه های مکرر بسته این است که آنها تمام اطلاعات مورد نیاز برای به دست آوردن تمام الگوهای مکرر و فرکانس دقیق آنها بدون نیاز به مجموعه داده های اصلی را حفظ می کنند. دو الگوریتم استخراج مشتق شده مکرر اول و معروف ترین Closet و CHARM است.

Closet (پی و همکاران، 20 0 0) بر اساس: i) فشرده سازی الگوهای مکرر در ساختار درختی حاوی الگوهای مکرر برای مجموعه های استخراج بسته شده بدون تولید کاندید، II) فشرده سازی یک مسیر تک درخت برای انجام شناسایی سریع الگوهای مکرر بسته، iii) اجرای مکانیزم پیش بینی مبتنی بر پارتیشن برای کاوش مقیاس پذیر در پایگاه داده های بزرگ.

Closet با استفاده از روش تقسیم و فتح برای استخراج الگوهای مکرر بسته استفاده می کند. اولا موارد مکرر در فرکانس نزولی یافت می شوند و مرتب می شوند. سپس فضای جستجو به زیر مجموعه های غیر همپوشانی تقسیم می شود و هر زیر مجموعه ای از مجموعه های مکرر بسته به صورت مجزا با ساختن پایگاه های داده ی مشروط مرتبط استخراج می شود.

CHARM از سوی دیگر، Zaki & Hsiao)، 2002(از روش پایین به بالا برای استخراج مجموعه های مکرر بسته استفاده می کند. این روش فضاهای مجموعه نمونه ها و جابجایی ها را از طریق یک درخت جستجوی مجموعه نمونه - مجموعه زمان دوبعدی، با استفاده از یک جستجوی هیبریدی کارآمد که در طول جستجو در سطوح مختلف، از بین میرود، بررسی میکند. CHARM همچنین با استفاده از یک تکنیک به نام diffsets برای کاهش حافظه از محاسبات متوسط استفاده می کند. در نهایت، از یک رویکرد مبتنی بر هش استفاده می کند تا هر مجموعه غیر بسته که در طی جستجو پیدا می شود حذف شود.

LCM(Uno et al، 2003 یکی دیگر از الگوریتم های استخراج معادلات مکرر بسته است. این تعریف یک رابطه پدر و کودک بین الگوهای بسته است. ثابت شد که هر رابطه والدین یک درخت را ایجاد می کند که از طریق آن می توان تمام الگوهای بسته را پیدا کرد. LCM همچنین یک روش کارآمد برای عبور از هر درخت در زمان چندجمله ای با توجه به مقدار مجموعه های مکرر بسته در مجموعه داده ها معرفی کرد.

از CHARM، Closet و LCM، دیگر الگوریتم ها نیز برای استخراج اقلام مکرر بسته شده از قبیل COBBLER (Pan، Tung، Cong، Xu & 2004)، TD-Close (Han & Shao، 2006)، PGMiner (Moonesinghe، Fodeh، TTD-Close (لیو و همکاران، 2009) CFIM-P (Nair)، TTD-Close (نجد و صدرالدینی، 2007)،

ICMiner (لی، وانگ، ونگ، چن و وو، 2008) & Tripathy، 2011)، DBV-Miner (Vo، Hong، & Le، 2012(، NAFCP (Le & Vo، 2015(و اخیرا BVCL (هاشم، کریم، ساملی اله و احمد، 2017) پیشنهاد شده است.

بر خلاف همه این الگوریتم های قبلی، کار ما یک الگوریتم را برای استخراج الگوهای مکرر بسته در مجموعه داده ها ارائه می دهد که در آن اشیا با ویژگی های عددی و غیر عددی توصیف می شوند.

3. مفاهیم و عبارات پایه

در این بخش، برخی از مفاهیم مربوط به استخراج الگوی مشابه مکرر و یافتن الگوی مکرر بسته معرفی شده است. در ابتدا، مفاهیم رایج توصیف شده اند. سپس، مفاهیم یافتن الگوی مشابه مکرر ذکر شده است. سوم ، مفاهیم یافتن الگوی مکرر بسته نیز بیان شده اند.

تعریف اول: (دامنه یک ویژگی). دامنه یک ویژگی دامنه کاربرد است: $A \longrightarrow 2^{\mathcal{V}}$ که به صورت زیر تعریف شده است:

$$Domain(A) = \{V \in \mathcal{V} \mid \exists O \in \mathcal{O} \ : \ V = O[A]\}.$$

تعریف دوم: (الگو). یک الگو در D یک جفت $T = (O, \mathcal{A}_T) \in (\mathcal{O} \times \{2^{\mathcal{A}} \backslash \{\}\})$ است. T.O علامت O و T.A نشان دهنده A_T است. همچنین T مجموعه ای از تمام الگوهای D را نشان می دهد. T یک مجموعه غیر خالی و محدود است. با توجه به دو الگو $T\ 1 \in T$ و $T\ 2 \in T$، اگر $A = T\ 2$ T1.A = T2.A و

$$\forall A \in T_1.\mathcal{A} \ ; \ T_1.O[A] = T_2.O[A].$$

تعریف 3: (الگوی فوق العاده). SupPatterns از یک الگو، SupPatterns کاربرد است: $T \longrightarrow 2^{T}$ که تعریف شده است:

$$SupPatterns(T) = \{T_{sup} \in \mathcal{T} \ \mid \ T.\mathcal{A} \subseteq T_{sup}.\mathcal{A} \quad \text{و} \quad T_{sup}.O[A] = T.O[A] \ \ \forall A \in T.\mathcal{A}\}$$

اگر $T_{sup} \in SupPatterns(T);$ سپس ما می گوییم که T sup یک الگویی فوق العاده از T است و T زیر الگوی T sup است.

تعریف 4 (ویژگی های با ترتیب خوب). ترتیب خوب می تواند در A تنظیم شود به این دلیل که یک مجموعه غیر خالی و محدود است. A≥ نشان دهنده یک نظم کامل است که ترتیب خوبی را تعریف می کند. اگر A 1 ≤ A A 2 و A 1 = A 2 باشد، A 1 <A A 2.

تعریف 5 (پیشوند). پیشوند یک الگو تا زمانی که یک ویژگی پیشوند کاربرد باشد: $(\mathcal{T} \times \mathcal{A}) \longrightarrow \mathcal{T}$ تعریف می شود:

$$Prefix(T, A) = T_{pr} \in \mathcal{T} \mid T_{pr}.O = T.O \quad \text{و} \quad \forall A_{pr} \in T_{pr}.\mathcal{A} \;\; A_{pr} \in T.\mathcal{A} \quad \text{و} \quad A_{pr} \leq_{\mathcal{A}} A$$

تعریف 6 (ویژگی قبلی) ویژگی قبلی از یک ویژگی کاربرد قبلی است: $A \longrightarrow (\mathcal{A} \cup \{A_0\})$ تعریف می شود:

$$Previous(A) = \begin{cases} A_0 & \text{if } \forall A' \in \mathcal{A}\ A \leq_{\mathcal{A}} A' \\ Previous_0(A) & \text{otherwise} \end{cases}$$

$Previous_0(A) =$ جایی که A0 یک ویژگی خاص است، به طوری که $A_0 \notin \mathcal{A}$.

$$A_{prev} \in \mathcal{A} \mid \forall A' \in \mathcal{A} \;\; A_m <_{\mathcal{A}} A \;\; \text{and } A' \leq_{\mathcal{A}} A_{prev}$$

مفاهیم مکرر معماری معادلات مشابه در زیر شرح داده شده است.

تعریف 8 (تابع شباهت بولین). تابع شباهت بولی در D یک برنامه کاربردی F است: $(T \times O) \rightarrow \{ True,False\}$ که $\forall T \in T$ ، $O \in O$ true = T ,O) = F (T ⇒ O = O .T .B مجموعه ای از تمام توابع شباهت بولی است که می توان در D تعریف کرد.

تعریف 9 (رخداد). رخداد یک الگو در D رخداد f برنامه است: $T \rightarrow 2^{O}$ که Occurrences F (T) = { O ∈ O | F (T , O) = T rue } .

تعریف 10 (فرکانس). فرکانس یک الگوی در D ، فرکانس F کاربرد است: $T \longrightarrow \{ 1 , 2 , . . . , O\}$ که FrequencyF(T) = Occurrences F (T) .

تعریف 11 (الگوی مشابه مکرر). الگوی $T \in T$ اگر فرکانس F(T) ≥M ، یک الگوی مشابه در D است که M حداقل آستانه است. M نشان دهنده دامنه M = { 1 , 2 , ... , O} همچنین، S مجموعه ای از همه الگوهای مشابه مکرر در D است.

با تعاریف فوق، یک مشکل یافتن الگوی مشابه مکرر را می توان به صورت زیر بیان کرد: با توجه به یک مجموعه داده $D = (O, A, V, P)$، $F \in B$ و $M \in M$، مسئله یافتن الگوی مشابه مکرر شامل یافتن مجموعه ای از تمام الگوهای مکرر مشابه S است.

تعریف 12 (تابع شباهت بولین مونوتونی غیر افزایشی). F یک تابع شباهت بولین مونوتونی غیر افزایشی است اگر

$$\forall O, T, T_{sup};\ O \in \mathcal{O}\ ;\ T \in \mathcal{T}\ ;\ T_{sup} \in SupPatterns(T)\ [F(T, O) = False]$$

$$N \Rightarrow [F(T_{sup}, O) = False]$$ نشان دهنده مجموعه ای از تمام توابع مشابه مشابه بولین یکسان است که می تواند در D تعریف شود.

نمونه 1. نمونه تابع شباهت بولین مونوتونی غیرواقعی: معادله

$$F_{eq}(T, O) = \begin{cases} True & \text{if } \forall A \in T.\mathcal{A}\ T.O[A] = O[A] \\ False & \text{otherwise} \end{cases}$$

نمونه 2. نمونه تابع شباهت بولین مونوتونی غیر افزایش: محصول

$$F^{\alpha}_{prod}(T, O) = \begin{cases} True & \text{if } \prod_{A \in T.\mathcal{A}} C_A(T.O[A], O[A]) \geq \alpha \\ False & \text{otherwise} \end{cases}$$

که $\forall A \in A \text{ as } C A : (\text{Domain } (A) \times \text{Domain } (A)) \longrightarrow [0, 1]$

لم 1 (یکنواختی فرکانس). $\forall T, T \text{ sup } ; T \in T ; T \text{ sup} \in \text{SupPatterns } (T) \text{ Frequency } F (T) \geq \text{Frequency } F (T \text{ sup})$

اثبات این لم یک نتیجه مستقیم از $F \in N$.

لم 2 (ویژگی بسته شدن پایین). $\forall T \in T \text{ if } T / \in S \text{ then } \forall T \text{ sup} \in \text{SupPatterns } (T) \ T \text{ sup} / \in S.$

اثبات این لم یک نتیجه مستقیم از یکنواختی فرکانس است.

در نهایت، دو مفهوم اصلی یافتن الگوی مکرر بسته (یعنی الگوی بسته و بسته شدن) با استفاده از نشانه های مشابه، رسمی می شوند.

تعریف 13 (الگوی بسته). با توجه به یک مجموعه داده $D = (O, A, V, P)$ و $F = Feq$، الگوی $Te \in T$ یک الگوی بسته در D است اگر $\forall T eSup \in SupPatterns(Te)$ ،فرکانس $F(TeSup) <$فرکانس $F(Te)$. Eeq مجموعه ای از تمام الگوهای بسته در D را با استفاده از $F = Feq$ نشان می دهد.

تعریف 14 (بسته شدن). با توجه به یک مجموعه داده $D = (O, A, V, P)$ و $F = F eq$ بسته شدن، بسته شدن کاربرد است: $T \rightarrow E eq$، به طوری که $Closure(T) = T cl \in E eq \mid T cl \in SupPatterns(T)$ $y\ F requency\ F(T) = F requency\ F(T cl)$.

4. ترکیب مفهوم الگوی مشابه مکرر و الگوی مشابه مکرر بسته

در این بخش، مفاهیم الگوی بسته برای یافتن مجموعه نمونه مکرر، برای یافتن الگوی مشابه مکرر گسترش می یابد.

تعریف 15 (الگوی مشابه بسته شده). با توجه به یک مجموعه داده $D = (O, A, V, P)$ و $F \in N$، یک الگوی مشابه بسته در D یک الگوی $T e \in T \mid \forall T eSup \in SupPatterns(T e)\ Frequency\ F(T eSup) < Frequency\ F(T e)$ است، جایی که E نشان دهنده مجموعه ای از تمام الگوی مشابه بسته در D با استفاده از F است.

تعریف 16 (بسته شدن). بسته شدن یک بسته شدن کاربرد است: $T \rightarrow E$، که $Closure\_(T) = T cl \in E \mid T cl \in SupPatterns(T)$ و $F requency\ F(T) = F requency\ F(T cl)$.

تعریف 17 (الگوی مشابه مکرر بسته). مسئله استخراجالگوی مشابه مکرر شامل پیدا کردن مجموعه ای از تمام الگوهای مشابه مکرر در $E \cap S$ است.

تعاریف 15 و 16، پسوند تعاریف هستند. 13 و 14 از بخش 3 و با تغییر $F = Feq$ به $F \in N$ مجاز به استفاده از هر تابع شباهت بولین مونوتونی غیر افزایشی به عنوان یک نتیجه تعریف 17 است.

توجه داشته باشید که اگر تعریف 16 در برخی از مجموعه داده ها با توابع مشابهی مورد استفاده قرار گیرد، پس از آن تصاویر برخی از الگوهای به صورت یکجانبه تعریف نمی شوند. دو نمونه زیر، تناقضاتی را با استفاده از تعریف 16 با توجه به یک مجموعه داده D و یک تابع شباهت بولین مونوتونی $F \in N$ نشان می دهند.

نمونه 3. با توجه به یک مجموعه داده $D = (O, A, V, P)$، به طوری که $O = \{ O 1 , O 2 \}$, $A = \{ X,$ $Y \}$, $V = \{ x, y 1 , y 2 \}$ و P به صورت زیر تعریف می شود:

$\mathcal{P}$	X	Y
O_1	x	y_1
O_2	x	y_2

الگوهای $T 1 = (O 1 , \{ X\})$، $T 2 = (O 1 , \{ X, Y \})$ ، و $T 3 = (O 2 , \{ X, Y \})$ را در نظر بگیرید. توجه داشته باشید که T 2 و T 3 فوق الگوی T1 هستند و هر دو الگوهای مشابه بسته هستند. بنابراین، T 2 و T 3 می توانند تصویر T 1 برای بستن برنامه باشد (از تعریف 16) که تناقضی از تعریف خود است.

نمونه 4. با توجه به یک مجموعه داد $D = (O, A, V, P)$، به طوری که $O = \{ O 1 , O 2 \}$، $A = \{ X,$ $Y, Z\}$، $V = \{ x, y 1 , y 2 , z 1 , z 2 \}$ و P به صورت زیر تعریف می شود:

$\mathcal{P}$	X	Y	Z
O_1	x	y_1	z_1
O_2	x	y_2	z_2

باید توجه داشت که $Domain (X) = \{ x \}$، $Domain (Y) = \{ y 1 , y 2 \}$ و $Domain (Z) = \{ z 1 ,$ $z 2 \}$. همچنین، با توجه به $F \in N$، $F = F_{prod}^{0.5}$ (در نمونه 2 تعریف شده) و معیارهای مقایسه زیر:

C_X	x
x	1

C_Y	y_1	y_2
y_1	1	0.5
y_2	0.5	1

C_Z	z_1	z_2
z_1	1	0.5
z_2	0.5	1

الگوهای $T1 = (O1, \{X\})$، $T2 = (O1, \{X, Y\})$، $T3 = (O2, \{X, Y\})$، $T4 = (O1, \{X, Z\})$، $T5 = (O2, \{X, Z\})$. توجه کنید که $F\text{requency}\,F(T1) = 2$, $F\text{requency}\,F(T2) = 2$, $F\text{requency}\,F(T3) = 2$, $F\text{requency}\,F(T4) = 2$، و $F\text{requency}\,F(T5) = 2$. همچنین، T2، T3، T4 و T5 فوق الگوی T1 هستند و الگوهای مشابه بسته هستند. بنابراین، T2، T3، T4 و T5 می تواند تصویر T1 برای بسته شدن برنامه باشد (تعریف 16) که در نتیجه، یک تضاد از تعریف خود است.

نمونه های فوق، تعریف بسته شدن جدید (تعریف 21) را برای الگوهای مکرر مشابه تطبیق می دهند. به این ترتیب، نرم افزار FCCLure به طور مشترک با برخی از لم های مرتبط به عنوان یک نتیجه معرفی شده است.

تعریف 18 (FClosure). FClosure برنامه FClosure است: $T \rightarrow 2^E$، به طوری که $F\,Closure(T) = \{Te \in E \mid Te \in SupPatterns(T), Frequency\,F(Te) = Frequency\,F(T)\}$.

به عنوان مثال، تعریف 18 در نمونه 4 اعمال می شود به عنوان یک نتیجه $F\,Closure(T1) = \{T2, T3, T4, T5\}$.

تعریف 19 (نظم خوب در دامنه ویژگی ها). با توجه به یک مجموعه داده $D = (O, A, V, P)$، $\forall A \in$ A یک مرتبه خوب بر روی دامنه (A) می تواند به دلیل مجموعه ای غیر خالی و محدود باشد. $A\geq$ نشان دهنده یک

نظم کامل است که تعریف خوبی را بر Domain (A) می دهد. اگر A v 1 ≤A A v 2 و $A_{v1} \neq A_{v2}$ سپس $A_{v1} <_A A_{v2}$.

تعریف 20 (L≥ رابطه). L≥ رابطه یک رابطه باینری در T است، با توجه به اینکه T 1 ≤ L T 2 Iff T 2 ∈ SupPatterns (T 1) یا $\exists A_{dif} \in T_1.\mathcal{A},$ ، که:

- $T_1.O[A_{dif}] <_{A_{dif}} T_2.O[A_{dif}]$.
- $\forall A_m \in T_2.\mathcal{A} \mid A_m <_A A_{dif} \;\; A_m \in T_1 \downarrow \mathcal{A}$ and $T_1.O[A_m] = T_2.O[A_m]$.

لم 3 (ترتیب لغوی). L≥ یک نظم خوب در T را تعریف می کند.

اثبات. $\forall A \in \mathcal{A}, \quad (\mathcal{A}, \leq_A)$ و (دامنه (A)، A ≤) مجموعه های خوبی هستند. بنابراین L≥ یک ترتیب لغوی در T را تعریف می کند (Weisstein، 2002).

نمونه 5 (نمونه ای از ترتیب لغوی). با توجه به مجموعه داده D و تابع شباهت F نمونه 4، A ≥، به طوری که X AY <AZ>، یک ترتیب کامل Y ≥، به طوری که Y1 ≤ Yy2 و یک نظم کل Z≥، به طوری که yz 1 ≤ Y z 2، ترتیب لغوی L≥ در T به ترتیب زیر است:

1. $(O_1, \{X\}) = x$
2. $(O_1, \{X, Y\}) = xy_1$
3. $(O_1, \{X, Y, Z\}) = xy_1z_1$
4. $(O_2, \{X, Y\}) = xy_2$
5. $(O_2, \{X, Y, Z\}) = xy_2z_2$
6. $(O_1, \{X, Z\}) = xz_1$
7. $(O_2, \{X, Z\}) = xz_2$
8. $(O_1, \{Y\}) = y_1$
9. $(O_1, \{Y, Z\}) = y_1z_1$
10. $(O_2, \{Y\}) = y_2$
11. $(O_2, \{Y, Z\}) = y_2z_2$
12. $(O_1, \{Z\}) = z_1$
13. $(O_2, \{Z\}) = z_2$

در حال حاضر، تعریف جدید بستن، به نام بستن لغوی، معرفی شده است:

تعریف 21 (بستن لغوی برای تابع شباهت بولانی غیر همزمان). بستن لغوی یک الگو، کاربرد $FCL : T \rightarrow E,$ است، به طوری که $FCL(T) = T_{fcl} \in \mathcal{E} \mid T_{fcl} \in FClosure(T),$ ، T fcl حداقل الگو در FClosure (T) است.

نمونه 6. نمونه ای از بسته شدن لغوی. با استفاده از نمونه 5 و استفاده از تعریف 21:

- $FCL(O_1, \{X\})) = (O_1, \{X, Y\})$
- $FCL(O_1, \{X, Y\}) = (O_2, \{X, Y, Z\})$

بستن لغوی FCL یک برنامه کاربردی است که برای همه دامنه آن به دلیل لم 5 تعریف شده است، که اثبات آن بر اساس لم زیر است:

لم 4 (FClosure هرگز خالی نیست). با توجه به مجموعه داده $\mathcal{D} = (\mathcal{O}, \mathcal{A}, \mathcal{V}, \mathcal{P})$ و

$$F \in \mathcal{N}, \quad \forall T \in \mathcal{T} \ \|FClosure(T)\| \geq 1.$$

.

اثبات. $\forall T \in \mathcal{T} \quad T \in \mathcal{E} \ \text{ or } T \notin \mathcal{E}.$

اگر $T \in \mathcal{E}$:

-1 $T \in FClosure\ (T)$ چون $T \in E$ و $T \in SupPatterns\ (T)$.

-2 $FClosure(T) \geq 1$

اگر $T \notin \mathcal{E}$:

-1 $H(T) \subseteq \mathcal{T}$ را طبق زیر بسازید:

$$H(T) = \{H \in SupPatterns(T) \mid Frequency_F(H) = Frequency_F(T)\}.$$

-2 H(T) ناتهی است زیرا $T \notin \mathcal{E}$. همچنین یک مجموعه متناهی است زیرا $H(T) \subseteq \mathcal{T}$.

-3 رابطه $H_{tam} \subseteq (H(T) \times H(T))$ به صورت زیر تعریف می شود:

$$H_{tam} = \{(H_1, H_2) \in (H(T) \times H(T)) \mid \|H_1.\mathcal{A}\| \geq \|H_2.\mathcal{A}\|\}.$$

Htam یک ترتیب کامل در H(T) است.

4- مجموع ترتیب کامل ((H (T)، H tam) یک مجموعه مرتب است، زیرا H (T) یک مجموعه غیر خالی و محدود است و H tam یک ترتیب کامل است. در نتیجه، یک عنصر حداقل در H (T) وجود دارد، یعنی

$$\exists H_m \in H(T) \mid \forall H \in H(T) \ \|H_m . A\| \geq \|H . A\|$$

5- $H_m \in \mathcal{E}$

6- $H_m \in FClosure(T)$

7- $FClosure(T) \geq 1$

لم 5 (FCL به خوبی تعریف شده است). برای $\forall T \in \mathcal{T}$ FCL دقیقا تعریف شده است.

اثبات.

1- $\forall T \in \mathcal{T}$ ، $FClosure(T)$ غیرتهی است.

2- $\forall T \in \mathcal{T}$ ، $FClosure(T) \subseteq \mathcal{E} \subseteq \mathcal{T}$ و T با استفاده از $\leq_L$ به خوبی منظم شده است.

3- $\forall T \in \mathcal{T}$ ، $FCL(T)$ با استفاده از 1 و 2 به خوبی تعریف شده و چون برای همه مجموعه های با ترتیب خوب و غیرتهی یک حداقل المانی وجود دارد.

5. الگوریتم الگوریتم مشابه (CFSP-Miner)

در این بخش، الگوریتم CFSP-Miner برای استخراج الگوهای مشابه مکرر را به وجود می آوریم که در آن تابع شباهت ویژگی بسته شدن از پایین را از لم 2 نگه میدارد. الگوریتم با عبور از یک درخت عمل می کند، که توسط یک رابطه پدر و کودک تعریف شده است که شامل تمام الگوهای مشابه بسته است. قبل از اینکه تعریف رابطه پدر و فرزند را معرفی کنیم، باید تعریف جدیدی را ارائه دهیم:

تعریف 22 (پیشوند کوچک با FCL برابر). پیشوند جزئی با FCL برابر یک الگوی مشابه بسته، کاربرد PRFCL: $E \rightarrow A$ است، به طوری که:

$$PrFCL(T) = A_{pr} \in T.\mathcal{A} \mid FCL(Prefix(T, A_{pr})) = T$$ و

$$\forall A_m \in T.\mathcal{A} \mid A_m <_{\mathcal{A}} A_{pr}) \quad FCL(Prefix(T, A_m)) \neq T$$

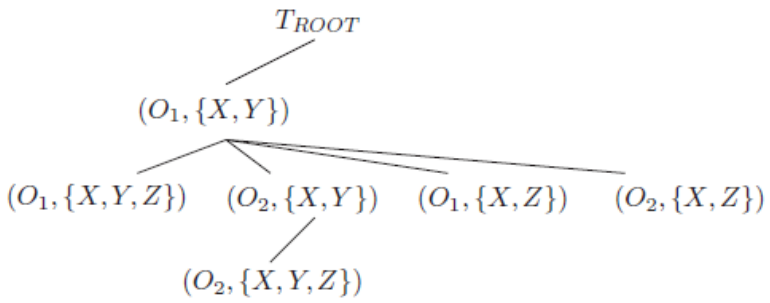
تعریف 23 (رابطه پدر و کودک در میان الگوهای مشابه بسته). پدر الگوی مشابه بسته، کاربرد پدر است: E→EU TROOT، به طوری که:

$$Father(T_e) = \begin{cases} FCL(Prefix(T_e, Previous(PrFCL(T_e)))) & if\ Previous(PrFCL(T_e)) = A_0 \\ T_{ROOT} & otherwise \end{cases} \quad (4)$$

که T ROOT یک الگوی خاص است، به طوری که $T_{ROOT} \notin \mathcal{T}$. توجه داشته باشید که پدر برای تمام دامنه آن تعریف شده است.

نمونه 7. نمونه از رابطه پدر و فرزند. با استفاده از تعریف 23 در نمونه 5 می توان دید که:

- $Father((O_1, \{X, Y, Z\})) = (O_1, \{X, Y\})$
- $Father((O_2, \{X, Y\})) = (O_1, \{X, Y\})$
- $Father((O_1, \{X, Y\})) = T_{ROOT}$
- $Father((O_2, \{X, Y, Z\})) = (O_2, \{X, Y\})$



نمونه ای از درخت رابطه پدر و فرزند

تعریف 23 (روابط پدر و فرزندان) می تواند برای ساخت یک گراف که در واقع درخت است استفاده شود.

تعریف 24 (نمودار ارتباط پدر و فرزند). G یک گراف غیرقابل هدایت است، به طوری که $G = (V, A)$ که در آن

$$A = \{(T_1, T_2) \in (V \times V) \mid Father(T_2) = T_1\}$$ و $$V = \mathcal{E} \cup \{T_{ROOT}\}$$

لم 6 (G یک درخت است). نمودار رابطه پدر و فرز G یک درخت است.

اثبات این لم بر اساس لم 7 است.

اثبات.

1- G به وسیله لم 7 غیرچرخان است است.

2- $\|A\| = \|V\| - 1$ زیرا پدر یک کاربرد است که برای $\forall T \in \mathcal{E}$ به غیر از T_{ROOT} تعریف می شود.

3- G با 1 و 2 یک درخت است.

به عنوان مثال، شکل 2، درخت نسبت پدر و فرزند از نمونه 5 را نشان می دهد.

لم 7 (PrFCL از پدر به کودک رشد می کند). $PrFCL(T_{father}) <_{\mathcal{A}}$

$$PrFCL(T)\ \forall T \in \mathcal{E} \mid T_{father} = Father(T) \neq T_{ROOT}$$

اثبات.

$$T_{father} = FCL(Prefix(T, Previous(PrFCL(T))))$$
$$T_{father} = Pather(T)\ .$$

1-

2- $PrFCL(T_{father}) \leq_{\mathcal{A}} Previous(PrFCL(T))$

3- $PrFCL(T_{father}) <_{\mathcal{A}} PrFCL(T)$

لم 8 (محدودیت PrFCL پایین). $\forall T \in \mathcal{T}\ \ Tail(T) \leq_{\mathcal{A}} PrFCL(T_{fcl})$ و $T_{fcl} = FCL(T)$

اثبات.

1- $Intermediate(T) \subseteq \mathcal{T}$ را مطابق زیر بسازید:

$$Intermediate(T) = \{Z \in SupPatterns(T)\ y\ T_{fcl} \in SupPatterns(Z)\}$$

2- $\forall Z \in SupPatterns(T),\ FCL(Z) = T_{fcl}$

3- $\forall A_{int} \in \mathcal{A} \mid Tail(T) \leq_{\mathcal{A}} A_{int} \leq_{\mathcal{A}} Tail(T_{fcl}),\ Prefix(T_{fcl}, A_{int}) \in Intermediate(T).$

4- از 2 و 3 : $FCL(Prefix(T_{fcl}, A_{int})) = T_{fcl}$ ، $\forall A_{int} \in \mathcal{A} \mid Tail(T) \leq_{\mathcal{A}} A_{int} \leq_{\mathcal{A}} Tail(T_{fcl})$

5- از 4 $Tail(T) \leq_{\mathcal{A}} PrFCL(T_{fcl})$.

با داشتن درختی که حاوی تمام الگوهای مشابه است، یک درخت عبور از ریشه می تواند تعریف شود تا همه آنها را پیدا کند. این را می توان با استفاده از یک کاربرد کودکان زیر تعریف کرد:

تعریف 25 (کودکان). کودکان یک الگوی مشابه بسته، کاربرد کودکان است: $\mathcal{E} \cup T_{ROOT} \longrightarrow 2^{\mathcal{E}}$، به طوری که:

$$Children(T) = \{T_{child} \in \mathcal{E} \mid Father(T_{child}) = T\}.$$

برای پیدا کردن کودکان یک الگوی مشابه بسته، از لم 9 استفاده خواهیم کرد. قبل از معرفی آن، دو تعریف زیر لازم است:

تعریف 26 (گسترش یک الگو). گسترش یک الگو کاربرد گسترش: $\mathcal{T} \longrightarrow 2^{\mathcal{E}}$ است، با توجه به آن:

$$Extensions(T) = \left\{ T_{ext} \in \mathcal{E} \mid T_{ext} \neq FCL(T) \quad and \quad \begin{array}{l} \exists T_{+1} \in \mathcal{T},\ suchthat: \\ T = Prefix(T_{+1}, Prevoius(Tail(T_{+1}))) \\ T_{ext} = FCL(T_{+1}) \\ T_{+1} = Prefix(T_{ext}, Tail(T_{+1}) \end{array} \right\}$$

به عنوان مثال، با استفاده از تعریف 26 در نمونه 5، می توان دید که:

- $(O_2, \{X, Y\}) \in Extensions((O_1, \{X\}))$, in this case $T_{+1} = (O_2, \{X, Y\})$
- $(O_2, \{X, Y, Z\}) \in Extensions((O_2, \{X, Y\}))$, in this case $T_{+1} = (O_2, \{X, Y, Z\})$

تعریف 27 (معکوس FCL). معکوس FCL یک الگوی مشابه بسته، کاربرد IFCL است: $\mathcal{E} \longrightarrow 2^{\mathcal{E}}$ بطوری که:

$$IFCL(T) = \{T_{ifcl} \in \mathcal{E} \mid FCL(T_{ifcl}) = T\}$$

لم 9 (پیدا کردن کودکان). $\forall T \in \mathcal{E}$ و $\forall T_{child} \in \mathcal{E}$، اگر $T_{child} \in Children(T)$ که $T_{child} \in Extensions(T_{ifcl})$

$T_{child} \in Children(T)$.

1- $T_{ifcl} = Prefix(T_{child}, Previous(PrFCL(T_{child})))$ و $T_{child} \in Extensions(T_{ifcl})$

2- $FCL(T_{ifcl}) = T$ زیرا $T_{ifcl} \in IFCL(T)$.

3- $Father(T_{child}) = T$

4- $T_{child} \in Children(T)$

در حال حاضر ثابت شده است که اگر $T_{child} \in Children(T)$ برای برخی $T_{ifcl} \in IFCL(T)$ سپس $T_{child} \in Extensions(T_{ifcl})$ است.

اثبات. با این اثبات شروع می شود که اگر ((T ext ∈ Extensions (T) سپس T = P re f ix (T ext , P re v ious (P rF CL (T ext)))

1- $T = Prefix(T_{ext}, Previous(PrFCL(T_{ext})))$

2- اگر $T_{+1} = (T.O, T.A \cup PrFCL(T_{ext}))$ یک الگو باشد، توجه کنید که $T = Prefix(T_{+1}, Previous(Tail(T_{+1})))$ و $T_{+1} = Prefix(T_{ext}, Tail(T_{+1}))$

3- $T_{ext} = FCL(T_{+1})$

4- $T_{ext} \neq FCL(T)$

5- $T_{ext} \in Extensions(T)$

اکنون این اثبات می شود که اگر $T_{ext} \in Extensions(T)$ سپس $T = Prefix(T_{ext}, Previous(PrFCL(T_{ext})))$.

1. $T_{ext} \in Extensions(T)$ implies that $\exists T_{+1} \in \mathcal{T}$
2. $T_{ext} = FCL(T_{+1})$
3. $T_{ext} \neq FCL(T)$ because $T_{ext} \in Extensions(T)$.
4. $PrFCL(T_{ext}) = Tail(T_{+1})$ by 2 and 3.

5. $Prefix(T_{ext}, Previous(PrFCL(T_{ext}))) = T$
   $Previous(Tail(T_{+1})) = Tail(T)$.

لم 11 (تصویر یک الگوی مشابه بسته خود است). $FCL(T) = T \ \forall T \in \mathcal{E}$

اثبات.

1- $\forall T \in \mathcal{E} \quad T \in FClosure(T).$

2- $\forall T \in \mathcal{E} \quad s \nexists T_x \in FClosure(T) \mid T_x \neq T$ زیرا اگر یک Tx وجود داشته باشد سپس این حقیقت را اثبات می کند که $T \in \mathcal{E}$.

3- $\forall T \in \mathcal{E}\ FCL(T) = T$

لم 12 (IFCL هرگز تهی نیست) $\|IFCL(T)\| \geq 1\ \forall T \in \mathcal{E}.$

اثبات کافی است که آن را در نظر داشته باشید که $FCL(T) = T\ \forall T \in \mathcal{E}$

ابتدای CFSP-Miner (الگوریتم 1) از مجموعه الگوهای مشابه مکرر بسته، از طریق الگوی خاص T ROOT با استفاده از لم 9، که برای یافتن بچه های مجموعه یک الگوی مشابه بسته استفاده می شود، عبور می کند.

Algorithm 1: CFSP-Miner $_{Baseline}(\mathcal{D}, F, M, T\ )$.

Input: Dataset $\mathcal{D} = (\mathcal{O}, \mathcal{A}, \mathcal{V}, \mathcal{P})$,
Similarity Function $F \in \mathcal{N}$,
Minimum Frequency Threshold $M \in \mathcal{M}$
Frequent Closed Similar Pattern $T \in \mathcal{S} \cap \mathcal{E}_{eq}$
Output: Frequent Closed Similar Patterns Set $\mathcal{SE}_{eq}$
foreach $T_{ifcl} \in IFCL(T)$ **do**
  foreach $T_{child} \in Extensions(T_{ifcl})$ **do**
    if $Frequency_F(T_{child}) \geq M$ **then**
      $\mathcal{SE}_{eq} \leftarrow$
      $\mathcal{SE}_{eq} \cup T_{child} \cup$ CFSP-Miner$_{Baseline}(\mathcal{D}, F, M, T_{child})$

بیایید یک مثال از اجرای الگوریتم بازخورد CFSP-Miner Baseline را برای عملکرد داده ها و شباهت های نمونه 4 و آستانه فرکانس حداقل M = 2 ببینیم. نمونه ای از درخت ارتباط پدر و کودک (شکل 2) به ما کمک خواهد کرد. از آنجا که الگوریتم بازگشتی است، با سادگی، ما تنها با توضیح تنها یک تکرار تمرکز خواهیم کرد. تکرار انتخاب شده به عنوان ورودی F requency F (T) = 2 , T = (O 1 , { X, Y }) دریافت می کند.

اول، تمام عناصر متعلق به IFCL (T) یافت می شوند. به یاد داشته باشید IFCL (T) مجموعه ای از تمام عناصر است که T به عنوان بستن لغوی استفاده می شود. سپس IF CL (T) = { (O 1 , { X, Y }) , (O 1 , { X}) , (O 1 , { Y }) } چون:

Tهمواره به IFCL(T) تعلق دارد.

$$FCL((O_1, \{X\})) = T \text{ and } Frequency_F((O_1, \{X\})) = 2 .$$
$$FCL((O_1, \{Y\})) = T \text{ and } Frequency_F((O_1, \{Y\})) = 2 .$$

برای هر عنصر در IFCL (T)، ضمیمه آن جستجو می شود. ضمیمه یک الگوی T مجموعه ای است که شامل بسته شدن لغوی هر عنصر به دست آمده با اضافه کردن یک ویژگی به T بعد از آخرین ویژگی آن است.

$Extensions((O_1, \{X, Y\})) = \{FCL((O_1, \{X, Y, Z\}))\}$ زیرا در نمونه 4 تنها ویژگی دیگری که وجود دارد، ویژگی Z است و $\{FCL((O_1, \{X, Y, Z\}))\} = \{(O_1, \{X, Y, Z\})\}$ زیرا هیچ الگوئی وجود ندارد که شامل آن باشد. سپس Extensions ((O 1 , { X, Y })) = {(O 1 , { X, Y, Z })}.

Extensions ((O 1 , { X})) = { F CL ((O 1 , { X, Z})) , F CL ((O 2 , { X, Y })) , F CL ((O 2 , { X, Z})) }

زیرا

- اولین راه اضافه کردن ویژگی به $(O_1, \{X\})$، $(O_1, \{X, Z\})$ است.
- راه دیگری برای اضافه کردن ویژگی به $(O_1, \{X\})$. ، $(O_2, \{X, Y\})$ است. توجه کنید که $O_1[X] = O_2[X]$
- $(O_2, \{X, Z\})$ همانند موارد قبلی است

همچنین F CL ((O 1 , { X, Z})) = (O 1 , { X, Z}) چون فرکانس آن 2 است و هیچ الگوی وجود ندارد که حاوی آن باشد. به همین دلیل F CL ((O 2 , { X, Y })) = (O 2 , { X, Y }) و F CL ((O 2 , { X, Z})) = (O 2 , { X, Z}) سپس Extensions ((O 1 , { X})) = { (O 1 , { X, Z}) , (O 2 , { X, Y }) , (O 2 , { X, Z}) }.

((O 1 , { Y })) Extensions تهی است زیرا هیچ $T_eXt \neq FCL(T),$ مانند $T_eXt = FCL(T')$ و $T' = Prefix(T_{ext}, Tail(T')$ وجود ندارد.

کودکان ({ X, Y } ، (O 1), (O 2 , { X, Y }), (O 2 , { X, Z }), (O 1 , { X, Y, Z }) { X, Z }). کودکان با فرکانس بزرگتر یا مساوی 2 به الگوهای مکرر بسته مشابه اضافه می شوند و هر کدام ورودی یک تکرار دوباره بازگشتی است.

مهمترین گام CFSPs-Miner شامل یافتن مجموعه IFCL یک الگوی مشابه بسته است. برای انجام این مرحله به طور موثر، یک ویژگی ساده شده (لم 13) و شرایط لازم و کافی (لم 14 و 16) برای یک الگوی متعلق به IFCL معرفی می شود.

لم 13 (ساده کردن جستجو در مجموعه IFCL). اگر $\overline{FCL(T_{sup}) \neq T_e,}$ سپس

$$\forall T_e \in \dot{\mathcal{E}} \mid T_e \in SupPatterns(T_{sup}) \quad FCL(T) \neq T_e \quad \forall T \in \mathcal{T}, \quad \forall T_{sup} \in SupPatterns(T),$$

اثبات.

دو مورد وجود دارد: $Frequency_F(T_{sup}) \neq Frequency_F(T_e)$ یا $\overline{Frequency_F(T_{sup}) = Frequency_F(T_e)}$

اگر $Frequency_F(T_{sup}) \neq Frequency_F(T_e)$:

1. $Frequency_F(T) \geq Frequency_F(T_{sup})$ by Lemma 1.
2. $Frequency_F(T_{sup}) > Frequency_F(T_e)$ by Lemma 1.
3. $Frequency_F(T) > Frequency_F(T_e)$ by 1 and 2.
4. $FCL(T) \neq T_e$ by 3.

اگر $Frequency_F(T_{sup}) = Frequency_F(T_e)$:

دو زیر شاخه وجود دارد: $Frequency_F(T_{sup}) < Frequency_F(T)$ یا $\overline{Frequency_F(T_{sup}) = Frequency_F(T)}$

اگر $Frequency_F(T_{sup}) < Frequency_F(T)$ باید توجه داشت که $Frequency_F(T) \neq Frequency_F(T_e).$

اگر $Frequency_F(T_{sup}) = Frequency_F(T)$:

1- $T_e \in FClosure(T_{sup})$ با تعریف 18

2- $FCL(T_{sup}) <_L T_e$ با تعریف 18 به دلیل $T_e \neq FCL(T_{sup})$.

3- $T_e \in FClosure(T)$ و $FCL(T_{sup}) \in FClosure(T)$ توسط تعریف 18

4- $FCL(T) \leq_L FCL(T_{sup}) <_L T_e$ نتیجه 3 و 2

5- $FCL(T) \neq T_e$ نتیجه 5

لم 14 (شرط لازم برای تعلق به IFCL). اگر $Frequency_F(T) \neq Frequency_F(T_e)$ یا $Tail(T) <_{\mathcal{A}} PrFCL(T_e)$.

پس $FCL(T) \neq T_e \ \forall T \in \mathcal{T}, \ \forall T_e \in \mathcal{E} \mid T_e \in SupPatterns(T)$.

اثبات. برای مورد $Frequency_F(T) \neq Frequency_F(T_e)$، ضروری است که توجه کنیم تعریف FCL به برابری $Frequency_F$ نیاز دارد.

برای مورد $Tail(T) <_{\mathcal{A}} PrFCL(T_e)$:

1- $Prefix(T_e, \quad Previous(PrFCL(T_e))) \in SupPatterns(T)$ زیرا

$T_e \in SupPatterns(T)$ and $Tail(T) <_{\mathcal{A}} PrFCL(T_e)$

2- $FCL(Prefix(T_e, Previous(PrFCL(T_e)))) \neq T_e$ با تعریف 22

3- $FCL(T) \neq T_e$ با 1و 2 و لم 14.

برای معرفی شرایط کافی برای الگوی متعلق به IFCL، تعاریف 28 و 29 و لم 15 در زیر آمده است.

تعریف 28 (الگوهای فوق العاده با یک ویژگی دیگر و فرکانس برابر). الگوهای فوق العاده با یک ویژگی دیگر و فرکانس یکسان الگوی $SupPatterns^{=}_{+1} : \mathcal{T} \longrightarrow 2^{\mathcal{T}}$ است، به طوری که:

$$SupPatterns^{=}_{+1}(T) = \{T^{=}_{+1} \in SupPatterns(T) \mid \|T^{=}_{+1}.\mathcal{A}\| = \|T.\mathcal{A}\| + 1 \text{ and } Frequency_F(T^{=}_{+1}) = Frequency_F(T)\}$$

تعریف 29 (استخراج یک الگو). تشخیص یک الگو کاربردِ Derivation است: $\mathcal{T} \longrightarrow \mathcal{T}$ ، به طوری که:

$$Derivation(T) = \begin{cases} \min\limits_{T' \in SupPatterns^{=}_{+1}(T)} T' & if \ \|SupPatterns^{=}_{+1}(T)\| > 0 \\ T & otherwise \end{cases}$$

توجه داشته باشید که Derivation برای تمام دامنه آن تعریف شده است.

لم 15 (ویژگی استخراج). $\forall T \in \mathcal{T} \ \ FCL(T) = FCL(Derivation(T))$

اثبات. ابتدا $FCL(Derivation(T)) \leq_L FCL(T)$ ثابت می شود:

1- $FCL(Derivation(T)) \in SupPatterns(T)$ زیرا $FCL(Derivation(T)) \in SupPatterns(Derivation(T))$ و $Derivation(T) \in SupPatterns(T)$.

2- $Frequency_F(FCL(Derivation(T))) = Frequency_F(Derivation(T)) = Frequency_F(T)$

3- $FCL(Derivation(T)) \in FClosure(T)$

4- $FCL(Derivation(T)) \leq_L FCL(T)$

اکنون $FCL(T) \leq_L FCL(Derivation(T))$ اثبات می شود:

1- $FCL(T) \in SupPatterns(Derivation(T))$ از تعریف 21 و 29

2- $Frequency_F(T) = Frequency_F(Derivation(T)) = Frequency_F(FCL(Derivation(T)))$

3- $FCL(T) \in FClosure(Derivation(T))$ از 1 و 2

4- $FCL(T) \leq_L FCL(Derivation(T))$ از 3 و تعریف 21

پس از آن $FCL(T) = FCL(Derivation(T))$ زیرا $\leq_L$ یک ترتیب کامل است.

لم 16 (شرایط کافی برای متعلق به IFCL). اگر $\overline{FCL(Derivation(T))} = T_e$، سپس $\overline{FCL(T)} = \overline{T_e} \ \ \forall T \in \mathcal{T} \ \ and \ \forall T_e \in \mathcal{E}$.

اثبات این لم یک نتیجه مستقیم از لم 15 است.

در این مرحله، می توان $IFCL(T_e)\ \forall T_e \in \mathcal{E}$ را به صورت کارآمد ، با تمام ضوابط T را با استفاده از لم 13 و 14 برای ساده کردن فضای جستجو، و لم 16 برای بررسی اینکه آیا یک زیرالگو متعلق به IFCL (T e) است، بدست آورد.

برای رسیدن به پیاده سازی کارآمد CFSP-Miner (الگوریتم 2)، مجموعه IFCL و Extensions به طور همزمان ساخته می شوند. این کار با گسترش الگوها، به ترتیب لغوی، با اضافه کردن ویژگی های جدید و ویژگی های آنها انجام می شود.

هنگامی که الگوی جدید گسترش یافته برای اولین بار فرکانس مشابه با نسخه غیر گسترش داده شده خود را دارد، زیرگروه های آن، از جمله ویژگی tail و با FCL برابر، نیز گسترش یافته است. هر زیرمجموعه می تواند برای به دست آوردن بچه های جدید استفاده شود.

قبل از گسترش یک الگو و برای جلوگیری از تکرار همان تحلیل، باید تأیید شود که هیچ الگوی فوق العاده ای کمتر (با استفاده از L ≥) از آن با فرکانس مشابه وجود ندارد.

برای تعریف L≥ ضروری است که یک نظم خوب در A از توانایی دامنه های ویژگی ایجاد شود. با این حال، برای تعریف $\forall A \in \mathcal{A}\ \leq_{Domain(A)}$ برای ویژگی های عددی، ترتیب سنتی تعریف شده توسط ≥ استفاده می شود، در حالیکه برای ویژگی های غیر عددی، ترتیبی که در آن، مقدار ها در مجموعه داده ها ظاهر می شود، استفاده می شود.

بیایید یک نمونه از اجرای الگوریتم CFSP-Miner را برای مجموعه داده ها و عملکرد شباهت نمونه 4 و حداقل آستانه فرکانس M = 2 ببینیم. CFSP-Miner از الگوی ویژه T ROOT شروع می شود.

هر Texp با گسترش T ROOT با یک مقدار مشخص شده است. سپس T exp روی {O1}، ({X}، O1)، ({Y}، O1)، ({Z}، O1)، ({Y}، O2)، ({Z}، O2) } تکرار می شود. برای هر T exp، F requency F (T exp) = 2.

پرچم DerivationFound = T rue به معنای آن است که دارای یک استخراج است: یک الگو با فرکانس برابر و یک فیلد دیگر کهکوچکترین آن است که در ترتیب لغوی گسترش یافت. به عبارت دیگر، DerivationFound = T rue مستلزم آن است که بسته نباشد، اما مسیر که در FCL این الگو ظاهر می شود یافت شده است.

Algorithm 2: CFSP-Miner$(\mathcal{D}, F, M, T)$.

Input: Dataset $\mathcal{D} = (\mathcal{O}, \mathcal{A}, \mathcal{V}, \mathcal{P})$,
Similarity Function $F \in \mathcal{N}$,
Minimum Frequency Threshold $M \in \mathcal{M}$
Frequent Closed Similar Pattern $T \in \mathcal{S} \cap \mathcal{E}_{eq}$
Output: Frequent Closed Similar Patterns Set $\mathcal{SE}_{eq}$

```
if ∃T_men ∈ SupPatterns^{=}_{+1}(T) | T_men <_L T then
    return
DerivationFound ← False
foreach T_exp ∈ SupPatterns(T) such that
‖T_exp.A‖ = ‖T.A‖ + 1 , Tail(T) <_A Tail(T_exp) do
    if ¬DerivationFound and
    Frequency_F(T_exp) = Frequency_F(T) then
        DerivationFound ← True
        foreach T_equal−fcl ∈ T such that T_exp ∈
        SupPatterns(T_equal−fcl), Tail(T_equal−fcl) = Tail(T_exp),
        FCL(T_equal−fcl) = FCL(T_exp) do
            SE_eq ← SE_eq ∪ CFSP-Miner(D, F, M, T_equal−fcl)
    else
        SE_eq ← SE_eq ∪ CFSP-Miner(D, F, M, T_exp)
if ¬DerivationFound then
    SE_eq ← SE_eq ∪ T
```

برای هر T exp، فرکانس T exp متفاوت از فرکانس T ROOT است. سپس یک فراخوانی بازگشتی به الگوریتم برای هر T exp انجام می شود. تاکنون هیچ الگویی بسته نیست.

ما تمام فراخوانی های بازگشتی را توضیح نمی دهیم، بلکه بر روی CFSP-Miner (D, F , M, (O 1 , { X}) (تمرکز می کنیم. برای این فراخوانی هر T exp با گسترش (O 1، {X}) با یک مقدار مشخصه پیدا می شود. سپس T exp بر {(O1، X}، ({Y ،(O1، {{{Z ،{Y ،(O1 ،Y}) تکرار می شود.

برای هر T exp، F requonent F (T exp) = 2. توجه داشته باشید که (O 1 = T exp، X}، Y}) یک استخراج از ({X ،O1}) است زیرا دارای همان فرکانس است و اولین الگو است که در ترتیب لغوی گسترش یافته است. بنابراین فراخوانی مجدد بعدی برای هرT equal−fcl : زیر مجموعه ای از (O1، X}، Y}) است که دارای یک FCL مشابه ({X ،O1}، Y}) و آخرین خصوصیت (O 1، X}، Y}) می باشد. در این مورد تنها T equal−fcl ، {X ،O1}، Y}) است. در نهایت، در فراخوانی بازگشتی، آن را به مجموعه الگوهای مشابه بسته مکرر اضافه می کند، زیرا هیچ استخراجی ندارد. فراخوانی های برگشت پذیر بعدی توضیح داده نخواهد شد.

یک وضعیت مشابه برای (O1 = exp T، Y}، Z}) و سپس (O1، Y، Z} نیز به مجموعه الگوهای مشابه متداول اضافه شده است.

به دنبال تمام فراخوانی های بازگشتی، تمام الگوی های متداول بسته مکرر ظاهر می شود.

5.1. تجزیه و تحلیل پیچیدگی

برای تحلیل پیچیدگی محاسباتی CFSP-Miner، بدترین حالت را مورد بررسی قرار می دهیم. ورودی CFSP-Miner شامل یک مجموعه داده از m اشیاء O است که هر کدام با مجموعه ای از ویژگی های A شرح داده شده است. بنابراین اندازه ورودی MN است. از سوی دیگر، اندازه خروجی، تعداد c از الگوهای مکرر مشابه بسته است. فرض می کنیم که هزینه محاسبه شباهت بین یک الگو و یک شیء، O (n) است، زیرا صفات n باید مقایسه شوند.

بر تعداد عملیات که باید برای پیدا کردن هر یک از الگوهای متداول بسته انجام شود.، تمرکز می کنیم. برای هر یک از الگوهای مشابه مکرر بسته، باید آن را با توجه به تمام اشیاء در مجموعه داده مقایسه کرد تا فرکانس آن محاسبه شود. بنابراین، محاسبه فرکانس یک کاندیدای الگو، O (mn) است. برای هر یک از کاندیداهای مشابه بسته، IFCL نیز باید محاسبه شود، که O (n 2) است. بنابراین، پیچیدگی CFSP-Miner برابر است با O (cmn 3).

تجزیه و تحلیل مشابهی می تواند برای الگوریتم ObjectMiner و STreeDCMiner انجام شود. با توجه به اینکه IFCL نیازی به محاسبه ندارد، پیچیدگی محاسباتی در هر دو مورد O (pmn) است، جایی که p تعداد الگوهای مشابهی است که به عنوان خروجی به دست می آید.

اگر تعدادی از الگوهای مکرر بسته مشابه شبیه تعداد الگوهای مشابه مکرر است، توجه داشته باشید که پیچیدگی محاسباتی ObjectMiner و STreeDC-Miner کوچکتر از پیچیدگی محاسباتی CFSP-Miner است. اگر تعداد الگوهای بسته مشابه بسیار کمتر از تعداد الگوهای مشابه شکل گرفته باشد، از سوی دیگر پیچیدگی محاسباتی CFSP-Miner کمتر از پیچیدگی محاسبات ObjectMiner و STreeDC-Miner می باشد. این شرایط در بخش بعدی آزمایش شده است.

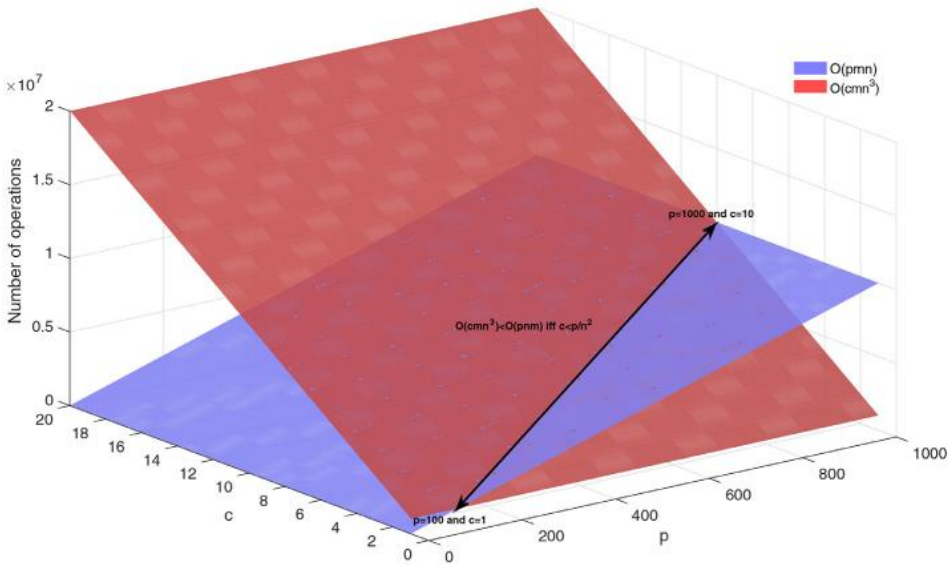
به صورت تحلیلی، $\forall c > 0,\ \exists p > 0,\ c < p/n^2$ ، به طوری که cmn 3 <pmn. شکل 3 یک مورد خاص در مورد تعداد عملیات مورد نیاز CFSP-Miner و ObjectMiner و STreeDC-Miner را در تعدیل تعداد الگوهای مشابه مکرر p از [1، 10 0 0] و تعدادی از الگوهای مکرر مشابه بسته c از [1،20]است؛ و m = 1000 اشیاء و n = 10 ویژگی. این شکل نشان می دهد که یک منطقه وجود دارد که تعداد عملیات مورد نیاز توسط CFSP-Miner کمتر از تعداد عملیات مورد نیاز ObjectMiner و STreeDC-Miner است

همچنین لازم به ذکر است که تعدادی از الگوها در خروجی برای سه الگوریتم به صورت غریزی در رابطه با اندازه ورودی محدود شده اند.

6. نتایج تجربی

در این بخش عملکرد الگوریتم پیشنهادی CFSP-Miner ارزیابی می شود. ما نتایج تجربی را به دو بخش تقسیم می کنیم. در بخش اول (بخش 6.1) مقایسه ای از لحاظ زمان مورد نیاز برای استخراج الگوهای مشابه مکرر توسط هر الگوریتم (CFSP-Miner، ObjectMiner و STreeDC-Miner) ارائه شده است. مهم است که مشخص شود هر الگوریتم ObjectMiner و STreeDC-Miner مجموعه ای از همه الگوهای مشابه مرسوم S را بدست آورده است، در حالی که الگوریتم پیشنهادی، CFSP-Miner، تنها مجموعه ای از تمام الگوهای مشابه بسته $S \cap E$ را به دست می آورد. در بخش دوم (بخش 6.2) مقیاس پذیری الگوریتم CFSP-Miner نشان داده شده است.

این آزمایش ها بر روی یک کامپیوتر با پردازنده Intel (R) Core (TM) 2 Duo در 1.83 Ghz و 2 گیگابایت رم انجام شد. الگوریتم CFSP-Miner در CSharp اجرا شد. برای ObjectMiner و الگوریتم STreeDC-Miner، پیاده سازی جاوا نویسندگان آنها مورد استفاده قرار گرفت.



یک مورد خاص از تعداد عملیات های مورد نیاز توسط CFSP-Miner، ObjectMiner و STreeDC-Miner، تعدادی از الگوهای مشابه مکرر p از [1، 10 0 0]، تعدادی از الگوهای بسته شده مشابه شایع از [1،20] ، و با توجه به m = 10 0 0 اشیاء و n = 10 ویژگی.

6.1. کارایی الگوریتم CFSP-Miner

جدول 1 شرح مجموعه داده ها 1 مورد استفاده را نشان می دهد. یک تابع تشابه متفاوت می تواند برای هر یک از مشکلات خاص تعریف شود. با این حال، هدف از این آزمایش ها حل کردن یک مشکل خاص نیست، بلکه صرفا به منظور ارزیابی کارایی الگوریتم پیشنهاد شده است. به همین دلیل توابع شباهت به هیچ مشکلی خاص در این کار وابسته

نیستند.

Table 1
Description of datasets.

Datasets	Objects	Non-numerical features	Numerical features
Dermatology	366	34	1
Flags	194	20	10
Mushroom	8124	22	0
Waveform	5000	40	1
Vehicle	946	1	18
Wine	178	1	13

برای این آزمایش، تابع شباهت F^{α}_{prod} ، در نمونه 2 تعریف شده، با $\alpha = 0.1$ استفاده شد. به عنوان معیار مقایسه ما از موارد زیر استفاده کردیم:

اگر A عددی است:

$$C_A(A_{v1}, A_{v2}) = \begin{cases} ss1 & \text{if } \frac{|A_{v1} - A_{v2}|}{Max_A - Min_A} \leq \theta \\ 0 & otherwise \end{cases}$$

اگر A ویژگی غیر عددی است:

$$C_A(A_{v1}, A_{v2}) = \begin{cases} 1 & \text{if } A_{v1} = A_{v2} \\ 0 & otherwise \end{cases}$$

شکل 4 زمان اجرا الگوریتم ها را برای هر مجموعه داده ، وقتی که درصد فرکانس M٪ از 10 تا 90 تغییر می کند را نشان می دهد. توجه کنید که $M\% = \frac{M}{\|\sigma\|}$ که در آن M حداقل آستانه فرکانس است. CFSP-Miner از زمان اجرا ObjectMiner و STreeDCMiner در 4 (Waveform Flags, Dermatology, Mushroom,) از 6 مجموعه داده ها بهتر عمل می کند. در حالی که مخالف آن در داده های WineوVehicle اتفاق می افتد.

این رفتار با مشاهده درصد تکرار الگوهای مشابه که بسته هستند (درصد در شکل 5) و با در نظر گرفتن تجزیه و تحلیل پیچیدگی الگوریتم ها توضیح داده شده است. به خاطر داشته باشید که برای هر یک از کاندیداهای مشابه بسته ، IFCL نیز باید توسط CFSP-Miner محاسبه شود، در حالی که STreeDC-Miner و ObjectMiner این گام را برای هر یک از کاندیداهای مشابه ارائه نمی کنند. سپس زمانی که تعداد الگوهای مشابه مکرر بسته تقریبا برابر با مقدار الگوهای مکرر مشابه است (مجموعه داده های WineوVehicle)، STreeDC-Miner و ObjectMiner زمان اجرای بیشتری نسبت به CFSP-Miner دارند. از طرف دیگر، مقدار الگوهای مکرر مشابه بسته که میزان تایید را بیان می کند و همچنین مقدار IFCL محاسبه شده نیز کاهش می یابد. سپس زمان اجرا CFSP-Miner نسبت به زمان اجرا STreeDC-Miner و ObjectMiner کاهش می یابد. در این آزمایش، CFSP-Miner در زمان اجرا از STreeDCMiner و ObjectMiner بهتر است زمانی که کمتر از 80 درصد از الگوهای مشابه مکرر بسته است (Flags، Dermatology، Mushroom، and Waveform datasets).

مهم است که مشخص شود که ویژگی اصلی CFSPMiner توانایی آن در پیدا کردن الگوهای مشابه "بسته" است، که باعث کاهش تعداد الگوهای مشابه مکرر بدون از دست دادن اطلاعات می شود. CFSP-Miner همچنین می تواند زمان اجرا ObjectMiner و STreeDC-Miner را با توجه به درصد الگوهای مکرر مشابه بسته، بهتر نماید. برای تجزیه و تحلیل دقیق تر رفتار CFSP-Miner، مجموعه داده های مختلف 2 با درصد تعریف شده از الگوهای مکرر مشابه که بسته اند، به صورت خودکار تولید می شوند.

اول، سه عدد طبیعی X (برای مقدار اشیاء)، Y (برای مقدار ویژگی ها)، و Z (برای قدرتمند بودن دامنه های ویژگی) ثابت می شوند. سپس D به طوری ساخته شده است که $\|O\| = X,$ و $\|A\| = Y$ ، که در آن Vrandomیک مقدار برای ویژگی تصادفی V با توزیع یکنواخت روی $\{1, 2, \ldots, Z\}$ است. $O(X)A(Y)D(Z)$ مجموعه ای از تمام مجموعه های داده D را که توسط X، Y و Z تعریف شده است، نشان می دهد.

اولین آزمایش برای اثبات اینکه CFSPMiner هنگامی که درصد نزدیک به 100٪ است، دارای عملکرد ضعیف است انجام می شود. اجازه دهید $\mathcal{D}_0 \in O(10000)A(10)D(100)$ یک مجموعه داده باشد، به طوری که $Percent(M) = 100\%$ برای حداقل مقدار آستانه فرکانس 10٪، 30٪، 50٪، 70٪ و 90٪ باشد.

برای به دست آوردن D0، لازم بود که کمتر از 10 برابر مجموعه داده های تصادفی تولید شود. این امر می تواند ادعا شود زیرا ویژگی یک متغیر تصادفی گسسته با توزیع یکنواخت روی {1، 2، . . . ، 100} احتمال بسیار کم $\exists T \in \mathcal{T} \mid FCL(T) \neq T$ و احتمال بالای $Percent(M) = 100\%$ را تضمین می کند. اثبات این موضوع نامربوط است، اما کافی است بگوییم که D0 با این خصوصیات به دست آمده است.

شکل a6 نشان می دهد که CFSP-Miner بدترین عملکرد را از الگوریتم های آزمایش شده برای Dataset D0 دارد، که رفتار پیش بینی شده در زمانی است که درصد نزدیک به 100٪ است. با این حال، همانطور که در زیر نشان داده شده است، عملکرد CFSP-Miner ، هنگامی که درصد کاهش می یابد، بهبود می یابد.

برای اثبات این که یک متغیر تصادفی پیوسته با توزیع نرمال $N(Z/2, 1/Z)$ برای تولید مقادیر برای یک ویژگی، استفاده می شود. پس از آن، A چنین ویژگی را نشان می دهد و $O(X)A(Y)D(Z)^*$ مجموعه ای از تمام مجموعه های داده D را می توان به این طریق تولید کرد.

برای آزمایش بعدی * (100) D (10) A (0 0 0 0) O ∈ 1 D با درصد Percent (M) < 75% ، 2 D

** (10 0) D (10 0 0 0) A (10) O ∈ با Percent (M) < 50% و (10) A (10 0 0 0) O ∈ 3 D

*** (10 0) D با Percent (M) < 25% برای حداقل فرکانس آستانه {10٪، 30٪، 50٪، 70٪، 90٪} ایجاد شد.

برای به دست آوردن 1 D، لازم بود که کمتر از 20 بار مجموعه داده تصادفی تولید کنیم. این واقعیت توجیه شده است زیرا توزیع احتمالی که از آن مقدار Z تولید می شود، تضمین می کند که احتمال FCL (T) = T بالا است و درصد احتمال، احتمال کاهش بالایی دارد. همانند موارد قبلی، اثبات این موضوع مربوط نیست و کافی است بگویم که 1 D با این ویژگی ها به دست آمده است.

** (10 0) D (10 0 0 0) A (10) O ∈ 2 D با درصد Percent (M) < 50% و (10 0 0 0) O ∈ 3 D

*** (10 0) D (10 0) A با درصد Percent (M) < 25% به طور مشابه به دست آمد.

توجه داشته باشید که زمان اجرای CFSP-Miner در شکل 6، برای مجموعه داده ها با درصد کمتر از 80 (یعنی 1 D، 2 D و 3 D) از الگوهای مشابه مکرر بسته بهترین است. این نتیجه نتایج تجربی قبلی را تایید می کند: زمانی که کمتر از 80 درصد از نمونه های مکرر مشابه بسته شده اند، CFSP-Miner در زمان اجرا از STreeDC-Miner و ObjectMiner بهتر عمل می کند.

برای تمام آستانه های حداقل فرکانس، CFSP-Miner همچنین عملکرد زمان اجرا خود را به تدریج از 0 D تا 3 D (به عنوان مثال، کاهش درصد) بهبود می بخشد.

6.2. مقیاس پذیری الگوریتم CFSP-Miner

برای نشان دادن مقیاس پذیری CFSP-Miner، چندین آزمایش مختلف با ابعاد مجموعه داده ها انجام شد. مجموعه داده های تصادفی با Percent(M) = 100% به صورت خودکار تولید می شوند. این مجموعه داده ها انتخاب شدند زیرا همانطور که در بخش 6.1 نشان داده شده است، این بدترین حالت برای الگوریتم CFSP-Miner را نشان می دهد.

شکل 7 مقیاس پذیری الگوریتم CFSP-Miner با توجه به a) اشیاء، b) ویژگی ها و c) دامنه ها را نشان می دهد. در شکل 7 دیده می شود (الف) زمانی که تعداد اشیا افزایش می یابد، زمان اجرا افزایش می یابد، همانطور که انتظار می رود. بدترین زمان اجرا هنوز در محدوده قابل قبول نسبت به نتایج در بخش قبلی است.

اتفاق مشابهی در شکل ب7 هنگامی که تعداد ویژگی ها افزایش می یابد، نشان داده شده است. توجه داشته باشید که افزایش تعداد ویژگی ها باعث افزایش چشمگیر فضای جستجو از الگوهای مشابه مکرر می شود که زمان اجرا کند می شود. علی رغم این واقعیت، CFSP-Miner هنوز در مقایسه با نتایج در بخش قبلی، زمانهای قابل قبول را به دست می آورد. این مقیاس پذیری خوبی را برای طیف وسیعی از اشیاء و ویژگی های نشان داده شده در این آزمایش نشان می دهد.

شکل 7 (c) نشان می دهد که زمانی که حجم دامنه افزایش می یابد زمان اجرا کاهش می یابد. این افزایش در اندازه دامنه موجب کاهش احتمال داشتن مقادیر مکرر به دلیل استفاده از توزیع یکنواخت می شود. تعداد الگوهای مکرر کاهش می یابد و زمان اجرای آن کاهش می یابد.

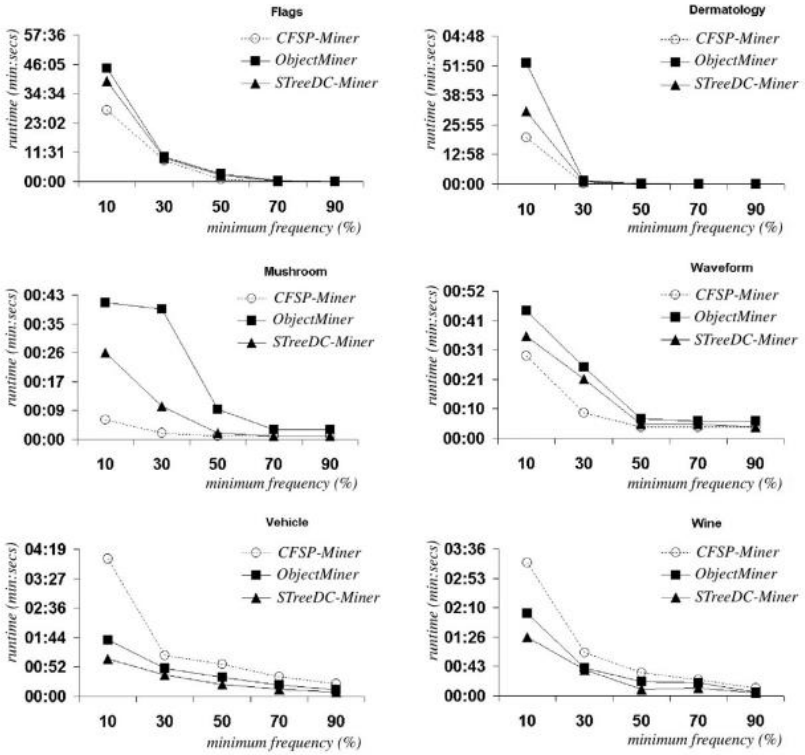
به طور خلاصه، ما می توانیم از شکل 7 مشاهده کنیم که تعداد ویژگی ها پارامتری است که بیشترین تاثیر منفی بر مقیاس پذیری الگوریتم ما را دارد. این مطابق با تجزیه و تحلیل پیچیدگی ارائه شده در بخش 5.1 است.

7. نتیجه گیری

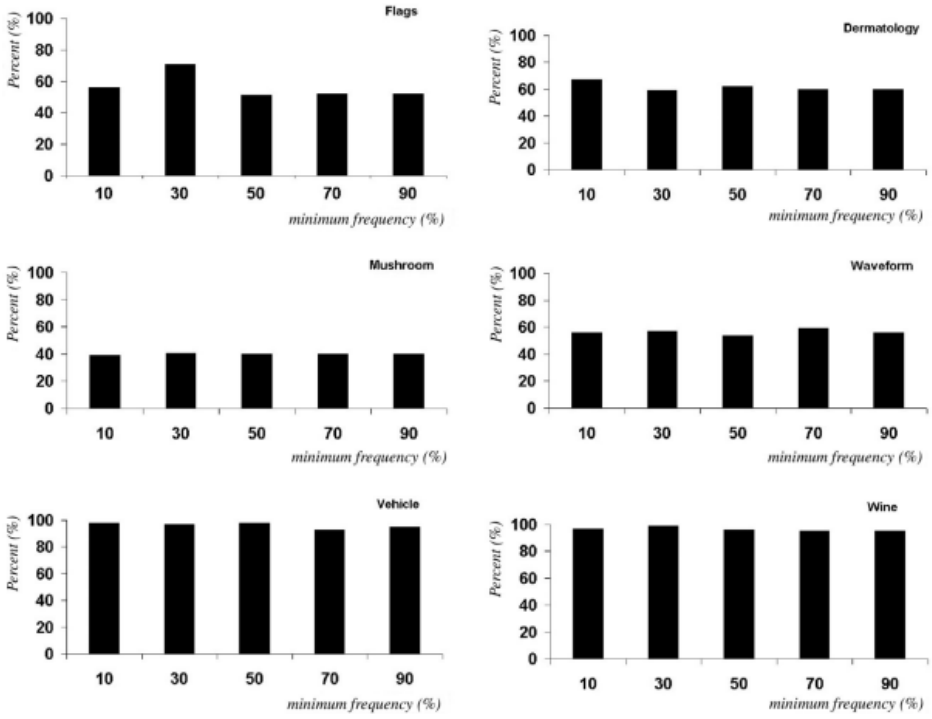
در این مقاله ما مفهوم یافتن الگوی مشابه مکرر بسته را برای کشف یک مجموعه کاهش یافته از الگوهای مشابه مکرر بدون از دست دادن اطلاعات ارائه دادیم. ما همچنین یک الگوریتم استخراج الگوی مشابه مکرر را ، به نام CFSP-Miner پیشنهاد کردیم ، که با استفاده از توابع مشابه شباهت های ریاضی برای پیدا کردن همه الگوهای مشابه بسته به کار می رود.

نتایج نشان می دهد که الگوریتم پیشنهادی CFSP-Miner کارآمدتر از سایر الگوریتم های یافتن الگوهای مشابه مکرر موجود در پیشینه است، مگر اینکه در مواردی که تعداد الگوهای مشابه مکرر و تعدادی از الگوهای مشابه بسته مکرر تقریبا برابر باشند. یکی دیگر از قدرت های CFSP-Miner توانایی آن در پیدا کردن الگوهای مشابه "بسته" بدون از دست دادن اطلاعات است. در بسیاری از مجموعه داده های تجزیه و تحلیل شده، CFSP-Miner مقدار تقریبی نمونه های مکرر کشف شده را تقریبا 50٪ کاهش داد. CFSP-Miner همچنین می تواند برای استفاده در مجموعه داده های با ابعاد بزرگ مورد استفاده قرار گیرد زیرا نتایج تجربی نشان می دهد که افزایش تعداد اشیاء، تعداد ویژگی ها یا اندازه هر دامنه ویژگی، زمان اجرا را در محدوده قابل قبول حفظ می کند.

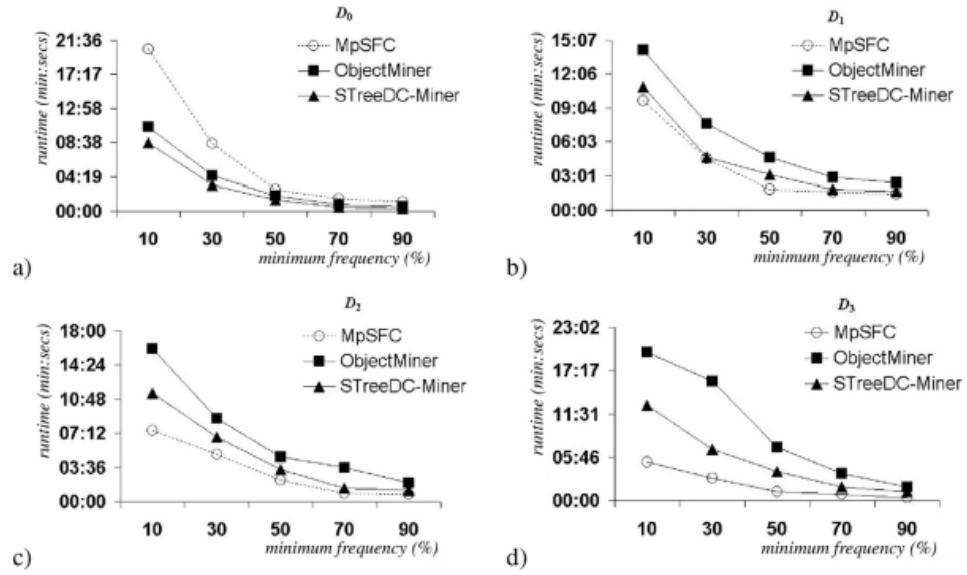
برای کار آتی، ما بهبود بهره وری CFSPMiner، کاوش در ایده های ارائه شده در کارهای اخیر برای بهبود بهره وری از الگوریتم های یافتن الگوهای مکرر بسته سنتی و مطالعه امکان سنجی گسترش این نتایج به یافتن الگو مکرر مشابه بسته را تجسم. می کنیم. گسترش روش های یافتن الگوهای مشابه مکرر بسته، یافتن قوانین ارتباطات برای توابع شباهت غیر بولین و توابع شباهت غیر مونوتونی یکی دیگر از کارهای جالب آینده است.



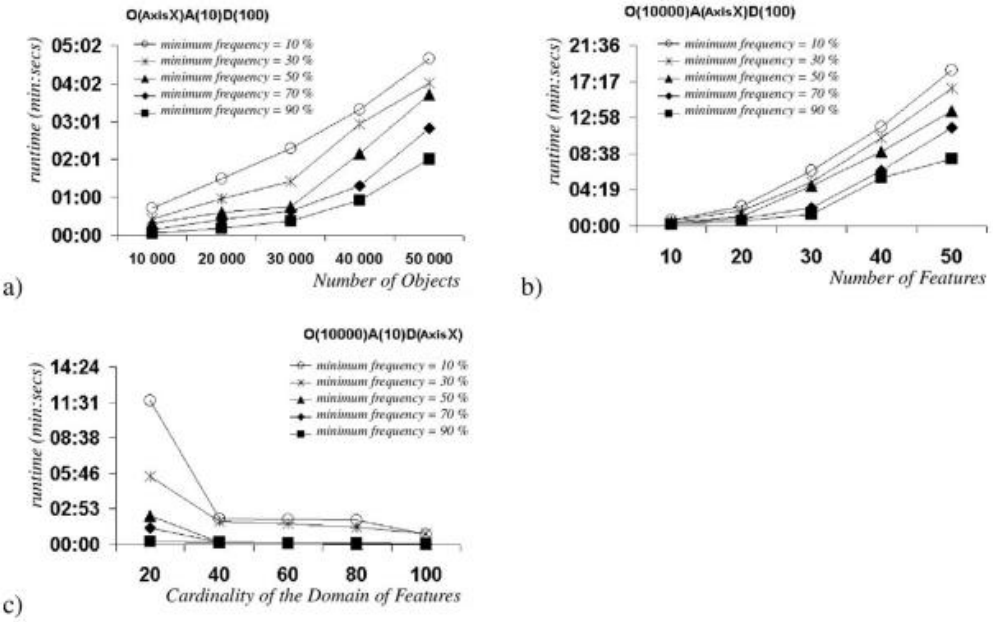
شکل 4. الگوریتم های CFSP-Miner، ObjectMiner و STreeDC-Miner برای مجموعه داده های جدول 1، زمان اجرا.



شکل5. درصد از الگوهای مشابه مکرر بسته به الگوهای مکرر مشابه



شکل 6. زمان اجرا الگوریتم های CFSP-Miner، ObjectMiner و STreeDC-Miner برای مجموعه داده ها



شکل 7. توانایی الگوریتم .CFSP-Minerالف) w.r.t اشیاء ب) ویژگی های w.r.t ج) دامنه w.r.t.

References

Agrawal, R., Imielinski, ´ T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In Acm sigmod record: 22 (pp. 207–216). ACM.

Agrawal, R., et al. (1994). Fast algorithms for mining association rules. In Proc. 20th int. conf. very large data bases, vldb: 1215 (pp. 487–499).

Alatas, B., Akin, E., & Karci, A. (2008). Modenar: Multi-objective differential evolution algorithm for mining numeric association rules. Applied Soft Computing, 8(1), 646–656.

Baeza-Yates, R., et al. (1999). Modern information retrieval: 463. ACM press New York. Beil, F., Ester, M., & Xu, X. (2002). Frequent term-based text clustering. In Proceedings of the eighth acm sigkdd international conference on knowledge discovery and data mining (pp. 436–442). ACM.

Burdick, D., Calimlim, M., & Gehrke, J. (2001). Mafia: A maximal frequent itemset algorithm for transactional databases. In Data engineering, 2001. proceedings. 17th international conference on (pp. 443–452). IEEE.

Chiu, C.-Y., Yeh, C.-T., & Lee, Y.-J. (2013). Frequent pattern based user behavior anomaly detection for cloud system. In 2013 conference on technologies and applications of artificial intelligence (pp. 61–66). IEEE.

Danger, R., Ruiz-Shulcloper, J., & Llavori, R. B. (2004). Objectminer: A new approach for mining complex objects.. In Iceis (2) (pp. 42–47). Citeseer.

Fan, Y., Ye, Y., & Chen, L. (2016). Malicious sequential pattern mining for automatic malware detection. Expert Systems with Applications, 52, 16–25.

Gómez-Herrera, J., et al. (1994). Prognostic of gas-oil deposits in the cuban ophiological association. Applying Mathematical Modeling, Geofisica Internacional, 33(3), 447467.

Han, H. L. J., & Shao, D. X. Z. (2006). Mining frequent patterns from very high dimensional data: A top-down row enumeration approach. In Proceedings of the sixth siam international conference on data mining: 124 (p. 282). SIAM.

Hashem, T., Karim, M. R., Samiullah, M., & Ahmed, C. F. (2017). An efficient dynamic superset bit-vector approach for mining frequent closed itemsets and their lattice structure. Expert Systems with Applications, 67, 252–271. doi:10.1016/j.eswa. 2016.09.023.

Hernández-León, R., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F., & Hernández– Palancar, J. (2012). Classification based on specific rules and inexact coverage. Expert Systems with Applications, 39(12), 11203–11211.

Hu, T., Sung, S. Y., Xiong, H., & Fu, Q. (2008). Discovery of maximum length frequent itemsets. Information Sciences, 178(1), 69–87.

Kalpana, B., & Nadarajan, R. (2008). Incorporating heuristics for efficient search space pruning in frequent itemset mining strategies. Current science, 94(1), 97–101.

Le, T., & Vo, B. (2015). An n-list-based algorithm for mining frequent closed patterns. Expert Systems with Applications, 42(19), 6648–6657. doi:10.1016/j.eswa.2015.04. 048.

Lee, A. J., Wang, C.-S., Weng, W.-Y., Chen, Y.-A., & Wu, H.-W. (2008). An efficient algorithm for mining closed inter-transaction itemsets. Data & Knowledge Engineering, 66(1), 68–91.

Li, J., Fu, A. W.-c., & Fahey, P. (2009). Efficient discovery of risk patterns in medical data. Artificial intelligence in medicine, 45(1), 77–89.

Li, J., et al. (2005). Mining risk patterns in medical data. In Proceedings of the eleventh acm sigkdd international conference on knowledge discovery in data mining (pp. 770–775). ACM.

Liu, H., et al. (2009). Top-down mining of frequent closed patterns from very high dimensional data. Information Sciences, 179(7), 899–924.

Lopez, F. J., Blanco, A., Garcia, F., Cano, C., & Marin, A. (2008). Fuzzy association rules for biological data analysis: a case study on yeast. BMC bioinformatics, 9(1), 1.

Moonesinghe, H., Fodeh, S., & Tan, P.-N. (2006). Frequent closed itemset mining using prefix graphs with an efficient flow-based pruning strategy. In Sixth international conference on data mining (icdm'06) (pp. 426–435). IEEE.

Nahar, J., Imam, T., Tickle, K. S., & Chen, Y.-P. P. (2013). Association rule mining to detect factors which contribute to heart disease in males and females. Expert Systems with Applications, 40(4), 1086–1093.

Nair, B., & Tripathy, A. K. (2011). Accelerating closed frequent itemset mining by elimination of null transactions. Journal of Emerging Trends in Computing and Information Sciences, 2(7), 317–324.

Nezhad, J. T., & Sadreddini, M. (2007). Ptclose: A novel algorithm for generation of closed frequent itemsets from dense and sparse datasets. In Proceedings of the world congress on engineering: 1.

Nguyen, L. T., & Nguyen, N. T. (2015). An improved algorithm for mining class association rules using the difference of obidsets. Expert Systems with Applications, 42(9), 4361–4369.

Ortiz-Posadas, M. R., Vega-Alvarado, L., & Toni, B. (2009). A mathematical function to evaluate surgical complexity of cleft lip and palate. Computer methods and programs in biomedicine, 94(3), 232–238.

Pan, F., Tung, A. K., Cong, G., & Xu, X. (2004). Cobbler: combining column and row enumeration for closed pattern discovery. In Scientific and statistical database management, 2004. proceedings. 16th international conference on (pp. 21–30).

IEEE. Pei, J., et al. (2000). Closet: An efficient algorithm for mining frequent closed itemsets.. In Acm sigmod workshop on research issues in data mining and knowledge discovery: 4 (pp. 21–30). Prabha, S., Shanmugapriya, S., & Duraiswamy, K. (2013). A survey on closed frequent pattern mining. International Journal of Computer Applications, 63(14).

Rodríguez-González, A. Y., Martínez-Trinidad, J. F., Carrasco-Ochoa, J. A., & Ruiz-Shulcloper, J. (2008). Mining frequent similar patterns on mixed data. In Iberoamerican congress on pattern recognition (pp. 136–144). Springer.

Rodríguez-González, A. Y., Martínez-Trinidad, J. F., Carrasco-Ochoa, J. A., & Ruiz-Shulcloper, J. (2011). Rp-miner: A relaxed prune algorithm for frequent similar pattern mining. Knowledge and information systems, 27(3), 451–471.

Rodríguez-González, A. Y., Martínez-Trinidad, J. F., Carrasco-Ochoa, J. A., & Ruiz-Shulcloper, J. (2013). Mining frequent patterns and association rules using similarities. Expert Systems with Applications, 40(17), 6823–6836.

Ruiz-Shulcloper, J., & Fuentes-Rodríguez, A. (1981). A cybernetic model to analyze juvenile delinquency. Revista Ciencias Matemáticas, 2(1), 123–153.

Uno, T., Asai, T., Uchida, Y., & Arimura, H. (2003). Lcm: An efficient algorithm for enumerating frequent closed item sets.. Fimi: 90. Citeseer.

Vo, B., Hong, T.-P., & Le, B. (2012). Dbv-miner: A dynamic bit-vector approach for fast mining frequent closed itemsets. Expert Systems with Applications, 39(8), 7196–7206. doi:10.1016/j.eswa.2012.01.062.

Weisstein, E. W. (2002). CRC concise encyclopedia of mathematics. CRC press.

Wen, J., Zhong, M., & Wang, Z. (2015). Activity recognition with weighted frequent patterns mining in smart environments. Expert Systems with Applications, 42(17), 6423–6432.

Zaki, M. J., & Hsiao, C.-J. (2002). Charm: An efficient algorithm for closed itemset mining.. In Sdm: 2 (pp. 457–473). SIAM.