ارزیابی الگوریتم های استخراج زیر گراف های پر تکرار

چکیده

استخراج گراف قلمرو تحقیقی مهمی در حوزه استخراج داده است. زمینه تحقیق بر تعیین زیر گراف های پرتکرار[1] در مجموعه داده های گراف تمرکز می کند. اهداف تحقیق عبارتند از: (i) مکانیسم های (سازوکارهای) مؤثر برای تولید زیر گراف های داوطلب (بدون تولید نسخه های کپی) و (ii) بهترین شیوه برای پردازش زیر گراف های داوطلب برای تعیین زیر گراف های پرتکرار مطلوب به شکلی که از نظر محاسباتی کار آمد و به لحاظ روال کار مؤثر باشد. این مقاله یک ارزیابی کلی از تحقیق های کنونی در حوزه استخراج داده های پرتکرار ارائه می دهد، و راه حل هایی برای پرداختن به موضوعات اصلی تحقیق پیشنهاد می دهد.

[1] frequent subgraphs

۱. مقدمه

هدف اولیه استخراج داده ها، استخراج کلان و مفید (به لحاظ آماری) دانش و اطلاعات از داده هاست (چن و دیگران ۱۹۹۶؛ هان و کامر ۲۰۰۶). داده های مهم می توانند به شکل های زیادی ظاهر شوند: بردارها، جدول ها، متن ها، تصاویر و غیره، همچنین، داده ها را می توان با ابزارهای گوناگونی بیان کرد. داده های ساختاری و نیمه ساختاری به طور طبیعی برای ارائه به شکل گراف مناسب هستند. به عنوان مثال، اگر به شبکه های متقابل پروتئین ـ پروتئین (حوزه کاربردی متداولی برای استخراج داده) توجه کنیم، می توان اینها را در قالب (فرمت) گراف بیان کرد به طوری که رأس ها نشان دهنده ژن ها هستند و کران های مستقیم یا غیر مستقیم نشان دهنده برهم کنش فیزیکی یا پیوندهای کنش (عملی) هستند (آلم و آرکین، ۲۰۰۳). به این دلیل که بیان داده های ساختاری و نیمه ساختاری در قالب گراف به سهولت انجام می گیرد، علاقه و اشتیاق فراوانی برای استخراج داده های گراف وجود داشته است (که اغلب به عنوان استخراج داده های گراف ـ محور یا استخراج گراف شناخته می شود). بعضی از زیر مجموعه های حوزه های تحقیقی متداول در استخراج گراف در جدول شماره ۱ فهرست شده اند.

جدول ۱: زیر مجموعه های حوزه های تحقیقی متداول در استخراج گراف

استخراج زیر گراف پرتکرار (کوک وهولدر ۱۹۹۴، ۲۰۰۰؛ اینوکوچی و دیگران ۲۰۰۰ ,ان و هان ۲۰۰۲)
استخراج الگوی گراف متناظر (کی و دیگران ۲۰۰۷؛ کی و دیگران ۲۰۰۹؛ اوزاکی و اوکاوا ۲۰۰۸)
استخراج الگوی گراف مطلوب (ان و دیگران ۲۰۰۸؛ فن و دیگران ۲۰۰۸)
استخراج الگوی گراف تقریبی (کلی و دیگران ۲۰۰۳؛ شاران و دیگران ۲۰۰۵؛ چن و دیگران ۲۰۰۷)
خلاصه سازی و جمع بندی الگوی گراف (کین و دیگران ۲۰۰۶؛ چن و دیگران ۲۰۰۸)
طبقه بندی گراف (هوآن و دیگران ۲۰۰۴؛ کودو و دیگران ۲۰۰۴؛ دشپاند و دیگران ۲۰۰۵)
خوشه بندی گراف (فلیک و دیگران ۲۰۰۴ ؛ هوآنگ و لای ۲۰۰۶؛ نیومن ۲۰۰۴)
شاخص گذاری گراف (شاشا و دیگران ۲۰۰۲؛ان و دیگران ۲۰۰۴)
جستجوی گراف (ان و دیگران ط ۲۰۰۵؛ان و دیگران ۲۰۰۶؛ چن و دیگران ط۲۰۰۷)
هسته اصلی گراف (گارتنر و دیگران ۲۰۰۳؛ کاشیما و دیگران ۲۰۰۳؛ بورگ وارت و کریگل ۲۰۰۵)
استخراج لینک (چاکراباتی و دیگران ۱۹۹۹؛ کوسالا و بلوکیل ۲۰۰۰؛ گنور ودایل ۲۰۰۵؛ لیو ۲۰۰۸)
استخراج ساختار وب (کلین برگ ۱۹۹۸؛ برین و پیج ۱۹۹۸)
استخراج جریان کار (گرکو و دیگران ۲۰۰۵)

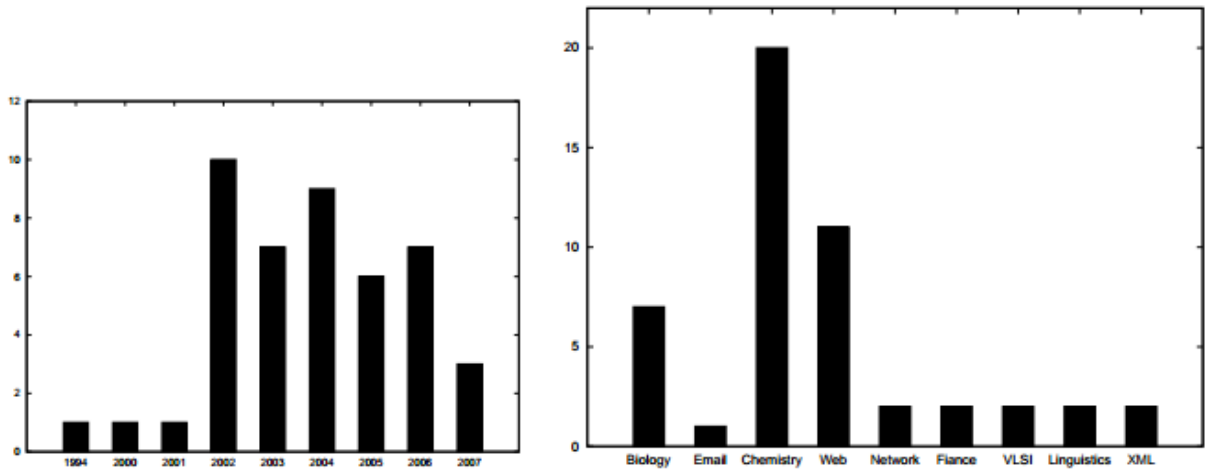
استخراج شبکه زیستی (هوو و دیگران ۲۰۰۵)

استخراج زیر گراف پرتکرار (FSM) جوهره و هسته اصلی استخراج گراف است. هدف اصلی FSM استخراج همه زیر گراف های پرتکرار در یک مجموعه داده معین است که تعداد وقوع آنها بالاتر از آستانه تعیین شده است. تصویر ۱، نمای کلی حوزه FSM از نظر تعداد الگوریتم های پیشنهاد شده در بازه زمانی ۱۹۹۴ تا امروز را ارئه می دهد. با مشاهده این تصویر می توانیم دوره های فعالیت در سال های ابتدایی ۹۰ (همزمان با معرفی مفهوم استخرارج داده ها) و پس از آن دوره فعالیت دیگری از ۲۰۰۲ تا ۲۰۰۷ را مشاهده کنیم.

در چند سال اخیر هیچ الگوریتم جدیدی معرفی نشده است که نشان می دهد این حوزه به مرحله بلوغ و کمال رسیده است، علیرغم اینکه کارهای زیادی که بر فرم ها و اشکال مختلف الگوریتم های موجود تمرکز کرده اند باقی مانده است.

به جز فعالیت های تحقیقی گسترده ای که وابسته به FSM است، اهمیت و اعتبار FSM نشان دهنده حوزه کاربردی فراوان آن است. تصویر (b) ۱یک نمای کلی از حوزه کاربردی FSM از لحاظ تعداد الگوریتم های FSM که در آثار و کتاب ها ارائه شده و حوزه کاربردی ویژه ای که نشانه گرفته اند، ارائه می دهد. از این تصویر می توان مشاهده کرد که سه حوزه کاربردی (شیمی، وب و زیست شناسی) در استفاده از الگوریتم های FSM برتری داشته اند.

تصویر ۱. توزیع مهمترین الگوریتم های FSM با توجه به سال معرفی و حوزه کاربردی



(a) Year (b) Application

ایده روشنی که پشت FSM وجود دارد، افزایش (تعمیم) زیر گراف های داوطلب به شیوه ای وسعت ـ محور یا عمق ـ محور است، و پس از آن تعیین اینکه آیا زیر گراف های داوطلب مشخص شده به قدر کافی در داده های گراف پرتکرار هستند که بتوان آنها را جالب فرض کرد (محاسبه پشتیبانی). دو موضوع عمده تحقیق در FSM، چگونگی (i) ایجاد زیر گراف های پرتکرار داوطلب و (ii) تعیین فراوانی پیشامد زیر گراف های ایجاد شده، به شکلی مؤثر و کارآمد است. ایجاد زیر گراف های داوطلب کارآمد مستلزم آن است که از ایجاد نسخه کپی یا داوطلب های زائد جلوگیری شود. محاسبه پیشامد زیر گراف ها نیازمند مقایسه زیر گراف های داوطلب با زیر گراف های داده های ورودی است، فرآیندی که به عنوان بررسی هم ریختی شناخته می شود. FSM را از بسیاری جهات می توان به عنوان بسط ایده استخراج مجموعه آیتم های پر تکرار (FIM) در چارچوب قوانین و اصول وابسته به استخراج ارزیابی کرد (برای مثال هایی در این زمینه به آگراوان و اسریکانت ۱۹۹۴ مراجعه کنید). متعاقباً، اکثر راه حل های پیشنهادی برای بررسی موضوعات تحقیقی عمده در زمینه FSM بر تکنیک های مشابهی استوارند که در حوزه FSM یافت می شوند. به عنوان مثال، ویژگی اسناد نزولی وابسته به مجموعه آیتم ها در مقوله ایجاد زیر گراف های داوطلب به طور گسترده ای مورد استفاده قرار می گیرد.

در این مقاله، نویسندگان یک ارزیابی کلی از "وضعیت عملی" کنونی FSM ارائه می دهند. با مراجعه به آثار و نوشته های مربوطه، می توانیم انواع گوناگونی از استراتژی های استخراج را مشخص کنیم که با توجه به انواع مختلف گراف برای تولید انواع مختلف الگوها به کار گرفته می شوند. برای تحمیل بعضی از مقررات به حوزه FSM، بر ماهیت الگوریتم های FSM تمرکز کرده ایم؛ این الگوریتم ها را بر اساس (i) استراتژی ایجاد داوطلب (ii) مکانیسم های عبور از فضای جستجو، و (iii) فرآیند شمارش پیشامد طبقه بندی کرده ایم. به منظور تسهیل درک مقوله FSM میان استخراج درختچه های گراف پرتکرار و حوزه کلی تر استخراج زیر گراف های پرتکرار تمایز و تفکیک قائل می شویم. بقیه این مقاله به شکل زیر سازماندهی شده است. بخش ۲ را با ارائه تعاریف رسمی و اصطلاحات و واژگان آغاز می کنیم. در بخش ۳،یک ارزیابی کلی از فرآیند FSM ارائه میشود. در بخش ۴ و ۵، به ترتیب الگوریتم های کنونی استخراج درختچه های گراف و زیر گراف ها را مورد ملاحظه قرار می دهیم. در نهایت، در بخش ۶، نتیجه گیری ها و راهنمایی های تکمیلی ارائه می شود.

۲. فرمالیسم

دو دستور العمل جداگانه برای FSM وجود دارد: (i) FSM بر مبنای فعالیت گراف و (ii) FSM بر مبنای گراف مجزا. در FSM بر مبنای فعالیت (روابط متقابل) گراف، داده های ورودی شامل مجموعه ای از گراف های

متوسط به نام فعالیت (Transaction) است. توجه داشته باشید که واژه "فعالیت" از حوزه استخراج قواعد مرتبط (آگراوان و اسریکانت ۱۹۹۴) وام گرفته شده است. در FSM بر مبنای گراف مجزا، داده های ورودی، به گونه ای که از نام الگوریتم مشخص است، شامل یک گراف بسیار بزرگ است.

زیر گراف g پرتکرار در نظر گرفته می شود اگر احتمال پیشامد بیشتر از مقدار آستانه از پیش تعیین شده باشد. احتمال پیشامد برای یک زیر گراف معمولاً به عنوان پشتیبانی آن تعریف می شود، و پیرو آن آستانه به عنوان آستانه پشتیبانی تعریف می شود. پشتیبانی g ممکن است با استفاده از شمارش فعالیت ـ محور یا شمارش پیشامد ـ محور محاسبه شود. شمارش فعالیت ـ محور فقط برای FSM فعالیت ـ محور قابل اجراست در حالیکه شمارش پیشامد ـ محور برای هر دو مورد قابل اجراست. با این وجود، شمارش پیشامد ـ محور به طور معمول با FSM بر مبنای گراف مجزا مورد استفاده قرار می گیرد.

در شمارش فعالیت ـ محور، پشتیبانی با توجه به تعداد فعالیت هایی که g در آنها وجود دارد تعریف می شود، هر شمارش فارغ از اینکه g یکبار یا بیشتر از یکبار در یک فعالیت خاص اتفاق می افتد، صورت می گیرد. به همین خاطر، پایگاه داده های $g = \{G_1, G_2, \dots, G_T\}$ شامل مجموعه ای از فعالیت های گراف و آستانه پشتبانی $\sigma = (0 < \sigma \subseteq 1)$ داده می شود؛ سپس مجموعه ای از فعالیت های گراف که در آن زیر گراف g وجود دارد بر اساس $\sigma g = \{G_i / g \subseteq G_i\}$ تعریف می شود. بنابر این، پشتیبانی g به صورت زیر تعیین می شود:

$$SUP\ g = |\sigma g(g)|/T$$

که $|\sigma g(g)|$ بیانگر مقدار $\sigma g(g)$ و T بیانگر تعداد گراف های موجود در G است. بر این اساس، g پرتکرار خواهد بود اگر و تنها اگر $SUPg(g) \geq \sigma$. در شمارش پیشامد ـ محور به راحتی تعداد پیشامد g در مجموعه ورودی ها را محاسبه می کنیم.

شمارش فعالیت ـ محور دارای این امتیاز است که ویژگی اسناد نزولی (DCP) را می توان برای کاهش چشمگیر محاسبات جاری همراه با ایجاد داوطلب در FSM به کار گرفت. در شمارش پیشامد ـ محور، می بایست یک مقیاس فراوانی جایگزین که ویژگی DC را حفظ می کند تثبیت شود، یا روش های اکتشافی برای کاهش هزینه های محاسبات می بایست اتخاذ شود. انواع گوناگونی از مقیاس های پشتیبانی وجود دارد (وانتیک ۲۰۰۲ , کوراموش و کاریپیس ۲۰۰۴ و ۲۰۰۵؛ وانتیک و دیگران ۲۰۰۶) که ممکن است برای FSM بر مبنای گراف مجزا اعمال شود؛ اینها در بخش ۵,۱,۲ مورد بحث قرار خواهند گرفت.

۲,۱. تعاریف اولیه

به طور کلی، گراف به صورت مجموعه ای از رأس ها (گره ها) که به وسیله مجموعه ای از کران ها (لینک ها) متصل شده اند، تعریف می شود (گیبونز ۱۹۸۵). گراف های مورد استفاده در FSM به عنوان گراف های ساده برچسب دار تلقی می شوند. در زیر گراف های بعدی، تعدادی از تعاریف بسیار متداول که در ادامه این مقاله مورد استفاده قرار می گیرند ارائه می شوند.

گراف برچسب دار: گراف برچسب دار را می توان به صورت $G(V,E,L_V,L_E,\vartheta)$ تعریف کرد، که V مجموعه ای از رأس هاست، $E \subseteq V \times V$ مجموعه ای از کران هاست؛ L_E, L_V به ترتیب مجموعه ای از برچسب های رأس و کران هستند؛ و ϑک تابع بر چسب است که تناظر $V \to L_V$, $E \to L_E$ را تعریف می کند.

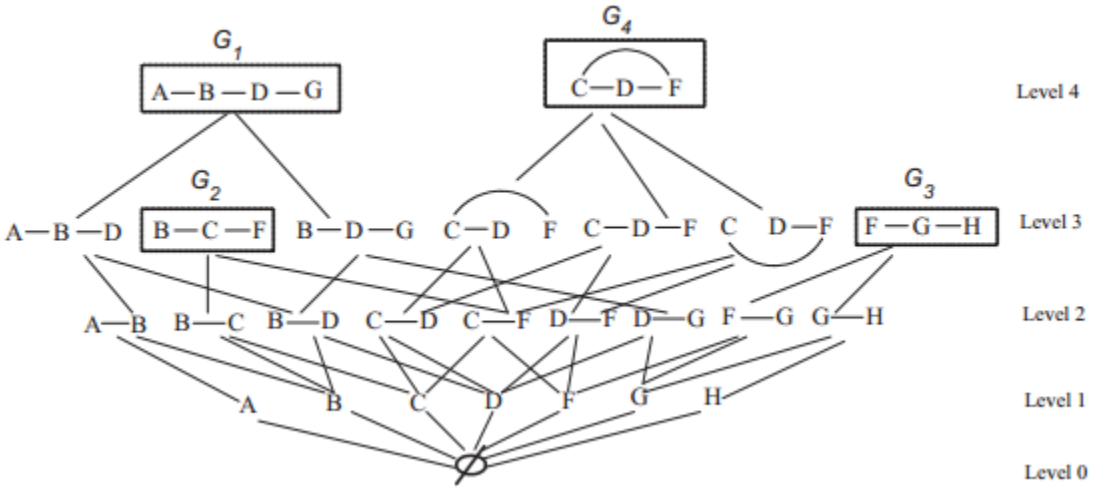
G گراف غیر مستقیم خواهد بود اگر $\forall e \in E$، که e یک جفت رأس نامنظم است. مسیر در G عبارتست از توالی رأس هایی که قابل مرتب کردن هستند به صورتی که دو رأس تشکیل یک کران را می دهند اگر و تنها اگر این دو رأس در لیست به صورت متوالی قرار گرفته باشند (وست ۲۰۰۰). G پیوسته است، اگر شامل مسیری برای هر جفت رأس موجود در آن باشد و در غیر اینصورت ناپیوسته خواهد بود. G کامل است اگر هر جفت رأس با یک کران اتصال داشته باشد و G بی حلقه گفته می شود اگر فاقد حلقه باشد.

زیر گراف: در مورد دو گراف معین $G_1(V_1,E_1,L_{V_1},L_{E_1},\varphi_1)$ و $G_2(V_2,E_2,L_{V_2},L_{E_2},\varphi_2)$، G_1 زیر گراف G_2 خواهد بود، اگر رابطه های زیر در مورد G_1 صدق می کند : (*i*) $V_1 \subseteq V_2$ و $\forall_V \in V_1, \vartheta_1(V) = \vartheta_2(V)$ (*ii*) $E_1 \subseteq E_2$ ، و $\forall (u,V) \in E_1, \vartheta_1(V,V) = \vartheta_2(u,V)$. G_1 یک گراف القاء شده G_2 خواهد بود اگر علاوه بر شرط های بالا موارد زیر نیز درباره آن صدق کند $\forall_u, V \in V_1, (u,V) \in E_1 \leftrightarrow (u,V) \in E_2$. G_2 نیز زیر گراف G_1 خواهد بود (ایوکوچی و دیرگارن ۲۰۰۲؛ هوآن ودیگران ۲۰۰۳).

هم ریختی گراف: $G_1(V_1,E_1,L_{V_1},L_{E_1},\varphi_1)$ هم ریخت گراف $G_2(V_2,E_2,L_{V_2},L_{E_2},\varphi_2)$ است اگر و تنها اگر تابع $f: V_1 \longrightarrow V_2$ وجود داشته باشد به طور که (*i*) $\forall u \in V_1, \vartheta_1(u) = \vartheta_2(f(u))$ و (*ii*) $\forall (u,V) \in E_1 \leftrightarrow (f(u),f(V))$ و(*iii*) $\forall (u,V) \in E_1, \vartheta_1(u,V) = \vartheta_2(f(u),f(V))$ تابع f یک هم ریختی بین G_1 و G_2 است. گراف G_1 زیر گراف هم ریخت گراف G_2 است اگر و تنها اگر یک گراف $g \subseteq G_2$ موجود باشد به طوری که G_1 هم ریخت g باشد (هوآن و دیگران ۲۰۰۳). در این حالت، g به عنوان تثبت کننده G_1 در G_2 تلقی می شود.

مشبک: با در نظر گرفتن پایگاه داده های ς، مشبک یک فرم ساختاری است که برای طراحی فضای جستجو برای یافتن زیر گراف های پرتکرار مورد استفاده قرار می گیرد، که در این شرایط هر رأس بیانگر یک زیر گراف پیوسته از گراف موجود در ς است (توماس و دیگران ۲۰۰۶). پایین ترین رأس، زیر گراف خالی را ترسیم می کند و رأس های بالاترین سطح معرف گراف های موجود در ς هستند. هر رأس p ما در رأس q در مشبک است، اگر q زیر گراف p باشد و q دقیقاً درک کران با p تفاوت دارد. رأس q زاده p است. تمام زیر گراف های هر گراف $G_i \in \varsigma$ که در پایگاه داده ها موجود باشند در مشبک وجود دارند و هر زیر گراف فقط یکبار در آن دیده می شود.

تصویر ۲. مشبک (ς) (تصویر بر مبنای تصویر مشابهی که در (توماس و دیگران ۲۰۰۶) ارائه شد، ترسیم شده است).



مثال: با در نظر گرفتن یک مجموعه داده گراف $\varsigma = \{G_1, G_2, G_3, G_4\}$، مشبک (ς) متناظر با آن در تصویر ۲ داده شده است. در تصویر مذکور، پایین رأس ∞ معرف زیر گراف خالی است ، و رأس های بالاترین سطح با G_1 , G_2 , G_3 , G_4 متناظرند. والدین زیر گراف های *B-D*، زیر گراف های *A-B-D* (متصل به کران *A-B*) و زیر گراف های *B-D-G* (متصل به کران *D-G*) هستند. همچنین، زیر گراف های *B-C* و *C-F* زاده های زیر گراف های *B-C-F* هستند.

گراف های درختی آزاد (مستقل): گراف غیر مستقیمی که پیوسته و فاقد حلقه است (چی و دیگران ۲۰۰۴؛ چی و دیگران *a* ۲۰۰۴)

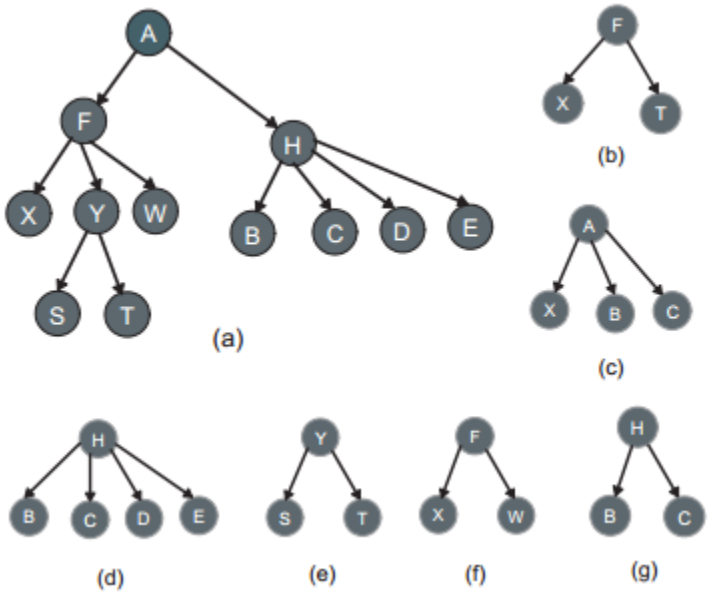
گراف درختی نامنظم برچسب دار: گراف نامنظم برچسب دار (به طور خلاصه گراف درختی نامنظم) یک گراف مستقیم فاقد حلقه است که به صورت $T\ (V, \emptyset, E, u_r)$ بیان می شود، که V مجموعه ای از رأس های T است؛ $\emptyset$ یک تابع برچسب دار است به صورتی که $\forall\ u_i \in V, \emptyset(V_i) \rightarrow V_i$؛ و $E \subseteq V \times V$ مجموعه ای از کران ها T است؛ و u_3 رأس متمایز است که ریشه T خوانده می شود. برای $\forall\ u_i \in V$، یک میسر منحصر به فرد ($u_1, u_2, \dots, u_i$) از ریشه u_r تا u_i وجود دارد. اگر رأس u_i در مسیر ریشه تا رأس u_i قرار داشته باشد، آنگاه u_i صورت قبلی u_j است، وu_j زاده u_i است. برای هر کران $(u_i, u_j) \in E$، u_i مادر u_j است و u_j زاده u_i است. رأس هایی که دارای والدین یکسان هستند خواهر نامیده میشوند. اندازه T با توجه به تعداد رأس های موجود در T تعیین می شود. هر رأس بدون زاده یک برگ است، در غیر اینصورت یک رأس میانی (رابط) است. مسیر اصلی T، مسیری است که از رأس ریشه تا برگ دست راست را در بر می گیرد. عمق (سطح) یک رأس در واقع طول مسیر از ریشه تا رأس مذکور است. مرتبه رأس u، که با مرتبه (u) مشخص می شود، تعداد کران های همراه با آن است (وست ۲۰۰۰) چی و دیگران ۲۰۰۴، چی و دیگران a ۲۰۰۴؛ آن و دیگران ۲۰۰۵).

گراف درختی منظم برچسب دار: گراف درختی منظم برچسب در یا (به طور خلاصه، گراف درختی منظم) یک گراف درختی غیر منظم برچسب دار است که دستور چپ ـ به ـ راست در میان زاده های هر رأس وضع شده است (آسای و دیگران ۲۰۰۲، آسای و دیگران ۲۰۰۳؛ چی و دیگران ۲۰۰۴).

گراف درختی فرعی *Bottom-up*: با در نظر گرفتن گراف درختی $T\ (V, \emptyset, E, u_r)$ (منظم یا غیر منظم)، $\vec{T}(\vec{V}, \vec{\emptyset}, \vec{E}, \overrightarrow{u_r})$ یک گراف درختی فرعی *bottom-up* است اگر و تنها اگر: *(i)* $\vec{V} \subseteq V$ ، *(ii)* $\vec{E} \subseteq E$، *(iii)* برچسب $\vec{V}$ و $\vec{E}$ در T در $\vec{T}$ نیز حفظ شود، *(iv)* $\forall u \in \vec{V}$ اگر $u \in \vec{V}$، آنگاه تمام زاده های u نیز می بایست در $\vec{V}$ وجود داشته باشند و *(v)* اگر T منظم باشد آنگاه دستور چپ به راست در میان رأس های خواهری موجود در T باید در $\vec{T}$ نیز محفوظ باشد (چی و دیگران ۲۰۰۴؛ والینته ۲۰۰۲).

گراف درختی فرعی القاء شده: با در نظر گرفتن گراف درختی بر چسب دار $T\ (V, \emptyset, E, u_r)$ (گراف درختی آزاد یا غیر منظم یا منظم)، $\vec{T}(\vec{V}, \vec{\emptyset}, \vec{E}, \overrightarrow{u_r})$ یک گراف درختی فرعی T است اگر و تنها اگر: (۱) $\vec{V} \subseteq V$؛ (۲) $\vec{E} \subseteq E$؛ (۳) بر چسب $\vec{V}$ و $\vec{E}$ متعلق به T در $\vec{T}$ نیز محفوظ باشد؛ (۴) اگر برای گراف درختی منظم بیان شوند، دستور چپ به راست میان خواهرها در $\vec{T}$ باید ترتیب فرعی متناظر با رأس های T باشد (چی و دیگران ۲۰۰۴؛ تان و دیگران ۲۰۰۶)

گراف درختی تثبیت شده: با در نظر گرفتن گراف درختی بر چسب دار $T\ (V, \emptyset, E, u_r)$، $\vec{T}(\vec{V}, \vec{\emptyset}, \vec{E}, \overrightarrow{u_r})$ یک گراف درختی فرعی T است، اگر و تنها اگر: $(i)\ \vec{V} \subseteq V$، $(ii)\ \forall u \in \vec{V}$، $\vec{\emptyset}(u)$، $(iii)\ \forall (u,v) \in \vec{E}$ به طوری که u مادر v باشد، u شکل قبلی v در T است و (iv) در صورت وجود گراف های درختی منظم $\forall (u,v) \in \vec{V}$، پیش دستور (u) > پیش (v) در $\vec{T}$ اگر و تنها اگر پیش دستور (u) > پیش دستور (V) در T، که پیش دستور یک رأس در واقع شاخص آن در گراف درختی برحسب عبور پیش دستور است.



تصویر ۳. انواع مختلف گراف های درختی.

جدول ۲ طبقه بندی تناظر دقیق الگوریتم های هم ریختی (زیر) گراف

الگوریتم ها	تکنیک های اصلی	انواع تناظر
اولمان	عقب نشینی + تابع برنامه ریزی	هم ریختی گراف و زیر گراف
SD	ماتکریس فاصله + عقب نشینی	هم ریختی گراف
نوتی	تنوری گروه + برچسب گذاری مجاز	هم ریختی گراف
VF	استراتژی *DFS* + قوانین احتمالی	هم ریختی گراف و زیر گراف
VF_2	مبنای منطقی VF_2 + ساختارهای پیشرفته داده ها	هم ریختی گراف و زیر گراف

تصویر ۳ برای خلاصه کردن جدول بالا، مثال هایی از گراف های درختی فرعی *bottom-up*، گراف های درختی فرعی القاء شده و گراف های درختی فرعی تثبیت شده ارائه می دهد. در تصویر مذکور، گراف درختی (a) نشاندهنده یک گراف درختی داده ها است، گراف های درختی (d) و (e) دو گراف درختی فرعی از گراف (a) هستند، گراف های درختی (f) و (g) دو گراف درختی فرعی القاء شده (a) هستند و گراف درختی (b) و (c) دو گراف درختی فرعی تثبیت شده (a) هستند. رابطه بین این ۳ نوع گراف درختی فرعی را می توان به این شکل بیان کرد: گراف درختی فرعی تثبیت شده $\leq$ گراف درختی فرعی القاء شده $\leq$ گراف درختی فرعی *bottom-up*

۲,۲. ارزیابی هم ریختی گراف

هسته اصلی *FSM*، ارزیابی هم ریختی (زیر) گراف است. به نظر نمی رسد هم ریختی گراف قابل حل در زمان چند بعدی یا *NP-compelete* باشد، در حالی که هم ریختی زیر گراف؛ که مایل به تثبیت آن هستیم فارغ از اینکه این زیر گراف در ابر گراف موجود باشد یا نه، چند بعدی است (گاری و جانسون ۱۹۷۹). وقتی گراف ها را به گراف های درختی محدود می کنیم، ارزیابی هم ریختی (زیر) گراف تبدیل به ارزیابی هم ریختی گراف درختی فرعی می شود. ارزیابی هم ریختی گراف درختی در زمان تک بعدی (خطی) قابل حل است (الگوریتم پیشنهادی در هاپ گرافت و تارجان ۱۹۷۲ را ببینید). الگوریتم های سریع تر در زمینه ارزیابی هم ریختی گراف های درختی فرعی، با دشواری بدترین حالت زمان ($O(k^{1/5}n)$)، توسط ماتولا (۱۹۷۸) و چانگ (۱۹۸۷) پیشنهاد شد، و پس از آن توسط شامیر و تی سور (۱۹۹۹) به صورت $O(\frac{k^{\frac{1}{5}}}{10gk}n)$ ارتقاء یافت (k و n معرف اندازه گراف درختی فرعی و گراف درختی که باید از لحاظ تعداد رأس ها مورد جستجو قرار بگیرند، هستند).

ارزیابی هم ریختی زیر گراف برای *FSM* بسیار حیاتی است. تعداد قابل توجهی از تکنیک های کارآمد پیشنهاد شده اند که همگی بر کاهش محاسبات وابسته به ارزیابی هم ریختی زیر گراف تمرکز کرده اند. تکنیک های ارزیابی هم ریختی زیر گراف را به سختی می توان در مقوله "تناظر دقیق" قرار داد (اولمان ۱۹۷۶، اشمیت و دروفل ۱۹۷۶؛ مک کی ۱۹۸۱؛ کوردلا و دیگران ۱۹۹۸؛ کوردلا و دیگران ۲۰۰۱)ا در مقوله تناظر خطای مجاز گنجاند (شاپیرو و هارالیک ۱۹۸۱؛ بونک و آلرمن ۱۹۸۳؛ کریسس و دیگران ۱۹۹۵؛ مسمر و بونک ۱۹۹۸). اکثر الگوریتم های *FSM* از تناظر دقیق استفاده می کنند. طبقه بندی مهم ترین الگوریتم های تناظر دقیق برای ارزیابی هم ریختی زیر گراف در جدول ۲ داده شده است. در جدول ۲، ستون ۲ نشان دهنده مهم ترین

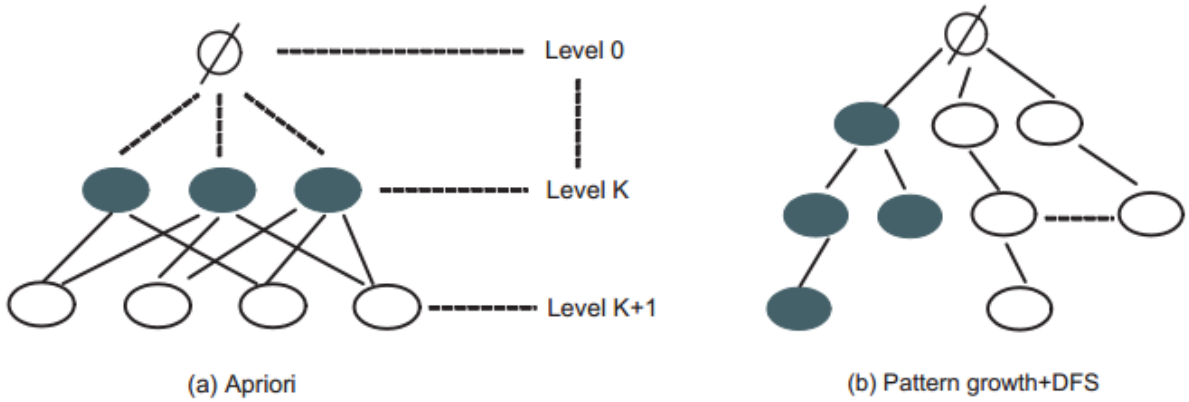
روش های به کارگرفته شده برای اجرای ارزیابی هم ریختی است، و ستون ۳ نشان می دهد که الگوریتم ارزیابی هم ریختی برای گراف اعمال شده است یا زیر گراف.

با مراجعه به جدول ۲، الگوریتم اولمان از یک پروسه عقب نشینی با تابع برنامه ریزی برای کاهش اندازه فضای جستجو استفاده می کند (ارلمان ۱۹۷۶). الگوریتم *SD*، در عوض، از یک ماتریکس فاصله معرف یک گراف به همراه پروسه عقب نشینی برای کاهش جستجو بهره می گیرد (اسمیت و دروفل ۱۹۷۶). الگوریتم نوتی (مک کی ۱۹۸۱) از تئوری گروه برای تغییر شکل و تبدیل گراف ها برای تطابق با فرم مجاز استفاده می کند به شکلی گراف ها برای بررسی مؤثرتر و کارآمدتر هم ریختی گراف آماده شوند. با این وجود، گفته شده است (کونته و دیگران ۲۰۰۴) که ساخت اشکال و فرم های مجاز می تواند منجر به پیچیدگی تصاعدی در بدترین حالت آن شود. اگر چه الگوریتم نوتی توسط کونته و دیگران (۲۰۰۴) به عنوان سریع ترین الگوریتم هم ریختی گراف شناخته شد، میازاکی (۱۹۹۷) نشان داد که بعضی از ویژگی های گراف ها نیازمند زمان فزاینده ای برای ایجاد برچسب مجاز هستند. الگوریتم های *VF* (کوردلا و دیگران ۱۹۹۸) و *VF₂* (کوردلا و دیگران ۲۰۰۱) از استراتژی جستجوی عمق ـ محور استفاده می کنند که مجموعه ای از قوانین احتمال برای هرس کردن گراف درختی جستجو به آن کمک می کند. *VF₂* نسخه اصلاح شده *VF* است که فضای جستجو را با کارایی بیشتری بررسی می کند به طوری که زمان تطابق و تناظر و محاسبات حافظه تا حد قابل ملاحظه ای کاهش می یابد. در فوگیا و دیگران (۲۰۰۱) تحلیل تجربی همه جانبه ای از این پنج الگوریتم ارائه شده است که نشان می دهد هیچ یک از الگوریتم های موجود برتری مطلقی بر دیگری ندارد. به طور کلی، اثبات شده که *VF₂* با توجه به اندازه و نوع گراف هایی که باید متناظر شوند بهترین عملکرد را نشان داده است.

۳. ارزیابی FSM

این بخش یک ارزیابی کلی از فرآیند *FSM* (استخراج زیر نمدارهای پرتکرار) ارائه می دهد. این نکته به طور گسترده ای پذیرفته شده است که تکنیک های *FMS* را میتوان به دو دسته تقسیم کرد. (*i*) رویکردهای آپریوری ـ محور (بر مبنای آپریوری) و، (*ii*) رویکردهای تعمیمی الگو. این دو دسته از لحاظ ماهیتی به نمونه های یافت شده در استخراج قوانین وابسته (*ARM*)، یعنی الگوریتم آپریوری (آکراوان و اسرکانیت ۱۹۹۴) و الگوریتم تعمیم *FP* (الگوی پرتکرار) شباهت دارد (همان و دیگران ۲۰۰۰). رویکرد آپریوری ـ محور در شیوه تولید و آزمایش با استفاده از استراتژی جستجوی گستره ـ محور (*BFS*) به منظور بررسی مشبک زیر گراف پایگاه داده معین موفق عمل می کند. از این رو، پیش از در نظر گرفتن (*K*+۱) زیر گراف، این رویکرد در ابتدا

باید تمام زیر گراف های *K* را بررسی کند. الگوریتم تعمیمی الگو از استراتژی *DFS* استفاده می کند که برای هر زیر گراف مشخص *g*، زیر گراف به تناوب تعمیم پیدا می کند تا جایی که تمام زیر گراف های پرتکرار *g* مشخص شوند (هان و کابر ۲۰۰۶). تفکیک بین دو رویکرد در تصویر ۴ نشان داده شده است.



(a) Apriori (b) Pattern growth+DFS

تصویر ۴ دو نوع فضای جستجو، توجه داشته باشید که مشبک زیر گراف به طور وارونه نشان داده شده است. رأس های متناظر به گراف های دارای کران های اندک در بالای تصویر نشان داده شده اند.

Algorithm 3.1: Apriori-based approach

Input: $\mathcal{G}$ = a graph data set, σ= minimum support
Output: $F_1, F_2, \cdots, F_k$, a set of frequent subgraphs of cardinality 1 to k

```
1  F_1 ← detect all frequent 1 subgraphs in G
2  k ← 2
3  while F_{k-1} ≠ ∅ do
4      F_k ← ∅
5      C_k ← candidate-gen(F_{k-1})
6      foreach candidate g ∈ C_k do
7          g.count ← 0
8          foreach G_i ∈ G do
9              if subgraph-isomorphism(g, G_i) then
10                 g.count ← g.count + 1
11             end
12         end
13         if g.count ≥ σ|G| ∧ g ∉ F_k then
14             F_k = F_k ∪ g
15         end
16     end
17     k ← k + 1
18 end
```

الگوریتم آپریوری ـ محور اصلی در ۳,۱ نشان داده شده است. در خط ۵ تمام زیر گراف های (*K*ـ۱) پرتکرار برای تولید داوطلب های زیرگراف *K* مورد استفاده قرار گرفته اند. اگر هر یک از زیر گراف های داوطلب *K*ـ۱ پرتکرار نباشند آنگاه از *DCP* (به بخش ۲ مراجعه کنید) می توان برای حذف مطمئن داوطلب ها استفاده کرد.

اکثر رویکردهای *FSM* موجود از یک استراتژی تناوبی (تکراری) برای استخراج الگو استفاده می کنند که در آن هر یک از تناوب را میتوان به ۲ فاز تقسیم کرد: (*i*) ایجاد داوطلب (خط ۵ در الگوریتم ۳,۱) و (*ii*) محاسبه پشتیبان (خطوط ۱۲ـ۶ الگوریتم ۳,۱). به طور کلی، تحقیقات حوزه *FSM* بر این دو فاز که از تکنیک های گوناگونی بهره می گیرند، تمرکز می کنند. از آنجایی که پرداختن به ارزیابی هم ریختی زیر گراف دشوارتر است، بیشتر تحقیقات چگونگی ایجاد کارآمد داوطلب های زیر گراف را نشانه گرفته اند. چون ارزیابی هم ریختی گراف های درختی فرعی را می توان در زمان $O(\frac{k^{\frac{1}{5}}}{10gk}n)$ حل کرد، دشواری محاسباتی در شرایط *FSM* کاهش پیدا می کند. از این رو، ارزیابی ارائه شده در این مقاله بین استخراج زیر گراف پرتکرار و استخراج گراف درختی فرعی پرتکرار تمایز قائل می شود. در بقیه این مقاله، به استفاده از واژه اختصاری *FSM* برای هر دو حزه استخراج زیر گراف و گراف درختی فرعی پرتکرار ادامه خواهیم داد؛ و از واژه های اختصاری *FGM* و *FTM* هرجا که نیاز به تفکیک حوزه ها باشد به ترتیب برای اشاره به استخراج زیر گراف و استخراج گراف درختی فرعی پرتکرار استفاده خواهیم کرد.

پیش از بررسی مشروح الگوریتم های خاص استخراج زیر گراف و گراف درختی فرعی (بخش های ۴و۵)، ابتدا تکنیک های بازنمایی گراف ها و گراف های درختی را مورد بررسی قرار خواهیم داد. هدف از این کار بازنمایی گراف ها و گراف های درختی به صورتی است که بتوان زیر گراف ها را برای سهولت *FSM* مطلوب مشخص کرد.

۳,۱. بازنمایی های مجاز

ساده ترین مکانیسمی که از طریق آن ساختار گراف قابل بازنمایی (ارائه) است، استفاده از ماتریکس تجانب یا فهرست تجانب است. با استفاده از ماتریکس تجانب، ردیف ها و ستون ها نشان دهنده رأس ها هستند، و محل تلاقی ردیف *i* و ستون *j* نشان دهنده کران بالقوه ای است که رأس های V_i و V_j را به هم متصل می کند مقدار قرار گرفته در تلاقی « *I,j*» به طور معمول نشان دهنده تعداد لینک های V_i و V_j است. با این وجود، استفاده از ماتریکس تجانب به درد ارزیابی هم ریختی نمی خورد، زیرا بسته به اینکه رأس ها (و کران ها) چگونه مشخص شوند می توان گراف ها را به شیوه های مختلفی ارائه کرد (واشیو و موتورا ۲۰۰۳). با توجه به آزمایش هم ریختی، انتخاذ یک استراتژی بر چسب گذاری یکپارچه مطلوب است که ضمانت کند هر دو گراف

مشابه فارغ از ترتیب ارائه رأس ها و کران ها به شیوه ای یکسان برچسب گذاری شوند (یعنی استراتژی برچسب گذاری مجاز)

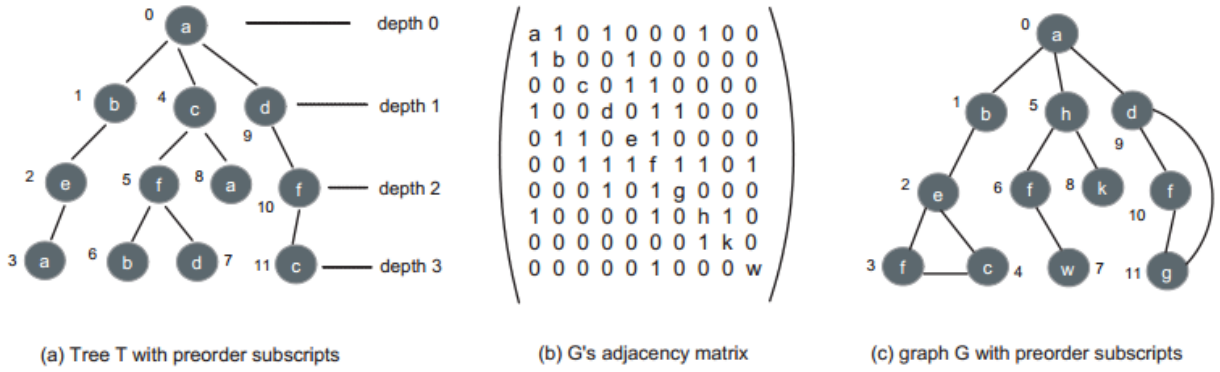
استراتژی برچسب گذاری مجاز یک کد منحصر بفرد برای هر گراف تعیین می کند (رید و کورینل ۱۹۷۷؛ فورتین ۱۹۹۶). بر چسب گذاری مجاز موجب سهولت بررسی هم ریختی می شود زیرا تضمین می کند که اگر یک جفت گراف هم ریخت باشند آنگاه بر چسب گذاری مجاز آنها نیز یکسان باشد (کوراموشی و کاریپیس ۲۰۰۱). روش ساده برای ایجاد برچسب مجاز، باز کردن ماتریکس تجانب مربوطه با ردیف ها یا ستون های به هم پیوسته برای تولید یک کد متشکل از فهرستی از اعداد صحیح با ترتیب واژگان نمایی مینیمم (یا ماکسیمم) است. برای کاهش بیشتر محاسبات حاصل از جایگشت های ماتریکس، بر چسب گذاری مجاز معمولاً با استفاده از طرح نامتغیر رأس فشرده می شود. (رید و کونیل ۱۹۷۷) که اجازه می دهد محتوای ماتریکس بر طبق بر چسب های رأس تقسیم بندی شود. طرح های مختلف بر چسب گذاری مجاز [تا کنون] پیشنهاد شده اند که بعضی از مهم ترین موارد در این زیر بخش شرح داده شده اند.

کد *DFS* مینیمم (*M-DFSC*): چندین شیوه گوناگون کد گذاری *DFS* وجود دارد، اما لزوماً هر یک از کران های سازنده یک گراف در کد *DFS* به وسیله ۵ ـ وجهی (i,j,L_i,L_e,L_j) بازنمایی می شود، که i و j شناسه های رأس هستند، L_i و L_j بر چسب هایی برای رأس های متناظرند و L_e بر چسب کران متصل کننده رأس هاست. بر اساس ترتیب واژگان نهایی *DFS*، *M-DFSC* هر گراف g را می توان به صورتک برچسب مجاز g تعریف کرد (ان و هان ۲۰۰۲). کدهای *DFS* برای دست چپی ترین شاخه و دست راستی ترین شاخه گراف ارائه شده در تصویر (c)۵ به تریب { (b، ۱، a، ۱، ۰)، (e، ۱، b، ۲، ۱)، (f، ۱، e،۲،۳)، (c، ۱، f، ۴، ۳)، (e، ۱، c،۲، ۴) } و { (d، ۱، a، ۹، ۰)، (f، ۱،، ۱۰، ۹)، (g، ۱، F، ۱۱، ۱۰)، (d، ۱، g،۹، ۱۱) } هستند.

ماتریکس تجانب مجاز (*CAM*): با در نظر گرفتن تجانب M از گراف g، کدگذاری M را می توان به وسیله توالی به دست آمده از تسلسل ورودی های مثلثی پایین تر یا بالاتر M شامل ورودی های قطرها به دست آورد. از آنجایی که جایگشت های مختلف مجموعه رأس ها با ماتریکس های تجانب مختلف متناظر است، فرم مجاز (*CAM*) g به صورت کد گذاری ماکزیمم (مینیمم) تعیین می شود. ماتریکس تجانبی که فرم مجاز از روی آن ایجاد می شود ماتریکس تجانب مجازا *CAM* را تعیین می کند. (اینوکچی و دیگران ۲۰۰۰، ۲۰۰۲، کوراموشی و کاریپیس ۲۰۰۱؛ هوان و دیگران ۲۰۰۳). کد گذاری گراف که در تصویر (c)۵ داده شده که توسط ماتریکس گراف (ط)۵ بازنمایی می شود، به این صورت است:

$b \cdot\cdot c \setminus \cdot\cdot d \cdot \setminus\setminus \cdot e \cdot\cdot \setminus\setminus\setminus f \cdot\cdot\cdot \setminus \cdot \setminus g \setminus \cdot\cdot\cdot\cdot \setminus \cdot h \cdot\cdot\cdot \cdot\cdot\cdot\cdot \setminus k \cdot\cdot\cdot\cdot\cdot \setminus \cdot\cdot\cdot w \}$

$\{ a\ 1$



(a) Tree T with preorder subscripts (b) G's adjacency matrix (c) graph G with preorder subscripts

دو طرح بالا برای هر گراف ساده غیر مستقیمی قابل اجراست. با این وجود، تعیین برچسب گذاری مجاز برای گراف های درختی ساده تر از گراف هاست چون گراف های درختی ساختاری ذاتی به همراه خود دارند. طرح های خاص بیشتری نیز وجود دارند که منحصراً بر گراف های درختی تمرکز می کنند. از میان این طرح ها، *DFS – LS* و *DLS* به گراف های درختی منظم ریشه ای می پردازند، *BFCS* و *DFCS* برای گراف های درختی نامنظم ریشه ای به کار می روند. هر یک از این طرح را به طور خلاصه در زیر توضیح خواهیم داد.

توالی برچسب *DFS* (*DFS – LS*): با در نظر گرفتن یک گراف درختی منظم برچسب دار *T*، بر چسب هایی $\forall V_i \in V$ در خلال ایجاد برش *DFS* در *T* به زنجیره *S* اضافه می شوند. هرگاه فرآیند عقب نشینی اتفاق بیفتد یک سمبل منحصر بفرد مانند "۱ـ"ا "S" یا "/" به زنجیره *S* اضافه می شود (زاکی ۲۰۰۲؛ زاکی *a* ۲۰۰۵؛ آن و دیگران ۲۰۰۶). کد *DFS-LS* برای گراف درختی داده شده در تصویر (*a*) ۵ به این صورت است:

$\{$ *abea$$$cfb$d$$a$$dfc$$$* $\}$

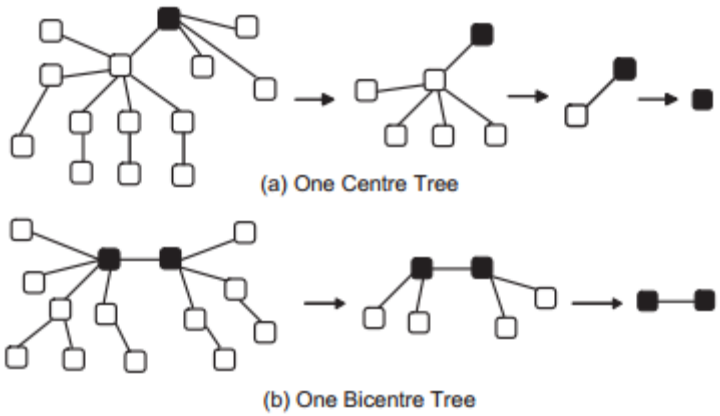
توالی عمق ـ برچسب (*DLS*): با در نظر گرفتن گراف درختی منظم برچسب دار *T*، جفت های عمق ـ برچسب متشکل از $\forall V_i \in V$، ($d(V_i)$, $L(V_i)$) در خلال ایجاد برش *DFS* در *T*، به زنجیره *S* اضافه می شوند. توالی عمق ـ بر چسب *T* به صورت $\{ (d(V_1)$ و $L(V_1))$ و... و$(d(V_k)$ ،$L(V_k)) \}$ تعریف می شود. (آسای و دیگران ۲۰۰۲، وانگ و دیگران *a* ۲۰۰۴). کد *DLS* برای گراف درختی ارائه شده در تصویر (*a*)۵ به این صورت است:

$\{$ (*c* و۳) و (*f* و۲) و (*d* و ۱) و (*a* و ۲) و (*d* و ۳) و (*b* و ۳) و (*f* و ۲) و (*c* و۱) و (*a* و ۳) و (*e* و۲)و(*b*و۱) و (*a* و۰) $\}$

زنجیره مجاز گستره ـ محور (*BFCS*): برای گراف درختی منظم بر چسب دار، هر یک از برچسب رأس از طریق قطع گراف درختی به روش *BFS*، به زنجیره *S* اضافه می شود. علاوه براین. نماد "*S*" برای تقسیم بندی گروه های خواهرها و نماد "#" برای اشاره به خاتمه کد گذاری زنجیره مورد استفاده قرار می گیرد "*S*" از نظر واژگان نمایی پیش از "#" در نظر گرفته می شود و هر دوی آنها بزرگتر از سایر برچسب های رأس ها و کران ها مرتب می شوند. با در نظر گرفتن گراف درختی نامنظم *T*، گراف های درختی منظم مختلف با کد گذاری زنجیره *BFS* مشابه با تحمیل قاعده ها و ترتیب های مختلف به زاده های رأس های میانی قابل تولید خواهند بود. *BFCS* گراف *T* از نظر واژگان نمایی مینمیم این کد گذاری هاست و گراف درختی منظم ریشه ای متناظر می تواند فرم مجاز گستره ـ محور *T* را تعیین کند (چی و دیگران ۲۰۰۵). گونه های مختلف *BFCS* را می توان در چی و دیگران (*c* ۲۰۰۴ و ۲۰۰۳) ملاحظه کرد. بنابراین، کد گذاری زنجیره *BFS* از گراف درختی نمونه در تصویر (*a*)۵ به این شکل است: *abcdefaf$a$bd$$c#*

زنجیره مجاز عمق ـ محور (*DFCS*): همانند *BFCS* است با این تفاوت که از *DFS* استفاده می کند کد گذاری زنجیره عمق ـ محور برای گراف درختی منظم بر چسب دار هر یک از رأس ها را با قطع گراف درختی به شیوه *DFS* برچسب می زند. آنگاه *DFCS* گراف درختی منظم ریشه ای متناظر، فرم مجاز عمق ـ محور (*DFCF*) *T* را تعیین می کند (چی و دیگران ۲۰۰۵). شکل های مختلف *DFCS* را نیز می توان در چی و دیگران (ط ۲۰۰۴ و ۲۰۰۳) مشاهده کرد. کد گذاری زنجیره ای *DFS* گراف درختی نمونه در تصویر (*a*) ۵ به صورت زیر است:

abea$$$cfb$d$$dfc$$$#



تصویر ۶: نمونه از دو نوع گراف درختی آزاد (مستقل)

بازنمایی مجاز گراف های درختی آزاد (مستقل): گراف های درختی فاقد ریشه هستند. در این شرایط، یک بازنمایی منحصر بفرد برای گراف درختی آزاد معمولاً با انتخاب یک رأس یا یک جفت رأس به عنوان ریشه (ها) طراحی می شود. این پروسه با حذف تمام رأس های برگ ها و کران های همراه آن آغاز می شود و تا جایی ادامه پیدا می کند که یک رأس مجزا یا دو رأس مجانب باقی بماند. در حالت اول، رأس باقیمانده به عنوان مرکز خوانده می شود، و گراف درختی نامنظم ریشه ای با مرکز به عنوان ریشه به دست می آید. پروسه در تصویر ۶(*a*) نمایش داده شده است. در حالت دوم، جفت رأس باقیمانده جفت ـ مرکز خوانده می شوند؛ یک جفت گراف درختی نامنظم ریشه ای با جفت ـ مرکز به عنوان ریشه به دست می آید (همراه با یک کران که دو ریشه را به هم متصل می کند). این پروسه در تصویر (*b*) ۶ نشان داده شده است. یک جفت گراف درختی مرتب می شوند به طوری که ریشه گراف کوچکتر به عنوان ریشه کل گراف درختی انتخاب می شود (چی و دیگران ۲۰۰۳، روکرت و کرامر ۲۰۰۴) پس از به دست آوردن گراف های درختی نامنظم ریشه ای، از هرگونه بازنمایی مجاز برای گراف درختی نامنظم ریشه ای می توان برای بازنمایی گراف های درختی آزاد استفاده کرد.

۳,۲. ایجاد داوطلب

به گونه ای که پیشتر گفته شد، ایجاد داوطلب مرحله ای لازم و اساسی در FSM است. چگونگی ایجاد قاعده مند زیر گراف های داوطلب بدون زوائد (یعنی هر زیر گراف فقط یکبار باید ایجاد شود) موضوعی کلیدی است. بسیاری از الگوریتم های FSM را می توان با توجه به استراتژی اتخاذ شده برای ایجاد داوطلب طبقه بندی کرد. یعنی از مهم ترین استراتژی ها به طور خلاصه در ادامه شرح داده شده اند. از آنجایی که بخش قابل ملاحظه ای از استراتژی های به خدمت گرفته در FTM با نمونه های مورد استفاده در FGM در هم آمیخته اند، نمی توان تمایز آشکاری بین استراتژی های ایجاد داوطلب در FTM و FGM قائل شد؛ یعنی استراتژی هایی که در ابتدا برای FGM پیشنهاد شدند به طورکسانی برای FTM نیز قابل اجرا هستند و برعکس.

۳,۲,۱.اتصال سطح ــ محور

استراتژی اتصال سطح ـ محور توسط کوراموشی و کاریپیس (۲۰۰۱) معرفی شد. اساساً، داوطلب زیر گراف (K+۱) از طریق ترکیب دو زیر گراف پرتکرار K که زیر گراف (K-۱) کسانی را به اشتراک گذاشته اند ایجاد می

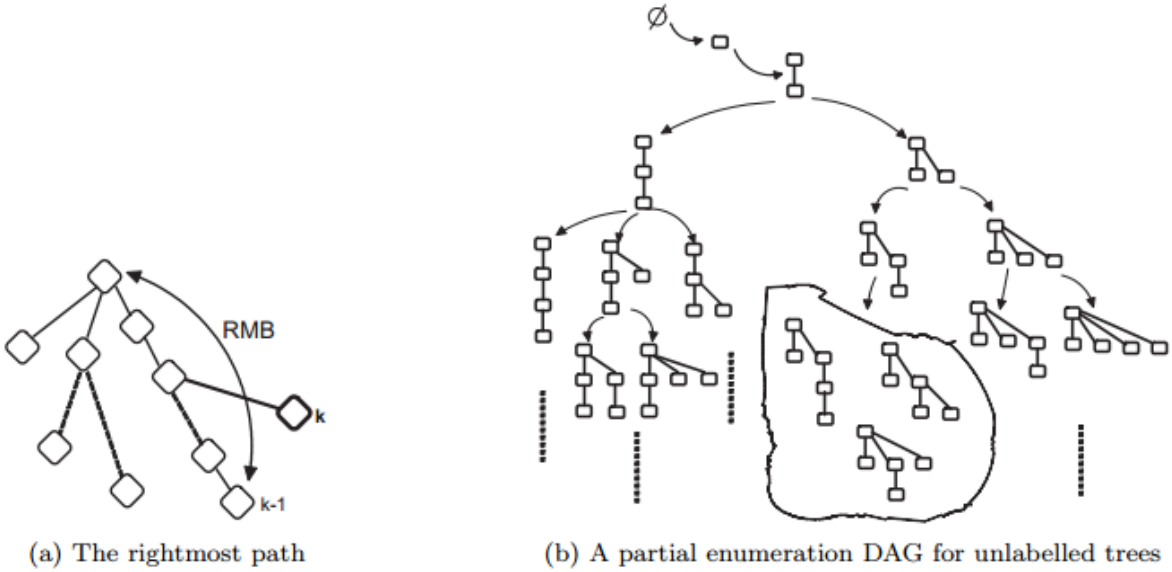
شود. این زیر گراف (K_۱) متعارف به عنوان مرکزی برای این دو زیر گراف پرتکرار K تلقی می شود. موضوع اصلی در مورد این استراتژی این است که یک زیر گراف K می تواند حداکثر K زیر گراف (K_۱) مختلف داشته باشد و عملیات اتصال ممکن است تعداد زیادی داوطلب زائد ایجاد کند. در کوراموشی و کاریپیس (a ۲۰۰۴)، این موضوع با محدود کردن زیر گراف های (K_۱) به دو زیر گراف (K_۱) دارای کوچکترین و دومین کوچکترین برچسب های مجاز برطرف شد. با اجرای این عملیات اتصال تعدیل شده، تعداد داوطلب های کپی ایجاد شده به طور چشمگیری کاهش یافت، سایر الگوریتم هایی که از این استراتژی بهره می گیرند و شکل های مختلف آن AGM است (اینوکوچی و دیگران ۲۰۰۰) عبارتند از: Dpmine (وانتیک و دیگران ۲۰۰۲؛ گورس و دیگران ۲۰۰۶)، و HSIGRAM (کوراموشی و کاریپیس ۲۰۰۵) که در ادامه مورد بررسی قرار خواهند گرفت.

۳,۲,۲. تعمیم دست راستی ترین مسیر

تعمیم دست راستی ترین مسیر متداول ترین استراتژی ایجاد داوطلب است؛ این استراتژی با اضافه کردن رأس ها به دست راستی ترین مسیر گراف درختی، تعداد (k+۱) گراف درختی فرعی از گراف درختی فرعی k پرتکرار ایجاد می کند (آسای و دیگران ۲۰۰۲؛ زاکی ۲۰۰۲؛ آسای و دیگران ۲۰۰۳؛ نیجن و کوک ۲۰۰۳). در تصویر ۷ (a)، "RMB" بیانگر دست راستی ترین شاخه است که مسیری از ریشه تا دست راستی ترین برگ (K_۱) است و یک رأس جدید K با ملحق کردن آن به هر یک از رأس های موجود در راستای RMB اضافه می شود.

DAG (گراف فاقد حلقه مستقیم) شمارشگر با استفاده از تعمیم دست راستی ترین مسیر، یک گراف درختی با Ø ریشه است که هر گره یک الگوی گراف درختی فرعی است. گره S به گره T متصل است اگر و تنها اگر T تعمیم دست راستی ترین مسیر S باشد. هر زیر گراف ـ۱ تعمیم دست راستی ترین ریشه Ø است و هر گراف درختی فرعی ـ (K+۱) تعمیم دست راستی ترین مسیر گراف درختی فرعی ـ K است. به همین خاطر تمام الگوهای گراف درختی را می توان با قطع به روش BFS یا DFS تعیین کرد (آسای و دیگران ۲۰۰۲). تصویر ۷ (b) قسمتی از DAG شمارشگر که با استفاده از تعمیم دست راستی ترین مسیر گسترش یافته است را نشان می دهد. هر مربع در تصویر ۷(b) نشاندهنده یک رأس در گراف درختی است. DAG شمارشی (که گاهی اوقات به صورت گراف درختی شمارش خوانده می شود) برای بیان چگونگی تعیین مجموعه ای از الگوها درک فرآیند جستجو مورد استفاده قرار می گیرد. DAG های شمارشی به طور گسترده در استخراج قوانین وابسته مورد استفاده قرار گرفته (باردو ۱۹۹۸؛ آگراوان و دیگران ۲۰۰۱)؛ و به دنبال آن به شیوه های مختلفی در

اکثر الگوریتم های استخراج گراف های درختی فرعی به کار گرفته شده اند (آسای و دیگران ۲۰۰۲، نیجن و کوک ۲۰۰۳؛ آسای و دیگران ۲۰۰۳؛ چی و دیگران a ۲۰۰۴؛ چی و دیگران ۲۰۰۵).



(a) The rightmost path

(b) A partial enumeration DAG for unlabelled trees

تصویر ۷. مثالی از تعمیم دست راستی ترین مسیر

۳,۲,۳. امتداد و اتصال

استراتژی امتداد و اتصال نخستین بار توسط هوآن و دیگران (۲۰۰۳) معرفی شد و پس از آن توسط چی و دیگران (۲۰۰۴a) مورد استفاده قرار گرفت. این استراتژی از بازنمایی BFCS بهره می گیرد؛ که در آن یک برگ در سطح زیرین گراف درختی BFCF به عنوان ساق (پایه) در نظ گرفته می شود. برای گره "V_n " درک گراف درختی شمارشگر، اگر ارتفاع گراف درختی BFCF متناظر با "V_n " را h در نظر بگیریم، تمام زاده های "V_n " را می توان از طریق یکی از دو عملیات زیر به دست آورد:

(a) عملیات امتداد: اضافه کردن یک ساق جدید به سطح پایینی گراف درختی BFCF به یک BFCF جدید با ارتفاع h+۱ نیاز دارد.

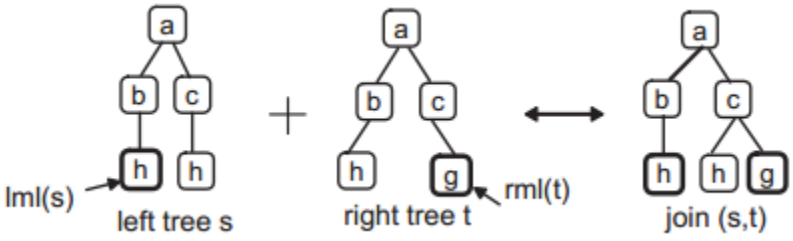
(b) عملیات اتصال: اتصال "V_n " و یکی از خواهرهای آن به یک BFCF جدید با ارتفاع h نیاز دارد.

۳,۲,۴. امتداد طبقه ـ محور معادل (هم ارز)

امتداد طبقه ـ محور هم ارز (زاکی ۲۰۰۲، ۲۰۰۵) بر اساس بازنمایی DFS-LS برای گراف های درختی طراحی شد. اساساً، گراف درختی فرعی ـ k+۱ با اتصال دو گراف درختی فرعی ـk پرتکرار ایجاد می شود. این دو گراف درختی فرعی ـ k باید در طبقه هم ارزی مشابه [C] باشند. تمام طبقه های هم ارز شامل کد گذاری پیشوند طبقه و فهرستی از اعضا هستند. هر یک از اعضاء طبقه را می توان به صورت جفت (L,p) بیان کرده که L برچسب رأس k ام است و P موقعیت عمقی والدین رأس k ام است. در زاکی (۲۰۰۲) تأیید شده است که تمام گراف های درختی (K+۱) با پیشوند [C] و اندازه (K_۱) را می توان با اتصال هر جفت از اعضای دارای طبقه هم ارز مشابه [C] ایجاد کرد.

۳,۲,۵. اتصال گراف درختی راست ـ و ـ چپ

استراتژی اتصال گراف درختی راست و چپ توسط هیدو و کوآوانو (۲۰۰۵) معرفی شد. این استراتژی اساساً از دست راستی ترین برگ (۲,۱ را ببینید) و دست چپی ترین برگ گراف درختی برای ایجاد داوطلب هایی به شیوه BFS ایجاد می کند. بگذارید LmL (T) بیانگر دست چپی ترین برگ T و Right T بیانگر دست راستی ترین گراف درختی به دست آمده حاصل از حذف LmL (T) باشد؛ و بگذارید rmL (T) نشان دهنده دست راستی ترین برگ و Left(T) بیانگر دست چپی ترین گراف درختی به دست آمده حاصل از حذف rmL(T) باشد. با در نظر گرفتن دو گراف درختی s و t که Right (s)= Left (s)، گراف درختی راست و چپ آنها به این صورت تعریف می شود: Join (s.t) = s □ rmL (t) = LmL (s) □t . شکلی که این عملیات اتصال را توضیح میدهد در تصویر ۸ داده شده است.



تصویر ۸: مثالی از اتصال گراف درختی راست و چپ

در میان این استراتژی های ایجاد داوطلب، اتصال سطح ـ محور و امتداد و اتصال بر FGM تمرکز کرده اند و سایر استراتژی ها همگی به FTM پرداخته اند.

۴. الگوریتم های استخراج گراف های درختی فرعی پرتکرار

بخش قبلی به موضوعات مربوط به بازنمایی (فرم های مجاز) و ایجاد داوطلب در چارچوب گراف های درختی و گراف ها پرداخته بود. در این بخش، بعضی از الگوریتم های مهم مورد بررسی قرار می گیرند. FTM توجه فراوانی را در حوزه هایی مانند: ارسال چند کیفیتی IP شبکه (چی و دیگران ۲۰۰۵) ، استخراج کاربرد شبکه (زاکی ط ۲۰۰۵) ، نسخه کامپیوتر (لیو و گیجر ۱۹۹۹) ، استخراج XML (زاکی و اگراول ۲۰۰۳؛ آن و دیگران ۲۰۰۵) ، بیوانفوماتیک (هین و دیگران، ۱۹۹۶؛ روکرت و کرامر ۲۰۰۴؛ ژانگ و وانگ ۲۰۰۶) و غیره. گرایش استخراج گراف درختی فرعی پرتکرار این است که ارزیابی هم ریختی زیر گراف تبدیل به ارزیابی هم ریختی گراف درختی فرعی است، که در زمان $O(\frac{K^{\frac{1}{5}}}{10gk}n)$ قابل حل باشد (شامیر و تی سور ۱۹۹۹). علاوه بر این، ساختار گراف های درختی ممکن است به طور مؤثری برای ساده کردن فرآیند استخراج به خدمت گرفته شود.

الگوریتم های FTM که در این بخش مورد بررسی قرار گرفته اند بر اساس ماهیت گراف های درختی مورد حذف الگوریتم FTM، در جدول ۳ به این صورت طبقه بندی شده اند: (i) گراف های درختی نامنظم، (ii) گراف های درختی منظم، (iii) گراف های درختی آزاد، یا (iv) گراف های درختی آمیخته (ترکیبی) (تلفیقی از (i)، (ii)، (iii)). هم چنین، الگوریتم ها بر طبق ماهیت گراف های درختی فرعی که تبدیل به خروجی می شوند (زیر گراف های درختی فرعی بیشنیه، گراف های درختی فرعی مسدود، گراف های درختی فرعی القاء شده، یا گراف های درختی فرعی تثبیت شده) و ماهیت متریک پشتیبانی به خدمت گرفته شده (شمارش فعالیت ـ محور که با T_c بیان می شود؛ یا شمارش پیشامد ـ محور که با O_c بیان می شود) نیز طبقه بندی می شوند.

برای ارزیابی جایگزین الگوریتم ها FTM، ممکن است خوانندگان تمایل داشته باشند به چی ودیگران (۲۰۰۴) مراجعه کنند که یک پایه نظری و مطالعه اجرایی از مجموعه ای از الگوریتم های FTM که پیش از ۲۰۰۴ ارائه شده اند، فراهم آورده اند.

	Maximal	*Closed*	*Induced*	*Embedded*	T_c	O_c
					Unordered tree mining	
TreeFinder	⋆			⋆	⋆	
uFreqT			⋆		⋆	
cousinPair				⋆	⋆	
RootedTreeMiner			⋆		⋆	
SLEUTH				⋆	⋆	
					Ordered tree mining	
FREQT			⋆		⋆	
TreeMiner				⋆	⋆	
Chopper				⋆	⋆	
XSpanner				⋆	⋆	
AMIOT			⋆		⋆	
IMB3-Miner			⋆	⋆		⋆
TRIPS			⋆		⋆	
TIDS			⋆		⋆	
					Free tree mining	
FreeTreeMiner			⋆		⋆	
FTMiner			⋆		⋆	
F3TM			⋆		⋆	
CFFTree		⋆	⋆		⋆	
					Hybrid tree mining	
CMTreeMiner	⋆	⋆	⋆		⋆	
HybridTreeMiner			⋆		⋆	

۴,۱. استخراج گراف درختی فرعی نامنظم

گراف های درختی فرعی نامنظم برچسب دار اغلب برای مدل سازی (طراحی) داده های ساختاری به کار گرفته می شوند، دو حوزه کاربردی رایج عبارتند از تحلیل ترکیب های شیمیایی و ساختار هایپر ـ لینک وب (آسای و دیگران ۲۰۰۳).

گراف درختی فرعی نامنظم FTM قصد دارد برای بازنمایی گراف های درختی (به طوری که در زیر بخش ۳,۱ شرح داده شد) از DLS-LS، DLS یا BFCS استفاده کند. یک نمونه از الگوریتم های DLS-DS ـ محور که اغلب اوقات به آن استناد می شود الگوریتم $SLE \cup TH$ است (زاکی a ۲۰۰۵). الگوریتم $SLE \cup TH$ بر اساس کارهای قبلی که FTM انواع دیگری از گراف های درختی را هدف گرفته بودند، طراحی شد. این الگوریتم برای محاسبه پشتیبانی از دامنه ـ فهرست ها استفاده می کند. زاکی و دیگران (۲۰۰۵) دو مکانیسم تعمیم برای

ایجاد داوطلب در نظر گرفتند (i) تعمیم طبقه ـ محور و (ii) تعمیم مجاز. با استفاده از تعمیم طبقه ـ محور، لزوماً تمام داوطلب های ایجاد شده با این مکانیسم به فرم مجاز مطلوب وفادار نمی مانند، در نتیجه لازم است هر یک از داوطلب ها مورد بررسی قرار بگیرند برای اطمینان از اینکه در فرم مجاز هستند. در عوض، تعمیم مجاز را می توان فقط برای گراف های درختی فرعی پرتکرار مجازی که دارای یک کران پرتکرار معلوم هستند اعمال کرد، هر چند که منجر به پیدایش تعداد زیادی داوطلب های تصادفی اما مجاز می شود. به طوری که زاکی و دیگران (a ۲۰۰۵) گفته اند، میان استفاده از دو مکانیسم تعادل و موازنه وجود دارد. آزمایش هایی انجام شده توس زاکی و دیگران (۲۰۰۵a) نشان داده است که استفاده از تعمیم طبقه ـ محور کارآمدتر از تعمیم مجاز است.

نمونه ای ثابت از الگوریتم های FTM گراف های درختی فرعی نامنظم که از بازنمایی DLS استفاده می کند الگوریتم uFreq T است (نیجن و کوک ۲۰۰۳). در مرحله ایجاد داوطلب، الگوریتم uFreq T از تکنیک تعمیم دست راستی ترین مسیر برای ایجاد داوطلب ها استفاده می کند. در مرحله محاسبه پشتیبان، الگوریتم تناظر گراف درختی که برای تعیین فراوانی الگوی جاری استفاده می شود به یک الگوریتم تناظر بیشنیه دو جزئی با کارآیی محاسباتی بیشتر تبدیل می شود. به منظور تسهیل این فرآیند محاسبه پشتیبان، الگوریتم uFreq T ساختار داده ها را حفظ می کند تا تمام تناظرهای بالقوه برای رأس های موجود در دست راستی ترین مسیر و نشانگرها به تناظرهای والدین ذخیره شوند.

چی و دیگران (۲۰۰۵) الگوریتم Rooted Tree Miner را پیشنهاد کردند که بر اساس کد گذاری BFCS طراحی شده است. این الگوریتم بر خلاف $SLE \cup TH$ا uFreq T فقط بر یافتن گراف های درختی فرعی القاء شده پرتکرار تمرکز می کند. از این رو، در مرحله ایجاد داوطلب، می توان دامنه ای از رأس های قابل قبول و مجاز در یک وضعیت مشخص را محاسبه کرد. در مرحله محاسبه پشتیبان، ابتدا یک فهرست پیشامد برای هر یک از گراف های درختی فرعی مشخص t ساخته می شود. این فهرست شناسه های هر یک از فعالیت های گراف در پایگاه داده های گراف های درختی که شامل t باشند ثبت می کند و تناظر بین شاخص های رأس در t و موارد موجود در فعالیت گراف را نیز ثبت می کند. با استفاده از این فهرست پیشامد، پشتیبان t برابر با تعداد جزء هایی است که دارای شناسه های (ID_s) متمایز هستند.

تمام موارد فوق الذکر از تکنیک های تناظر دقیق استفاده می کنند. نمونه ای از یک FTM گراف درختی نامنظم که از تناظر غیر دقیق استفاده می کند الگوریتم FreeFinder است (ترمیر و دیگران ۲۰۰۲). الگوریتم FreeFinder یک رویکرد آپریوری ـ محور که از ارتباط شکل قبلی ـ شکل کنونی برای استخراج گراف های

درختی فرعی تثبیت شده استفاده می کند، را به خدمت می گیرد. البته الگوریتم هایی که از تناظر دقیق استفاده می کنند تضمینی برای تشخیص مجموعه کاملی از گراف های درختی فرعی تکرار نمی دهند اما بسیار کارآمد هستند.

بعضی از الگوریتم های FTM گراف های درختی نامنظم بر برنامه های کاربردی خاص تمرکز کرده اند و می توانند از ویژگی های این برنامه ها برای افزایش کارآمدی الگوریتم ها بهره بگیرند. به عنوان مثال، شا شا و دیگران (۲۰۰۴) یک الگوریتم FTM گراف درختی نامنظم، cousin Pair، برای کاربرد در تکامل نژادی ارائه کردند. آنها یک الگوی جالب را به عنوان یک "جفت منسوب" تعیین کردند، یک جفت از رأس هایی که فاصله نسبی (منسوب) و آستانه پیشامد مینیمم را تأمین می کنند. با استفاده از اینگونه محدودیت ها (الزامات)، الگوهای جالب از پایگاه داده های گراف درختی استخراج شد. در این جا، هدف اصلی دستیابی به درک بهتری از تاریخچه تکامل گونه هاست. مزیت های آشکار الگوریتم هایی مانند cousin Pair این است که برای عموم مسائل قابل اجرا نیستند.

۴,۲. استخراج گراف درختی منظم

برخلاف استخراج گراف درختی نامنظم، ماهیت نظم بخشی در گراف های درختی منظم را می توان برای معرفی قابلیت ها و توانایی ها با توجه به ایجاد گراف های درختی فرعی و آزمایش هم ریختی گراف های درختی فرعی، مورد استفاده قرار داد. گراف های درختی فرعی داوطلب معمولاً با استفاده از تعمیم دست راستی ترین مسیر یا تعمیم طبقه ـ محور هم ارز گسترش پیدا می کنند. به عنوان مثال، آسای و دیگران (۲۰۰۲) از تعمیم دست راستی ترین مسیر با در نظر گرفتن الگوریتم FREQT آنها بهره می گیرند. به علاوه، فقط پیشامدهای دست راستی ترین برگ الگوها ذخیره می شوند تا محاسبه پشتیبانی کارایی بیشتری پیدا کند. آسای و دیگران، داده های نیمه ـ ساختاری (به طور مثال صفحات وب) را با استفاده از گراف درختی منظم برچسب دار برای ارزیابی FREQT طراحی کردند.

مزیت تعمیم دست راستی با در نظر گرفتن گراف های درختی منظم این است که ایجاد مجموعه های کپی از داوطلب ها قابل اجتناب خواهد بود. هیدو و کوآوانو (۲۰۰۵) اشاره کردند که تعیین و شمارش با استفاده از تعمیم دست راستی ترین مسیر که توسط FREQT و سایر الگوریتم های FSM اتخاذ می شود، منجر به ایجاد تعداد زیادی داوطلب های تصادفی می شود که در نهایت به محاسبه غیر ضروری پشتیبان منتهی می شود. به

دنبال آن، هیدو و کوآوانو الگوریتم AMIOT را برای استفاده از یک طرح شمارش جدید به منظور کاهش تعداد داوطلب های تصادفی در عین حفظ مزیت های استراتژی تعمیم دست راستی معرفی کردند. این طرح، اتصال گراف درختی راست و چپ، ضمانت می کند که مجموعه داوطلب های گراف های درختی فرعی همواره زیر مجموعه ای باشد از آنچه که توسط فرآیند تعیین و شمارش و با استفاده از تعمیم دست راستی ترین مسیر محقق شده است. عملکرد AMIOT، با در نظر گرفتن داده های ترکیبی و داده های XML، نشان داد که سریعتر از FREQT است. با این وجود AMIOT در مقایسه با FREQT، حافظه بیشتری اشغال می کند، که این موضوع به خاطر ماهیت استراتژی BFS مورد استفاده AMIOT است.

زاکی (۲۰۰۲) یک الگوریتم FTM به نام TreeM معرفی کرد که از تعمیم طبقه محور هم ارز (همراه با بازنمایی DFS-LS) استفاده می کند. ایده دامنه ـ فهرست ها، که بعدها در $SLE \cup TH$ نیز به خدمت گرفته شد برای تسهیل محاسبه سریع پشتیبان طراحی شد. TreeMiner بر خلاف FREQT و AMIOT بر شناسایی گراف های درختی فرعی پرتکرار تثبیت شده تمرکز می کند. عملکرد این الگوریتم با یک الگوریتم مبنا یعنی الگوریتم Pattern Matcher که از استراتژی BFS استفاده می کند، مقایسه شد. نتایج تجربی نشان داد که Treeminer هرگاه برای داده های واقعی اعمال شود عملکرد بهتری از Pattern Matcher نشان می دهد. با این وجود، تکنیک هرس (حذف زوائد) که Treeminer از آن بهره می گیرد به کارآمدی تکنیک به خدمت گرفته شده توسط Pattern Matcher نیست و آستانه پشتیبانی پایینی ارائه می دهد. Treeminer یک الگوریتم FTM است که مرتباً و به کرات به آن رجوع می شود.

وانگ و دیگران (۲۰۰۴a) نیز الگوریتمی به نام Chopper را برای استخراج گراف های درختی فرعی تثبیت شده پرتکرار از مجموعه داده های درختی پیشنهاد کردند، اما از بازنمایی DLS استفاده کردند. الگوریتم Chopper ابتدا از Poefix Span اصلاح شده (چی و دیگران ۲۰۰۱) برای استخراج الگوهای پرتکرار بهره گرفت. سپس پایگاه داده های گراف درختی با مراجعه به الگوهای زنجیره ای شناسایی شده مجدداً اسکن شد تا الگوهای داوطلب و محاسبه های پشتیبان به وجود بیایند. دو فرآیند استخراج الگوی زنجیره ای و تأیید الگوی گراف درختی فرعی در الگوریتم Chopper تفکیک شده اند، و از این رو یک هزینه های محاسباتی اضافی پدید آمد. به منظور افزایش کارآمدی Chopper، الگوریتم XSpanner متعاقباً برای تلفیق استخراج الگوی زنجیره ای و فرآیند تأیید الگوی گراف درختی فرعی طراحی شد. با استفاده از تکنیک های طراحی شده پایگاه داده ها، الگوریتم XSpanner یک گراف درختی فرعی پرتکرار بزرگتر را از نمونه های کوچکتر ایجاد می کند و این فرآیند را از یکی از رأس ها آغاز می کند. هر دو الگوریتم Chopper , XSpanner هنگامی

که آستانه پشتیبان کمتر از ۵٪ باشد از الگوریتم Treeminer پیشی می گیرند (عملکرد بهتری نشان می دهند). با این وجود، مشخص شد که وقتی آستانه پشتیبان با کاهش بیشتری مواجه شود آنگاه الگوریتم Xspanner در مقایسه با Chopper منسجم تر عمل می کند.

IMB3-Miner (تان و دیگران ۲۰۰۶) نیز استخراج گراف های درختی فرعی تثبیت شده پرتکرار (از پایگاه داده های گراف درختی نامنظم) را هدف گرفته اند، اما از پارامتری استفاده می کنند که از طریق آن سطح تثبیت کننده کنترل می شود. هرگاه سطح تثبیت کننده برابر با ۱ باشد، گراف های درختی فرعی پر تکرار شناسایی شده گراف های درختی فرعی القاء شده خواهند بود. به همین خاطر، با اصلاح سطح تثبیت کننده، الگوریتم را می توان برای استخراج گراف های درختی فرعی تثبیت شده و القاء شده به کار گرفت. این الگوریتم با تلفیق ساختار داده های فهرست تثبیت کننده و استراتژی تعیین TMG (استراتژی ویژه تعمیم دست راستی ترین مسیر)، تضمین می کند که گراف های درختی فرعی داوطلب بدون نسخه کپی ایجاد شوند. علاوه بر این، برای هر یک از گراف های درختی فرعی یک فهرست پیشامد ذخیره می شود تا سرعت محاسبه پشتیبان افزایش پیدا کند. بر خلاف موارد پیشین، به جای استفاده از T_c، از O_c برای محاسبه پشتیبان الگوها استفاده می شود. به لحاظ تجربی نشان داده شده است که IMB3-Miner در مقایسه با Treeminer و FREQT عملکرد بهتر و بالاتری ارائه می دهد. کاربرد O_c به جای T_c به طور معمول زمانی انتخاب می شود که تکرار و ترتیب الگوها دارای اهمیت زیادی باشد.

تاتیکوندا و دیگران (۲۰۰۶) برای استخراج گراف های درختی فرعی تثبیت شده یا القاء شده در یک پایگاه داده گراف های درختی منظم ریشه ای، الگوریتم های TRIPS , TIDS را معرفی کردند. TRPS از توالی پروفر (Prufer) و دست چپی ترین مسیر الگو به عنوان شرایط و موقعیت تعمیم استفاده می کند. TIDS از توالی DFS و تعمیم دست راستی ترین مسیر استفاده می کند. محاسبه پشتیبان برای هر دو الگوریتم از فهرست تثبیت کننده و ساختار مجموعه ـ محور، برای تسهیل ایجاد تناوبی الگوها بهره می گیرد. بین هزینه حفظ فهرست های تثبیت کننده، کارایی محاسبه پشتیبان موازنه برقرار است، در حالی که تعداد برچسب های رأس مجزا و متمایز در مقایسه با تعداد کلی رأس های موجود در پایگاه داده ها کم است. نتایج تجربی نشان می دهد که هر دو الگوریتم TRIPS و TIDS از نظر زمان اجرا و استفاده از حافظه چه برای داده های ترکیبی و چه مجموعه داده های واقعی، عملکردی بهتری از TreeMiner نشان می دهند. TRIPS و TIDS، هنگامی که اندازه پایگاه داده ها افزایش میابد قابلیت محاسبه خوبی بروز می دهند. و حتی در هنگام استفاده از مقادیر آستانه پشتیبان نسبتاً پایین این الگوریتم ها قادر به استخراج پایگاه داده های بزرگی خواهند بود.

۴,۳. استخراج گراف درختی آزاد

الگوریتم های استخراج گراف درختی آزاد، به طوری که از نام آن پیداست، به گراف های درختی فرعی پرتکرار در مجموعه ای از گراف هایی درختی آزاد می پردازند. یکی از نمونه های اولیه، الگوریتم TreeMiner است (چی و دیگران ۲۰۰۳) که از عملیات خود اتصالی برای ایجاد گراف درختی فرعی داوطلب و ایجاد الگوریتم هم ریختی گراف درختی فرعی یا محاسبه پشتیبان استفاده می کند (چانگ ۱۹۸۷). نتایج تجربی نشان می دهند که الگوریتم Free Tree Miner قادر است داده های حقیقی بزرگ را با دامنه وسیعی از مقادیر پشتیبان کنترل کند و به کار بگیرد؛ با این وجود، هنگامی که اندازه گراف درختی فرعی پرتکرار بیشینه با توجه به افزایش بالقوه گراف های درختی فرعی بزرگتر می شود، این الگوریتم قابلیت محاسبه و ارزیابی خوبی نشان نمی دهد.

فرآیند مشابهی که توسط روکرت و کرامر انجام شد یک بازنمایی مجاز برای گراف های درختی فرعی بر چسب دار تعریف کردند (روکرت و کرامر ۲۰۰۴). این ها درک الگوریتم استخراج گراف درختی آزاد به نام FTMiner تثبیت شد. این الگوریتم در هر گام متناوب از مرحله ایجاد داوطلب بیش از یک رأس را تعمیم می دهد. همچنین الگوریتم مفهوم جدول تعمیم را بر می گزیند، که یک ساختار داده برای ذخیره تمام تعمیم ها برای الگوی گراف درختی فرعی به همراه مجموعه ای از فعالیت های گراف شامل الگو است. با استفاده از این جدول تعمیم، الگوریتم نه تنها حساب فراوانی هر یک از الگوهای گراف درختی فرعی را نگه می دارد بلکه اطلاعات مورد نیاز برای تعمیم الگوی جاری را نیز گرد آوری می کند و از این رو تعداد اسکن های پایگاه داده ها را به طور چشمگیری کاهش می دهد. آزمایش های تجربی در مورد پایگاه داده های بزرگ نشان می دهد که الگوریتم مذکور قادر است الگوهای پرتکرار را در مجموعه ای شامل بیش از ۳۷/۳۳۰ ترکیب شیمیایی با آستانه پشتیبانی ٪۲ استخراج کند.

ژائو وو (۲۰۰۶) با تمرکز عمده بر کاهش هزینه ایجاد داوطلب، الگوریتم استخراج گراف درختی آزاد به نام F3TM را معرفی کنند. الگوریتم ایده مرز تعمیم را برای تعریف موقعیت ها (رأس ها) به منظور گسترش گراف های درختی فرعی پرتکرار در مرحله ایجاد داوطلب معرفی می کند، و از تکنیک های هرس هم ریختی ـ محور و هرس مجاز را برای افزایش کارایی ایجاد داوطلب به کار می گیرد. تحقیقات عملکرد و اجرا نشان داده است که F3TM در مقایسه با سایر الگوریتم های استخراج گراف درختی آزاد مانند , Free Tree Miner FTMiner در مورد پایگاه داده های شیمیایی ۴۲/۳۹۰ ترکیب کارایی بیشتری داشته است. CFF Tree از مکانیسمی به نام هرس موقعیت بی خطر برای افزایش گراف های درختی فرعی فقط در موقعیت های بی

خطر و مطمئن بهره می گیرد، بنابراین وقتی تصمیم می گیرد که کدام یک از شاخه های گراف درختی شمارش و تعیین را حذف کند قابلیت های بیشتری ارائه می دهد. علاوه براین، CFF Tree از مکانیسم هرس برچسب بی خطر برای افزایش گراف های درختی فرعی روی رأس هایی با برچسب هایی که به لحاظ واژگان نمایی کمتر از "رأس افزاینده" جدید هستند، استفاده می کند، که بعداً برای حذف بعضی از فرآیندهای غیر ضروری تعیین به کار گرفته می شود. ارزیابی CFFTree نشان داد که در زمینه یافتن الگوهای مسدود با استفاده از پردازش ـ پسین، این الگوریتم پایه خود یعنی F3TM پیشی گرفته است.

۴,۴. استخراج گراف درختی ترکیبی

الگوریتم های استخراج گراف های درختی ترکیبی بر شکل های کلی تر گراف های درختی تمرکز کرده اند. به معنای دقیق کلمه، آنها را می توان به عنوان الگوریتم هایی طبقه بندی کرد که یکی از اهداف زیر را نشانه گرفته اند:

(i) گراف های درختی آزاد یا نامنظم، یا (ii) گراف های درختی منظم یا غیر منظم. نمونه ای از گروه اول، الگوریتم Hgbrid Treeminer است، و نمونه ای از گروه دوم، الگوریتم CMTreeMiner است.

Hgbrid TreeMiner (پی و دیگران a ۲۰۰۴) از بازنمایی BFCS استفاده می کند. در گراف درختی شمارش، هر گره نشانده یک گراف درختی در BFCF است. برای گره V در گراف درختی شمارش، زاده های V ممکن است با استفاده از عملیات تعمیم یا اتصال ایجاد شوند. عملیات اتصال برای یک جفت از گره های خواهری با ارتفاع (عمق) h اعمال شود، که موجب ایجاد یک گراف درختی BFCF با ارتفاع مشابه می شود. عملیات تعمیم با تعمیم یک برگ جدید در هر یک از سطح های پایین گراف درختی BFCF به ارتفاع h اعمال می شود که موجب ایجاد یک گراف درختی BFCF به ارتفاع (h+۱) می شود. این استراتژی شمارش ترکیبی برای استفاده از گراف درختی آزاد نیز تعمیم یافت. نتایج تجربی گزارش شده نشان می دهند که Hybrid TreeMiner سریعتر از Hybrid Treeminer عمل می کند و حافظه کمتری اشغال می کند.

CMTreeMiner برای استخراج گراف های درختی فرعی پیشینه و مسدود در مجموعه ای از گراف های درختی برچسب دار منظم یا نامنظم معرفی شد (چی و دیگران b ۲۰۰۴). با استفاده از تکنیک های هرس و اکتشاف، گراف درختی شمارش فقط روی شاخه هایی رشد پیدا می کند که به طور بالقوه قادر به تولید گراف های درختی فرعی بیشینه یا مسدود باشند، بنابراین از محاسبات اضافی وابسته به فرآیند یافتن تمام گراف

های درختی فرعی پرتکرار اجتناب می شود. مزیت پیشنهادی الگوریتم CMTreeMiner این است که به طور مستقیم گراف های درختی فرعی پرتکرار بیشینه و مسدود را ایجاد می کند بدون اینکه ابتدا تمام گراف های درختی فرعی پرتکرار را ایجاد کند. نتایج تجربی نشان داد که: (i) برای پایگاه داده های گراف درختی منظم، CMTreeMiner سریعتر از HgbridTreeMiner عمل می کند.

۴,۵. خلاصه الگوریتم های استخراجی گراف های درختی پرتکرار

از آنچه که پیش از این گفته شد می توان مشاهده کرد که روش ها، تکنیک ها و استراتژی های مختلفی برای تحقق FTM پیشنهاد شده اند. از دیدگاه عملکرد و برنامه های کاربردی، این الگوریتم ها را می توان به سه حوزه اصلی تقسیم کرد:

a) **تحلیل دسترس وب:** نمونه های این حوزه عبارتند از: SLEUTH، RootedTreeMiner، TreeMiner، IMB3-Miner، chopper، Xspanner، TRIPS، TIDS، CMTreeMiner، و HgbridTreeMiner
b) **تحلیل چند قالبی IP:** نمونه ها عبارتند از FreeTreeMiner و CMTreeMiner
c) **تحلیل ترکیب های شیمیایی:** نمونه ها عبارتند از: FreeTreeMiner، FTMiner، F3TM، CFFTree، HgbridTreeMiner

از نقطه نظر استراتژی عبور که در فضای جستجو به خدمت گرفته می شود، الگوریتم های FTM را می توان به دو گروه طبقه بندی کرد:

a) **استراتژی BFS:** استراتژی BFS از مزیت اجرای هرس کامل بهره می گیرد؛ که با این وجود، نیازمند استفاده قابل ملاحظه از حافظه است. نمونه های این حوزه عبارتند از: Rooted TreeMiner، AMIOT، FreeTreeMiner، و HgbridTreeMiner
b) **استراتژی DFS:** هرس ضعیف نقطه ضعف استراتژی DFS است. با این وجود، اشغال حافظه کمتر استراتژی BFS است. نمونه ها عبارتند از:UFreqT، SLEUTH، FREQT، Tree Miner، IMB3-Miner، TIDS، FTMiner، و CMTreeMiner

جدول ۴ فهرستی از تکنیک های مهم مورد استفاده برای ایجاد داوطلب و محاسبه پشتیبان با در نظر گرفتن الگوریتم های شرح داده شده در این بخش ارائه می دهد. به طور کلی، هر الگوریتم استخراج گراف های درختی فرعی پرتکرار دارای نقاط قوت و نقاط ضعفی است. هیچ الگوریتم استخراج گراف درختی فرعی

پرتکراری با قابلیت جهانی وجود ندارد. از نظر کارایی و اثر بخشی FTM، تکنیک های زیر بهترین عملکرد را ارائه می دهند:

- توالی DFS و شکل های آن برای بازنمایی گراف درختی
- استراتژی DFS برای عبور از فضای جستجو
- رشد گراف درختی شمارش با تعمیم دست راستی ترین مسیر در مرحله ایجاد داوطلب
- فهرست پیشامد برای محاسبه پشتیبان

جدول ۴ خلاصه الگوریتم های FTM متداول و مکانیسم های آنها در ایجاد داوطلب و محاسبه پشتیبان

الگوریتم	ایجاد داوطلب	محاسبه پیشتیبان
TreeFinder	ایجاد مجموعه آیتم های آپریوری	تکنیک های خوشه بندی
ufreqT	تعمیم دست راستی ترین مسیر	تناظر دو جزئی بیشینه
SLEUTH	تعمیم طبقه هم ارز	فهرست های ـ دامنه
Cousinpair	فاصله منسوب	جدول مراجعه
RostedTreeMiner	گراف درختی شمارش	فهرست پیشامد
FREQT	تعمیم دست راستی ترین مسیر	فهرست پیشامد
TreeMiner	تعمیم طبقه هم ارز	اتصال فهرست دامنه
Chopper	n/a	n/a
XSpanner		
AMIOT	اتصال گراف درختی راست و چپ	فهرست پیشامد
IMB3-Miner	TMG	فهرست پیشامد
TRIPS	تعمیم دست چپی ترین مسیر	جدول شمارش
TIDS	تعمیم دست راستی ترین مسیر	جدول شمارش
FreeTreeMiner	خود اتصالی	هم ریختی گراف درختی فرعی
FTMiner	جدول های تعمیم	مجموعه های پشتیبان
F3TM	گراف درختی شمارش + مرز تعمیم	الگوریتم عمق نشینی اولمان
CFFTree		
CMYreeMiner	گراف درختی شمارش	n/a

HybridTreeMiner	تعمیم + اتصال	فهرست پیشامد

نمونه هایی از الگوریتم هایی که دست کم شامل ۳ تکنیک هستند عبارتند از: SLEUTHT، FREQT، TreeMiner و IMB3Miner. در میان این الگوریتم ها، FREQT و TreeMiner معمولاً به عنوان الگوریتم های پایه برای مقایسه با سایرین انتخاب می شوند. TreeMiner یک الگوریتم FTM آپریوری ـ مبنا است، در حالی که FREQT یک الگوریتم به شیوه تعمیم دست راستی ترین مسیر است. این دو روش دو زنجیره را درون قلمرو FTM معرفی می کنند. اگر چه هم ریختی گراف درختی فرعی در زمان $O(\frac{K^{\frac{1}{5}}}{10gk}n)$ قابل حل است، تنها تعداد اندکی از الگوریتم های استخراج گراف درختی فرعی پرتکرار برای محاسبه پشتیبان مستقیماً از آن استفاده می کنند، فهرست های پیشامد در اغلب موارد برگزیده می شوند. دلیل اصلی برای انتخاب فهرست های پیشامد این است که اجرای فرآیند محاسبه پیشامد بسیار سرراست و روشن است.

۵. الگوریتم استخراج گراف های درختی فرعی پرتکرار

به طوری که در تصویر (b) ۱ اشاره شد، الگوریتم های FGM دارای کاربرد های قابل ملاحظه ای در انفورماتیک شیمیایی و تحلیل شبکه های زیستی هستند. گونه های مختلفی از الگوریتم های FGM در آثار این حوزه به ثبت رسیده است. در مورد FTM، ایجاد داوطلب و محاسبه پشتیبان دو موضوع کلیدی هستند. از آنجایی که ارزیابی هم ریختی گراف درختی فرعی به عنوان NP کامل شناخته می شود، حجم قابل توجهی از کارهای تحقیقی به رویکردهای مختلف برای ایجاد داوطلب کارآمد و مؤثر پرداخته اند. مکانیسم مورد استفاده برای ایجاد داوطلب مهم ترین ویژگی متمایز کننده این الگوریتم هاست. بررسی الگوریتم های مشهور استخراج زیر گراف پرتکرار در این بخش در اختیار شما قرار می گیرد. خوانندگان علاقمند باید توجه داشته باشند که بررسی مبنای نظری FGM، پیش از ۲۰۰۳، در وایشوو موتودا (۲۰۰۳) قابل مشاهده است. بررسی اخیرتر در زمینه استخراج الگوهای پرتکرار شامل: مجموعه آیتم ها، توالی ها، و زیر گراف ها در هان و دیگران (۲۰۰۷) موجود است.

در راستای اهداف بحث، الگوریتم های FGM مورد بررسی در این بخش به FGM "هدف عمومی" و "وابسته به الگو" تقسیم می شوند. تفاوت این دو این است که در دومی، ماهیت الگوهایی که باید شناسایی شوند به خاطر ماهیت حوزه کاربردی آن تا حدودی تخصصی و محدود است (فقط به زیر گراف هایی که برخی

محدودیت های خاص را تأمین می کنند علاقمندیم). در نتیجه، دانستن ماهیت این الگوهای ویژه می تواند کاهش فضای جستجو را امکان پذیر کند.

۵,۱ استخراج زیر گراف های پرتکرار "هدف عمومی "

در این زیر بخش، تعدادی از الگوریتم های FGM هدف عمومی مورد بررسی قرار می گیرند. برای کمک به بحث، الگوریتم ها بر اساس سه ضابطه طبقه بندی می شوند: (i) جامعیت جستجو (جستجوی دقیقا جستجوی غیر دقیق)، (ii) نوع ورودی (گراف های فعالیت یا گراف مجزا)، و (iii) استراتژی جستجو (DFS ا BFS)

۵,۱,۱ FGM غیر دقیق

الگوریتم های FGM بر مبنای جستجوی غیر دقیق از یک مقیاس تقریبی برای مقایسه تشابه دو گراف استفاده می کنند، یعنی برای اینکه هر یک از دو زیر گراف در محاسبه پشتیبان شرکت کنند نیازی نیست که کاملاً و مطلقاً شبیه به هم باشند، به جای آن یک زیر گراف ممکن است در محاسبه پشتیبان برای ایجاد داوطلب شرکت کند اگر از بعضی جهات به داوطلب شبیه باشد. جستجوی غیر دقیق البته ضمانتی برای پیدا کردن تمام زیر گراف های پرتکرار نمی دهد. اما ماهیت مقایسه تقریبی زیر گراف ها اغلب به دستاوردهای خوب در زمینه کارآمدی محاسباتی منتهی می شود. تنها مثال های معدودی از الگوریتم های استخراج غیر دقیق زیر گراف های پرتکرار در آثار این حوزه وجود دارد. با این وجود، یکی از نمونه های غالباً ذکر شده، الگوریتم SUBDUE است (کوک و هوادر ۱۹۹۴، ۲۰۰۰). الگوریتم SUBDUE از اصل ارتفاع تعریف مینیمم فشرده کردن داده های گراف استفاده می کند؛ و از روش جستجوی اکتشافی که از اطلاعات پیشین استفاده می کند برای محدود کردن فضای جستجو بهره می گیرد. اگر چه، کاربرد SUBDUE نشان دهنده نتایج امیدوار کننده در حوزه هایی از قبیل تحلیل تصاویر و تحلیل مدار CAD است اما قابلیت ارزیابی و محاسبه این الگوریتم مسئله مهمی است؛ یعنی زمان راه اندازی همراه با اندازه گراف ورودی به طور خطی افزایش پیدا نمی کند. علاوه بر این، SUBDUE مستعد شناسایی تعداد اندکی از الگوهاست.

Grew یکی دیگر از الگوریتم های FGM بر مبنای جستجوی غیر دقیق است (کوراموشی و کاریپیس b ۲۰۰۴). با این وجود، Grew بر یافتن زیر گراف های پیوسته که دارای تثبیت کننده های متعدد قطعی ـ رأس در گراف های بزرگ مجزا هستند تمرکز کرده است. Grew از رویکرد اکتشافی استفاده می کند که ادعا می

شود قابل محاسبه است زیرا از ایده قرارداد کردن و بازنویسی گراف استفاده می کند. Grew به عمد و آگاهانه فراوانی هر یک از گراف های شناسایی شده را کمتر از حد برآورد می کند تا فضای جستجو در کاهش دهد. آزمایش ها بر ۴ مجموعه داده ضابطه نشان داد که در زمینه زمان راه اندازی، تعداد الگوهای شناسایی شده، و اندازه الگوهای شناسایی شده، الگوریتم Grew به طور چشمگیری از SUBDUE پیشی گرفته است.

دو الگوریتم FGM جستجوی غیر دقیق که اخیراً ارائه شده اند عبارتند از gApprox (چی و دیگران ۲۰۰۷a) و RAM (ژانگ وانگ ۲۰۰۸). الگوریتم gApprox از ایده محدوده بالایی برای محاسبه پشتیبان و یک مقیاس تقریب برای شناسایی زیر گراف های تقریباً پیوسته پرتکرار در شبکه های خیلی بزرگ استفاده می کند. تحقیقات تجربی بر مبنای شبکه های متقابل پروتئین ـ پروتئین نشان میدهد که gApprox کارآمد است و الگوهای شناسایی شده دارای محتوای زیستی بوده اند. RAM بر اساس تعریف رسمی از الگوهای تقریبی پرتکرار در چارچوب داده های زیستی به شکل گراف طرح ریزی شده است، که در آن اطلاعات کران غیر دقیق است. آزمایش های گزارش شده نشان می دهد که RAM می تواند بعضی از الگوهای مهم که به وسیله الگوریتم های استخراج بر مبنای جستجوی دقیق قابل شناسایی نیستند را پیدا کند.

۵,۱,۲. FGM دقیق

الگوریتم های FGM دقیق متداول تر از الگوریتم های جستجوی غیر دقیق هستند، از این الگوریتم ها می توان در استخراج بر مبنای فعالیت گراف یا استخراج بر مبنای گراف مجزا استفاده کرد. یک ویژگی بنیادی برای الگوریتم های جستجوی دقیق این است که استخراج جامع و کامل است؛ یعنی الگوریتم استخراج پیدا کردن تمام زیر گراف های پرتکرار در داده های ورودی را تضمین می کنند. به طوری که در کوراموشی و کاریپیس (۲۰۰۴b) اشاره شد. اینگونه الگوریتم های استخراج کامل فقط در گراف های پراکنده شامل تعداد زیادی برچسب برای رأس ها و کران ها کارآمد است. با توجه به این محدودیت، این الگوریتم ها متحمل مقایسه مشروح و فراگیر علنی و غیر علنی هم ریختی زیر گراف می شود که باعث افزایش قابل ملاحظه محاسبات می شود.

بحث درباره الگوریتم های FGM دقیق را با بررسی FGM بر مبنای فعالیت گراف، استخراج مجموعه ای از گراف های نسبتاً کوچک آغاز می کنیم؛ FGM بر مبنای گراف مجزا در انتهای این زیر بخش بررسی خواهد شد. با توجه به استخراج فعالیت گراف، الگوریتم ها را می توان براساس استراتژی عبوری اتخاذ شده به دو

گروه BFS و DFS تقسیم کرد. BFS از آن جهت کارایی بیشتری دارد که حذف (هرس) زیر گراف های تصادفی (به قیمت اشغال بیشتر حافظه و I/O بالاتر) در مراحل اولیه فرآیند FGM را امکان پذیر می کند، در حالیکه DFS حافظه کمتری اشغال می کند (در عوض با کارایی کمتری فرآیند حذف زیر گراف های تصادفی را انجام می دهد). ابتدا الگوریتم های BFS را بررسی خواهیم کرد.

الکوریتم های FGM بر مبنای BFS نیز مانند الگوریتم های استخراج قوانین وابسته از قبیل آپریوری (آکراوان و اسرکانیت ۱۹۹۴)، از DCP استفاده می کند، یعنی یک زیر گراف (K+۱) می تواند پرتکرار باشد اگر زیر گراف مادر بلا واسطه K پرتکرار نباشد. با استفاده از BFS، مجموعه کاملی از داوطلب های K پیش از انتقال به داوطلب های (K+۱) پردازش می شوند که K به واحد تعمیم برای افزایش داوطلب ها اشاره دارد که آنرا می توان در قالب رأس ها، کران ها یا مسیرهای عبور بیان کرد. چهار الگوریتم FGM دقیق با سابقه در زیر فهرست شده اند:

- AGM (اینوکوچی و دیگران ۲۰۰۰) الگوریتمی با سابقه و قدیمی است که برای تعیین زیر گراف های القاء شده پرتکرار مورد استفاده قرار می گیرد. AGM از ماتریکس تجانب برای بیان گراف ها استفاده می کند و از جستجوی سطح ـ محور برای شناسایی زیر گراف های پرتکرار بهره می گیرد. AGM فرض را بر این می گذارد که تمام رأس های یک گراف متمایز از یکدیگرند. ارزیابی AGM از داده های شیمیایی تولید سرطان نشان می دهد که از رویکرد القائی بر مبنای برنامه ریزی منطقی ادغام شده با جستجوی سطح ـ محور کارآمدتر است. AGM نه تنها زیر گراف های پیوسته را پیدا می کند، بلکه زیر گراف های ناپیوسته با چندین جزء گراف مجزا را نیز شناسایی می کند. نسخه کارآمدتر AGM به نام ACGM نیز تنها برای استخراج زیر گراف های پیوسته پر تکرار طراحی شده است (اینوکوچی و دیگران، ۲۰۰۲). الگوریتم ACGM از اصول مشابه و بازنمایی گراف شبیه به AGM استفاده می کند. نتایج تجربی نشان می دهد که ACGM به طور چشمگیری سریع تر از AGM و FSG است. اینوکوچی و دیگران تحقیقات اصلی خود را گسترش دادند تا زیر گراف های القاء شده پرتکرار را از پایگاه داده های گراف عمومی (اصلی) که می تواند حاوی گراف های مستقیما غیر مستقیم، برچسب دارا بدون برچسب و حتی لوپ باشد، استخراج کنند (اینوکوچی و دیگران ۲۰۰۳).
- FSG (کوراموشی و کاریپیس ۲۰۰۱، ۲۰۰۴a) بر یافتن تمام زیر گراف های پیوسته پر تکرار تمرکز کرده است. FSG از استراتژی BFS برای افزایش داوطلب ها استفاده می کند که به موجب آن جفت زیر گراف های پرتکرار شناسایی شده K برای ایجاد زیر گراف های (K+۱) به یکدیگر متصل می شوند. FSG از روش

برچسب گذاری مجاز برای مقایسه گراف ها استفاده می کند و پشتیبان الگوها را با استفاده از بازنمایی داده های لیست فعالیت عمودی که به طور گسترده در FIM استفاده شده است، بهره می گیرد. آزمایش ها نشان می دهد که هرگاه گراف ها شامل تعداد زیادی رأس و کران با برچسب های مشابه باشند، FSG عملکرد خوبی نخواهد داشت زیرا عملیات اتصال که توسط FSG به خدمت گرفته می شود منجر به هم ریختی متعدد هسته های مجزا یا چندگانه می شود.

- الگوریتم FSG به پایگاه داده های گراف شامل آرایش دو بعدی رأس ها و کران ها در هر گراف (که گاهی اوقات به عنوان گراف های مکانی (جغرافیایی) شناخته می شوند) می پردازد. با این وجود، در تحلیل ترکیب های شیمیایی، کاربرها اغلب به گراف هایی نشان می دهند که همپایه هایی همراه با رأس ها در فضای دو یا سه بعدی هستند (که این گراف ها گاهی به عنوان گراف های هندسی شناخته می شوند). gFSG (کوراموشی و کاریپیس ۲۰۰۲)، الگوریتم FSG را برای شناسایی زیر گراف های هندسی پرتکرار با درجه ای از خطای مجاز در میان فعالیت های گراف هندسی گسترش می دهد. زیر گراف های هندسی استخراج شده نامتغیرهای چرخشی، صعودی و ترجمه هستند. gFSG و FSG از رویکرد مشابهی برای ایجاد داوطلب استفاده می کنند. به منظور تسریع محاسبه هم ریختی هندسی، تعدادی از ویژگی های مکان شناسی و ثابت تغییر شکل هندسی، در فرآیند تناظر مورد استفاده قرار می گیرند. در فرآیند محاسبه پشتیبانی. ثابت تغییر شکل هندسی (مانند فهرست کران ـ ضلع) و فهرست های فعالیت برای تسهیل محاسبات به کارگرفته می شوند. ارزیابی آزمایشی با استفاده از یک پایگاه داده های شیمیایی شامل بیش از ۲۰/۰۰۰ ترکیب شیمیایی انجام گرفت تا نشان دهد که gFSG در مورد مقادیر پشتیبان عملکرد خوبی داشته و با توجه به اندازه داده ها به طور خطی به اوج می رسد.
- DPMine (وانتیک و دیگران ۲۰۰۲؛ کورس و دیگران ۲۰۰۶) از مسیرهای کران ـ قطعه به عنوان واحدهای تعمیم برای ایجاد داوطلب استفاده می کند. استفاده از واحد تعمیم گسترده موجب کاهش تعداد داوطلب های ایجاد شده می شود. DPMine ابتدا تمام مسیرهای پرتکرار را شناسایی می کند، در وهله دوم تمام زیر گراف های دارای دو مسیر را پیدا می کند، و در گام سوم جفت زیر گراف های پرتکرار با (K_۱) مسیر که دارای (K_۲) مسیر مشترک هستند را در هم ادغام می کند به این منظور که زیر گراف های دارای K مسیر را به دست آورد. نتایج تجربی نشان می دهد که محاسبه پشتیبان مهم ترین عامل کمک کننده به زمان محاسبه است. کورس و دیگران (۲۰۰۶) همچنین پیشنهاد می کنند که کاهش محاسبه پشتیبان مهم تر از کاهش برآورد ایجاد داوطلب است. (Dpmine می تواند هم در داده های فعالیت ـ مبنا و هم داده های مبتنی بر گراف مجزا به طور مؤثری عمل کند.)

الگوریتم های FGM که استراتژی DFS را انتخاب می کنند به حافظه کمتری نیاز دارند زیرا از مشبک تمام زیر گراف های پرتکرار به روش DFS عبور می کنند. پنج الگوریتم مشهور در زیر فهرست شده اند:

- MoFa (بورلگت و برتولد ۲۰۰۲) به استخراج زیر گراف های پیوسته پرتکرار که مولکول را توصیف می کنند، می پردازد. این الگوریتم فهرست تثبیت کننده زیر گراف های از پیش پیدا شده را ذخیره می کند و عملیات تعمیم فقط به این تثبیت کننده ها محدود می شود. MoFa همچنین از حذف ساختاری و اطلاعات پیش زمینه برای کاهش محاسبه پشتیبان استفاده می کند. با این وجود، MoFa نسخه های کپی فراوانی تولید می کند که منجربه محاسبه غیر ضروری پشتیبان می شود.
- gSpan (یان و هان ۲۰۰۲) از بازنمایی مجاز M-DFSC برای بیان انحصاری هر زیر گراف استفاده می کند الگوریتم از ترتیب واژگان نمایی DFS برای ساختک مشبک درخت مانند روی تمام الگوهای موجود استفاده می کند، که منجر به ایجاد فضای جستجوی طبقاتی به نام گراف درختی کد DFS می شود. هر گره این گراف درختی جستجو معرف یک کد DFS است. سطح K+۱ ام گراف درختی دارای گره هایی است که شامل کدهای DFS برای زیر گراف های K هستند. زیر گراف های K با تعمیم یک کران از سطح K ام گراف درختی ایجاد می شوند. این گراف درختی به روش DFS قطع می شود و تمام زیر گراف های دارای کدهای DFS غیر حداقلی حذف می شوند به طوری که از فرآیندهای ایجاد داوطلب های زائد جلوگیری شود. به جای نگهداری لیست تثبیت کننده، الگوریتم gSpan فقط لیست فعالیت برای هر الگوی شناسایی شده را حفظ می کند؛ ارزیابی هم ریختی زیر گراف تنها برای گراف های درون لیست عمل می کند. gSpan، در مقایسه با الگوریتم های مبتنی بر لیست تثبیت کننده در مصرف حافظه صرفه جویی می کند. آزمایش های تجربی نشان می دهد که از نظر گستردگی و دامنه عمل، gSpan در مقایسه با FSG عملکرد بهتری دارد. gSpan قطعاً مورد استناد ترین الگوریتم FSM است.
- ADI-Mine (وانک و دیگران ۲۰۰۴b) به موضوع استخراج مجموعه داده های گراف دیسکت ـ محور بزرگ می پردازد. ADI-Mine از یک ساختار شاخص گذاری عمومی به نام ADI استفاده می کند. آزمایش ها نشان می دهند که ADI-Mine می توان مجموعه داده های گراف با یک میلیون گراف را استخراج کند، در حالی که gSpan فقط می تواند پایگاه داده هایی شامل ۳۰۰/۰۰۰ گراف را استخراج کند.
- FFSM (هوآن ودیگران ۲۰۰۳) به گراف های متراکم و بزرگ با تعداد کمی از برچسب ها می پردازند؛ به عنوان مثال، استخراج ساختار پروتئین. FFSM از بازنمایی CAM استفاده می کند. از این رو یک ساختار درخت مانند، یک گراف درختی CAM زیر مطلوب برای در برگرفتن تمام الگوهای موجود ساخته شد. هر

گره در آن گراف درختی CAM زیر مطلوب با عملیات اتصال یا تعمیم قابل شمارش خواهد بود. FFSM لیست های تثبیت کننده برای هر یک از الگوهای شناسایی شده را ثبت می کند تا از آزمایش هم ریختی زیر گراف علنی در مرحله محاسبه پشتیبان اجتناب شود. ارزیابی عملکرد با استفاده از چندین مجموعه داده های شیمیایی نشان می دهد که FFSM از gSpen بهتر عمل می کند.

- GASTON استخراج مسیر پرتکرار، گراف درختی فرعی پرتکرار و زیر گراف پرتکرار را در یک الگوریتم ادغام می کند، با در نظر گرفتن این مسئله که گراف های درختی آزاد، پرتکرار ترین زیر ساختارها در پایگاه داده های مولکولی هستند. (توسط نیجسن و کوک، ۲۰۰۴). این الگوریتم با منشعب کردن فرآیند استخراج زیر گراف های پرتکرار به استخراج مسیر، سپس استخراج گراف درختی فرعی و سرانجام استخراج زیر گراف توانستند راه حلی برای این مسئله ارائه دهند. در نتیجه، استخراج زیر گراف فقط در صورت نیاز فراخوانده می شود. بنابراین، GASTON زمانی که گراف ها عمدتاً به شکل مسیر ها یا گراف های درختی هستند بهترین عملکرد را دارد زیرا پرهزینه ترین فرآیند بررسی هم ریختی زیر گراف در مرحله استخراج زیر گراف روی می دهد. GASTON، لیست تثبیت کننده را ذخیره می کند به این منظور که فقط والدینی که واقعاً ظاهر می شوند رشد پیدا کنند؛ و از این طریق از ارزیابی غیر ضروری هم ریختی جلوگیری شود. آزمایش های تجربی نشان می دهد که GASTON با دامنه وسیعی از سایر الگوریتم های FGM در رقابت است.

به خاطر تنوع الگوریتم های FGM، تعیین نقاط قوت وضعف الگوریتم های مختلف دشوار است. با این وجود، و ورلین و دیگران (۲۰۰۵) مقایسه ای مشروح از چهار استخراج کننده DFS ـ محور: MoFa، gSpan، FFam و GASTON با توجه به عملکرد آنها در مجموعه داده های شیمیایی گوناگون ارائه کردند. در آزمایش ها، آنها به این نکته پی بردند که استفاده از لیست های تثبیت، در ازای اشغال حافظه بیشتر، دستاورد چشمگیری ارائه نمی دهند. آنها همچنین تأیید کردند که استفاده از بازنمایی مجاز برای ارزیابی نسخه های کپی در مقایسه با ارزیابی هم ریختی زیر گراف آشکار و علنی به محاسبه کمتری نیاز دارد. با بهره گیری از دو ویژگی متمایز کننده اصلی داده های مولکولی، یعنی "تقارن ها در مولکول" و " توزیع غیر یکنواخت فراوانی انواع اتم ها و پیوندها "،جان و کرامر (۲۰۰۵) عملکرد gSpan در زمینه استخراج پایگاه داده های مولکولی را بهبود بخشیدند.

این زیر بخش را با بررسی الگوریتم های FGM دقیق بر مبنای گراف مجزا که در آنها فراوانی یک الگو از طریق محاسبه پیشامد ـ محور تعیین می شود، تکمیل می کنیم (پیش از این اشاره کردیم که DPMine می تواند در

داده های فعالیت ـ محور و داده های مبتنی بر گراف مجزا خوب عمل کند). یک موضوع اساسی درباره استخراج گراف مجزا، چگونگی تعیین پشتیبان الگو است. DCP که اغلب برای هرس فضای جستجو در زمان استفاده از محاسبه فعالیت ـ محور به کار گرفته می شود در صورت محاسبه پیشامد ـ محور معتبر نیست. از این رو، مقیاس های پشتیبان پیشامد - محور که DCP را بپذیرند (تأمین کنند) مطلوب خواهند بود. یکی از قدیمی ترین مقیاس های پشتیبان پیشامد ـ محور که DCP را تداوم همپوشی برای هر یک از الگوها، مقیاس پشتیبان پیشامد ـ محور به صورت اندازه مجموعه مستقل ماکزیمم (MIS) رأس های موجود در گراف همپوشی تعریف می شود. مقیاس MIS ابتدا در وانتیک (۲۰۰۲) و کوراموشی و کاریپیس (۲۰۰۵، c ۲۰۰۴) معرفی شد. در وانتیک و دیگران (۲۰۰۶)، تعریف رسمی به همراه برهان هایی برای شرایط کافی و ضروری برای مقیاس های پشتیبان پیشامد ـ محور در جهت تداوم DCP ارائه شد. کار آنها با معرفی یک مقیاس جدید برای پشتیبان پیشامد – محور، که DCP را تأمین می کند و در زمان چند فرمولی قابل محاسبه است تداوم یافت (کالدرز و دیگران ۲۰۰۸).

کوراموشی و کاریپیس (۲۰۰۵ و c ۲۰۰۴) دو الگوریتم HSIGRAM و VSIGRAM را برای یافتن زیر گراف های پرتکرار در گراف های غیر متراکم بزرگ ارائه کردند. این دو الگوریتم به ترتیب از استراتژی های BFS و DFS استفاده می کردند و پشتیبان هر الگو با مقیاس MIS بر مبنای گراف همپوشی تعیین شد (وانتیک و دیگران ۲۰۰۶) چندین شکل از مقیاس های MIS از جمله مقیاس های MIS دقیق و تقریبی به اجرا گذاشته شدند. آزمایش های تجربی نشان داد که هر دو الگوریتم در استخراج گراف های بزرگ خوب عمل می کنند، هرچند که VSIGRAM سریع تر از HSIGRAM بود. دلیل برتری عملکرد الگوریتم VSIGRAM این است که حساب جاسازی های زیر گراف های پرتکرار در راستای مسیر DFS را نگه می دارد، که منجر به ارزیابی کمتر در زمینه هم ریختی زیر گراف می شود. در مقیاسه با SUBDVE، نتایج نشان می دهد که SUBDUE عملکر پایین تری نسبت به الگوریتم های HSIGRAM و VSIGRAM دارد؛ SUBDUE بر زیر گراف های کوچک با فراوانی بالا تمرکز می کند و در نتیجه الگوهای چشمگیر و مهم را از دست می دهد. کار کوراموشی و کاریپیس توسط اسکریبرو و اسکوبرمیر (۲۰۰۵) برای استخراج الگوهای پرتکرار از یک اندازه مشخص، ولی با در نظر گرفتن مفاهیم فراوانی جایگزین ادامه یافت. این الگوریتم فراوانی ـ محور، FPF، برای دو شبکه زیستی مختلف اعمال شد تا درون مایه های شبکه شناسایی شوند. با کمال تعجب. مقایسه تعداد الگوهای پرتکرار که با استفاده از مفاهیم فراوانی جایگزین شناسایی شده اند نشان می دهد که فراوانی یک الگو به تنهایی برای شناسایی درون

مایه های شبکه کافی نیست، و مشخص نیست که آیا الگوهای پرتکرار می توانند نقش های کلیدی در شبکه زیستی به عهده داشته باشند.

۵,۲ استخراج زیر گراف پر تکرار وابسته به الگو

در FSM، کاربران معمولاً به نوع مشخصی از الگو بیش از مجموعه کاملی از الگوها علاقه نشان می دهند، یعنی بعضی زیر مجموعه های یک مجموعه از زیر گراف های پرتکرار بیشتر مورد توجه هستند. این" الگوهای خاص" بر اساس جغرافیای آنها و / یا بعضی از محدودیت های خاص پر ماهیت الگوها تشخیص داده می شوند. الگوریتم های FGM وابسته به الگو را می توان بر حسب ماهیت الگوهای هدف به گروه های زیر تقسیم کرد:

(i) الگوهای ارتباطی، (ii) الگوهای بیشینه و مسدود، (iii) دسته ها و (iv) سایر الگوهای ساختگی. هر یک از این گروه ها در ادامه به طور مفصل مرود بحث قرار خواهند گرفت.

۵,۲,۱. استخراج الگوی ارتباطی

گراف های ارتباطی برای طراحی شبکه های گسترده ای مانند شبکه های زیستی و اجتماعی مناسب هستند. یان و دیگران (۲۰۰۵a) اشاره کردند که استخراج الگوی ارتباطی دارای سه ویژگی است که برای تفکیک آن از استخراج زیر گراف پرتکرار هدف عمومی به کار می روند: (i) داده ها دارای برچسب های متمایز رأس هستند، (ii) داده ها شامل گراف های بسیار بزرگ هستند، (iii) تمرکز بر الگوهای پرتکرار با محدودیت های اتصال مشخص (مثلاً رتبه منیمم یک الگو). بنابراین، هدف استخراج گراف ارتباطی تعیین تمام الگوهای پرتکرار نمایش گر محدودیت اتصال مشخص شده است.

CLOSECUT و SPLAT، که هر دو توسط یان و دیگران پیشنهاد شده اند (۲۰۰۵a) به استخراج زیرگراف های پرتکرار (مسدود) با محدودیت های اتصال می پردازند. CLOSECUT برای تلفیق محدودیت های اتصال از رویکرد افزایش الگو، همراه با تکنیک های فشردگی و تجزیه گراف استفاده می کند. الگوریتم SPLAT از یک رویکرد کاهش الگو برای تلفیق تکنیک تجزیه گراف بهره می گیرد. آزمایش ها نشان داد که CLOSECUT در مورد الگوهای با اتصال کم هنگامی از آستانه پشتیبان بالا استفاده می شود عملکرد بهتری از SPLAT نشان می دهد؛ با این وجود در الگوهای با اتصال بالا که از آستانه پشتیبان پایین استفاده می شود الگوریتم SPLAT از

CLOSECUT سبقت می گیرد با در نظر گرفتن داده های زیستی، نتایج نشان داد که هر دو الگوریتم می توانند الگوهای جالب با مضامین زیستی قوی و عمیق پیدا کنند.

۵,۲,۲. استخراج الگوهای بیشینه و مسدود

تعداد زیر گراف های پرتکرار موجود به طور بالقوه با اندازه گراف افزایش میابد، یعنی برای K گراف پرتکرار تعداد زیر گراف های پرتکرار آن می تواند به بزرگی 2^k باشد. در (یان و هان ۲۰۰۳) مشاهده شد که در حدود ۱/۰۰۰/۰۰۰ الگوی گراف پرتکرار از ۴۲۲ ترکیب شیمیایی تولید شد (با استفاده از آستانه پشتیبان ٪۵)؛ که تعداد زیادی از اینها به لحاظ ساختاری تکراری بودند. از این رو، هر دو رویکرد FGM بیشینه و مسدود به عنوان مکانیسم هایی برای محدود کردن تعداد زیر مورد استفاده قرار می گیرند، MFS بیانگر مجموعه ای از زیر گراف های پر تکرار بیشینه است، CHS نشان دهنده مجموعه ای از زیر گراف های پرتکرار مسدود است، و FS نشان دهنده مجموعه ای از تمام زیر گراف های پرتکرار در پایگاه داده های گراف است. بنابراین: MFS $\subseteq$ CFS $\subseteq$ FS.

فرض کنیم: $MFS = \left\{\frac{g}{g} \in FS\Delta - (\exists h \in FS\Delta Gch\right\}$. وظیفه فرآیند استخراج زیر گراف های پرتکرار بیشینه یافتن تمام الگوهای گراف متعلق به MFS است. زیر گراف های پرتکرار بیشینه تمام ساختارهای مشترک بیشینه را کد گذاری می کند، در صورت وجود شبکه های زیستی، آنها به عنوان جانب ترین الگوها پنداشته می شوند (کویوتورک ۲۰۰۴). با این وجود، فراوانی زیر گراف های بیشینه به وجود نمی آید. دو نمونه از الگوریتم های FGM بیشینه عبارتند از SPIN و MARGIN.

الگوریتم SPIN (هوآن و دیگران ۲۰۰۴) یک الگوریتم استخراج زیر گراف های پرتکرار بر مبنای گراف درختی در برگیرنده است که برای شناسایی زیر گراف های پرتکرار بیشینه با نیت کاهش هزینه های محاسباتی اضافی طراحی شده است. مفهوم طبقه های هم ارز بر مبنای گراف درختی به واسطه ایده گراف درختی در برگیرنده مجاز ارائه شد. در SPIN، روش تقسیم بندی گراف از طبقه های هم ارز بر مبنای گراف درختی همراه با سه تکنیک هرس استفاده می کند. این الگوریتم دو فاز اصلی دارد: (i) الگوریتم های استخراج، و (ii) ارزیابی تمام گراف های درختی فرعی پرتکرار درون داده های ورودی با استفاده از الگوریتم های استخراج گراف های درختی فرعی پرتکرار مناسب. زیر گراف های پرتکرار بیشینه مطلوب با بهینه سازی پردازش پسین ایجاد می

شوند. عملکرد SPIN با gSpan و FFSM مقایسه شد. نتایج نشان داد که در مورد داده های ترکیبی و شیمیایی، SPIN در مقایسه با gSpan و FFSM عملکرد بهتری ارائه می دهد.

MARGIN (توماس و دیگران ۲۰۰۶) بر مبنای این باور طراحی شد که مجموعه ای از زیر گراف های پرتکرار بیشینه در مجموعه ای از زیر گراف های پرتکرار K دارای زیر گراف های تصادفی K+۱ قرار دارند. در نتیجه، فضای جستجوی MARGIN، با هرس مشبک پیرامون مجموعه زیر گراف های پرتکرار بیشینه به طور قابل ملاحظه ای کاهش میابد. سپس، مجموعه داوطلب ها با عملیات پردازش پسین شناسایی می شود. نتایج تجربی نشان داد که در بعضی پایگاه داده ها، MARGIN به لحاظ محاسباتی سریع تر از gSpan عمل می کند. با این وجود، کارایی MARGIN تا حد زیادی به برش اولیه بستگی دارد.

فرض کنیم $CFS = \left\{\frac{g}{g} \in FS\Delta - (\exists h \in FS\Delta g < h\Delta \sup(g) = \sup(h)\right\}$. وظیفه استخراج زیر گراف پرتکرار مسدود، یافتن الگوهای متعلق به CFS است. این الگوهای مسدود دارای تعدادی مضامین زیستی هستند، زیرا به طور کلی، یک بیوشیمی فقط به بزرگترین ساختارها با ویژگی های معین علاقه دارند (فیشر و مینل ۲۰۰۴). CLOSECUT و SpLAT دو نمونه از الگوریتم های FGM مسدود هستند که قبلاً مورد بررسی قرار گرفته اند (زیر بخش ۵,۲,۱ را ببینید). نمونه دیگر نیز CloseGraph (یان و هان ۲۰۰۳) است، که براساس الگوریتم gSpan طراحی شده است. الگوریتم CloseGraph از حذف زود هنگام بر مبنای پیشامد هم ارز برای هرس فضای جستجو استفاده می کند. در شرایطی که حذف زودهنگام شکست می خورد و اجرای آن عملی نمی شود، ارزیابی ناکامی حذف زودهنگام به اجرا در می آید. نتایج تجربی نشان داد که CloseGraph بهتر از gSpan و FSG عمل می کند.

۵,۲,۳. استخراج دسته ها

یک دسته (یا به ظاهر دسته) زیر مجموعه ای از یک زیر گراف با مکان ثابت است. الگوریتم نخست برای جستجوی دسته ها توسط هاراری و راس (۱۹۵۷) ارائه شد. از آن به بعد تعداد زیادی الگوریتم طراحی شدند که انواعی از مسائل جستجو و بررسی دسته ها را نشانه گرفته بودند (بوفر و دیگران ۱۹۹۹؛ گوتین ۲۰۰۴). اخیراً مشخص شد که شناسایی دسته های پرتکرار از مجموعه ای از فعالیت های گراف در حوزه های از قبیل ارتباطات، بازرگانی و بیوانفورماتیک سودمند است. نمونه هایی از برنامه های کاربردی که استخراج دسته ها، یا به ظاهر ـ دسته ها اعمال شده اند عبارتند از:

استخراج شباهت (آبلو و دیگران ۲۰۰۲)، استخراج بیان ژن (چی و دیگران ۲۰۰۵)، و شناسایی سهام بسیار متناظر از گراف های بازار اوراق بهادار (وانگ و دیگران ۲۰۰۶)، از الگوریتم های FGM هدف عمومی می توان برای شناسایی اینگونه "الگوهای خاص" بهره گرفت، هر چند محاسبات کارآمد تر خواهند بود اگر ویژگی های خاص دسته ها نیز در نظر گرفته می شوند. دو نمونه از الگوریتم های استخراج دسته، CLAN و Cocain، در پاراگراف های بعد مورد بررسی قرار خواهند گرفت.

CLAN (وانگ و دیگران) به استخراج دسته های مسدود پرتکرار از پایگاه داده های متراکم بزرگ می پردازد. الگوریتم از ویژگی های ساختار دسته برای تسهیل بررسی هم ریختی دسته یا زیر دسته از طریق معرفی یک بازنمایی مجاز دسته استفاده می کند. وانگ و دیگران چندین تکنیک هرس مختلف را برای کاهش فضای جستجو به کار می گیرد. نتایج تجربی نشان داد که CLAN می تواند به طور مؤثری مجموعه داده های بزرگ و متراکم را استخراج کند. با این وجود، ارزیابی غیر مستقیم فقط از مقادیر بالای آستانه پشتیبان استفاده می کند و قابلیت محاسبه نیز نشان داد که تنها از مجموعه داده های گراف غیر متراکم و کوچک استفاده شده است.

ژنگ و دیگران با تعمیم و گسترش کران CLAN یک شکل کلی و عمومی از الگوریتم استخراج دسته، به نام Cocain معرفی کردند (ژنگ و دیگران ۲۰۰۶)، تا γ دستها به ظاهر دسته را از مجموعه داده های گراف متراکم و بزرگ استخراج کنند. در Cocain، برای تأمین پارامتر تعیین شده کاربر γ وجود دسته ها مورد نیاز است. Cocain از ویژگی های به ظاهر دسته ها برای هرس فضای جستجو استفاده می کند که با یک طرح بررسی اسناد به منظور تسریع فرآیند شناسایی تلفیق شده است. با این وجود، ارزیابی غیر مستقیم Cocain فقط به داده های بازار اوراق بهادار ایالات متحده آمریکا می پردازد.

۵,۲,۴. استخراج داده های محدود

ایده اصلی استخراج الگوی پرتکرار بر مبنای محدودیت دسترسی کاربر این است که محدودیت ها را در فرآیند استخراج ادغام کند به این منظور که فضای جستجو را هرس کند. ژو و دیگران (۲۰۰۷) چارچوبی به نام gprune ارائه کردند تا محدودیت های مختلف را با فرآیند استخراج زیر گراف های پرتکرار درهم آمیزد. gprune، فضای جستجو در داده ها و الگوها مورد بررسی قرار گرفت و یک مفهوم تازه تحت عنوان ضد یکنواختی داده های تفکیک ناپذیر از الگو برای حمایت از هرس مؤثر فضای جستجو ارائه شد. با این حال،

تحقیقی تجربی نشان داد که مزیت این فرآیند هرس ضد یکنواختی با سرعت تابع محدودیت متناظر با آن دو برابر شد. علاوه براین، آزمایش ها نشان داد که تأثیر و کارایی تلفیق محدودیت ها با فرآیند FSM تحت تأثیر جنبه های زیادی از قبیل ویژگی های داده ها و هزینه هرس قرار می گیرد. بنابراین الگوریتم استخراج محدودیت ـ محور می بایست تعادل بین هزینه هرس و هرگونه مزیت بالقوه را در نظر بگیرد.

۵,۳. خلاصه

جدول ۵، خلاصه ای از رویکردهای بازنمایی مجاز، ایجاد داوطلب و محاسبه پشتیبان که در الگوریتم FGM به کار گرفته می شوند و در این بخش توضیح داده شده اند را ارائه می دهد. با در نظر گرفتن الگوریتم های FGM لیست شده در جدول می توان ملاحظه کرد که SUBUE، AGM، FSG، MoFa، gSpan، FFSM، GASTON با بیشترین فراوانی ذکر شده اند. در میان این الگوریتم ها، SUBDUE گسترده تر از سایر الگوریتم ها مورد استفاده قرار می گیرد. با این وجود، یکی از نقطه ضعف های اغلب نقل شده SUBDUE این است که الگوریتم مذکور فقط به دنبال پیدا کردن الگوهای کوچک است که در نتیجه ممکن است الگوهای جانب بزرگ تر را از دست بدهد. AGM و FSG دو استخراج کننده BFS ـ محور (برمبنای روش BFS) هستند. Mofa یک استخراج کننده تخصصی برای داده های مولکولی است و می تواند گراف های مستقیم را استخراج کند. FFSM و GASTON در مورد گراف های مستقیم قابل اجرا نخواهند بود؛ در حالیکه gspan، با چند تغییر اندک، می تواند با گراف های مستقیم سازگار شود.

جدول ۵ خلاصه از الگوریتم های *FGM* متداول و مکانیسم های ایجاد داوطلب و محاسبه پشتیبان آنها

الگوریتم	بازنمایی	ایجاد داوطلب	محاسبه پشتیبان
AGM/ACGM	*CAM*	اتصال سطح ـ محور	اسکن پایگاه داده ها
FSG	*CAM*	اتصال سطح ـ محور	لیست فعالیت
gFSG	*n/a*	اتصال سطح ـ محور	لیست کردن ـ زاویه، لیست
DPmine	*n/a*	اتصال سطح ـ محور	فعالیت، آمیخته
MoFa	*n/a*	اتصال سطح ـ محور	*n/a*
gSpan	*m-DFSC*	تعمیم	لیست جاسازی ها
ADI-mine	*M-DFSC*	تعمیم دست راستی	لیست فعالیت
FFSM	*CAM*	ترین مسیر	لیست فعالیت
GASTON	*n/a*		
HSIGRAM	*CAM*		

VSIGRAM	*CAM*	تعمیم دست راستی	لیست جاسازی ها
FPF	*CAM*	ترین مسیر	لیست جاسازی ها
DPMine	*n/a*	اصال + تعمیم	مقیاس های *MIS* مختلف
CLOSECUT	*M-DFSC*	مسیر گراف درختی و	مقیاس های *MIS* مختلف
SPLAT	*n/a*	شمارش گراف	مقیاس های *MIS*
SPIN	*n/a*	اصال سطح ـ محور	*n/a*
MARGIN	*n/a*	تعمیم	لیست فعالیت
CLoseGraph	*m-DFSC*	تعمیم	*n/a*
CLAN	توالی برچسب رأس	اتصال سطح ـ محور	مجموعه جاسازی ها
COCain	توالی برچسب رأس	تعمیم دست راستی	*n/a*
Gprune	*M-DFSC*	ترین مسیر	لیست فعالیت
		n/a	*n/a*
		اتصال	*n/a*
		Expandcat	لیست فعالیت
		تعمیم دست راستی	
		ترین مسیر	
		تعمیم *DFS* ـ محور	
		تعمیم *DFS* ـ محور	
		تعمیم دست راستی	
		ترین مسیر	

یکی از ویژگی های مشترک اکثر الگوریتم های موجود در جدول این است که فضای جستجو معمولاً به عنوان یک مشبک درخت مانند برای تمام الگوهای موجود که به ترتیب واژگان نمایی منظم شده اند طراحی می شود. هر گره در مسبک نشان دهنده یک الگو است و ارتباط بین الگوها در سطح (k+۱) و K فقط از طریق رأس یا کران قابل تفکیک خواهد بود (یعنی رابطه مادرـ زاده برقرار است). بنابر این، استراتژی های جستجو شامل قطع مشبک و ذخیره تمام الگوهای تأمین کننده بعضی از آستانه ها است. یکی از استراتژی های جستجو شامل قطع مشبک و ذخیره تمام الگوهای تأمین کننده بعضی از آستانه ها است. یکی از استراتژی

های BFS و DFS را می توان برای قطع مشبک مورد استفاده قرار داد. استخراج کننده های مبتنی بر استراتژی BFS برتری هایی بر استخراج کننده های مبتنی بر استراتژی DFS ارائه می دهند که می تواند محدوده بالایی "محکم تری" برای پشتیبانی زیر گراف های K از پشتیبانی وابسته به مجموعه کامل زیر گراف های شناسایی شده (K_۱) به دست آورد. اطلاعات این محدوده بالایی را میتوان برای محدود کردن تعداد زیر گراف های داوطلب ایجاد شده به خدمت گرفت. استخراج کننده های مبتنی بر استراتژی DFS به طور معمول یک محدوده بالایی برای داوطلب های K استخراج می کنند که تنها بر یک زیر گراف پرتکرار مادر K_۱ استوار است.

به طوری که در وورلین و دیگران (۲۰۰۵) اشاره شد، یک الگوریتم FGM کارآمد معمولاً دارای سه ویزگی متمایز است:

- **تعمیم محدود کننده:** تعمیم یک زیر گراف تنها زمانی معتبر خواهد بود که گستره تعمیم در گراف های درون فهرست پیشامدهای زیر گراف موجود باشد. نمونه هایی از این گونه عملیات، استراتژی تعمیم دست راستی ترین مسیر است که توسط *gspan* به کارگرفته می شود و تعمیمی دست راستی ترین مسیر که توسط *MoFa* مورد استفاده قرار میگیرد.
- **ایجاد داوطلب کارآمد:** این عملیات با استفاده از بازنمایی گراف مجاز تحقق پیدا می کند. این بازنمایی می تواند فیلترینگ نسخه های کپی داوطلب را پیش از اجرای آزمایش هم ریختی گراف تسهیل کند. دو بازنمایی مجاز اصلی عبارتند از: (*i*) *CAM* که توسط *AGM*، *FSG* و *FFSM* استفاده می شود؛ و (*ii*) *M-DFSC* که توسط *gspan* مورد استفاده قرار می گیرد.
- **هم ریختی زیر گراف بنیادی:** در هنگام محاسبه پشتیبان یک الگو باید تعادل بین استفاده علنی از هم ریختی زیر گراف و حفظ جاسازی های الگو برقرار شود. *FFSM* و *GASTON* نمونه هایی از حفظ جاسازی ها هستند و *FSG* و *gspan* نیز مثال هایی از به خدمت گرفتن هم ریختی زیر گرافند.

اگر چه الگوریتم های طبقه بندی شده می توانند در پایگاه داده های بسیار بزرگ، مزیت متمایزی ارائه دهند، تعداد اندکی از محققان از این الگوریتم ها برای *FGM* استفاده کرده اند. یک نمونه که توسط فاتا و برتولد (۲۰۰۵) پیشنهاد شده است، تعمیم *MoFa* برای سازگار شدن با محاسبات طبقه بندی شده استخراج الگوهای پرتکرار در مورد مجموعه داده های معرف ترکیبات مولکولی بزرگ است.

۶. بحث و نتیجه گیری

بررسی وضعیت کنونی *FSM* موجود، به ویژه الگوریتم هایی که در آثار این حوزه بیشترین مراجعه به آنها بوده است، ارائه شده است. ایجاد داوطلب و محاسبه پشتیبان از لحاظ محاسباتی پرهزینه ترین جنبه های الگوریتم های *FSM* هستند و محاسبه پشتیبان پرهزینه ترین جنبه است. به طور گسترده، ویژگی متمایز کننده الگوریتم های استخراج بررسی شده در این تحقیق، چگونگی توجه مؤثر به فرآیند ایجاد داوطلب و محاسبه پشتیبان است.

با مراجعه به آثار این حوزه، تعداد زیادی استراتژی های استخراج برای انواع مختلفی از گراف ها معرفی شده اند تا انواع مختلفی از الگوها تولید شوند. برای اینکه تعدادی ساختار به دامنه وسیعی از الگوریتم های *FSM* ترسیم شده در آثار این حوزه تحمیل شوند از یک سیستم طبقه بندی استفاده کردیم که در آن، الگوریتم های *FSM* بر حسب (*i*) استراتژی ایجاد داوطلب، (*ii*) استراتژی جستجو و (*iii*) رویکرد محاسبه فراوانی بررسی می شوند. به طور کلی نمی توان الگوریتم های *FTM* را مستقیماً بر گراف ها اعمال کرد، در حالی که الگوریتم های *FGM* را می توان برای گراف ها و همچنین گراف های درختی به کار برد. الگوریتم های *FTM* و *FGM* به طور متفاوتی برای اهداف مختلفی طراحی شده اند. بنابراین، در این تحقیق، این دو گونه الگوریتم را جداگانه شرح می دهیم. برنامه های کاربردی عمومی برای الگوریتم های *FTM* استفاده از وب و استخراج داده های *XML* هستند، درحالی که الگوریتم های *FGM* به داده های شیمیایی و بیوانفورماتیک می پردازند. اگر چه انتشارات تحقیقی فراوانی در مورد برنامه های کاربردی *FGM* وجود دارد اما بسیاری از موضوعات مهم باید مورد بررسی قرار بگیرند.

نخست، آیا می توانیم مجموعه ای فشرده و با معنی از زیر گراف های پرتکرار را به جای مجموعه کامل زیر گراف های پرتکرار شناسایی کنیم ؟ بسیاری از تلاش های پژوهشی به کاهش مجموعه حاصل از زیر گراف های پرتکرار چشم دوخته اند؛ به عنوان مثال، استفاده از زیر گراف های پرتکرار بیشینه، زیر گراف های پرتکرار مسدود، زیر گراف های پرتکرار تقریبی و زیر گراف های پرتکرار تشخیص دهنده. با این وجود، درک روشنی از فشرده ترین و فراگیر ترین زیر گراف های پرتکرار برای هر یک از برنامه های کاربردی مشخص وجود ندارد. در بسیاری از موارد، مجموعه حاصل از زیر گراف های پرتکرار به قدری بزرگ هستند که نمی توان آنها را جداگانه تحلیل کرد و بسیاری از زیر گراف های پرتکرار شناسایی شده اغلب دارای ساختار تکراری هستند. کارهای پژوهشی که بر چگونگی کاهش قابل ملاحظه اندازه مجموعه حاصل از زیر گراف های پرتکرار تمرکز کرده اند بیشتر مورد تقاضا هستند.

ثانیاً، آیا می توانیم با استفاده از طبقه بندی کننده های مبتنی بر زیر گراف های پرتکرار در مقایسه با سایر روش ها به طبقه بندی بهتری دست پیدا کنیم؟ آیا می توانیم تکنیک های انتخاب خاصیت را در فرآیند استخراج زیر گراف پرتکرار ادغام کنیم و تشخیص دهنده ترین زیر گراف های مؤثر در طبقه بندی را مستقیماً تعیین کنیم؟ هنوز هم فرصت زیادی برای محققان وجود دارد که از تکنیک های سنتی استخراج داده ها بهره بگیرند و آنها را با فرآیند *FSM* تلفیق کنند.

ثالثاً، به طوری که اکثر محققان اشاره کرده اند، زیر گراف های پرتکرار دقیق در مورد بسیاری از حوزه های کاربردی واقعی، فایده چندانی ندارند. بنابراین. آیا می توانیم الگوریتم های کارآمدتری برای ایجاد زیر گراف های پرتکرار تقریبی طراحی کنیم؟ فعالیت های اندکی در زمینه استخراج داده های پرتکرار تقریبی انجام شده است البته الگوریتم مشهور *SUBDUE* در این زمینه یک استثناء است.

نهایتاً، در حوزه هایی که مانند طبقه بندی تصاویر هستند، استخراج گردش ـ کار، استخراج شبکه اجتماعی، استخراج بر مبنای گراف مجزا، و غیره؛ هنوز کارهای زیادی می تواند برای بهبود عملیات استخراج انجام داد. همواره بین دشواری و پیچیدگی الگوریتم های *FSM* و کارایی زیر گراف های پرتکرار شناسایی شده توسط آنها تعادل وجود دارد. کارهای فراوانی برای غلبه بر این موضوع باید انجام بگیرد. آیا فراوانی یک زیر گراف واقعاً مقیاس خوبی برای شناسایی زیر گراف های جالب است؟ آیا می توانیم به جای اتخاذ معیارهای حوزه استخراج اصول و قوانین وابسته، معیارهای جلب کننده دیگری برای شناسایی زیر گراف طراحی کنیم؟

مراجع

Abello, A., Resende, M.G.C. and Sundarsky, S. 2002. Massive Quasi-Clique Detection, In Proceedings of the 5th Latin America Symposium on Theoretical Informatics, 598–612.

Agrawal, R. and Srikant, R. 1994. Fast Algorithm for Mining Association Rules, In Proceedings of the 20th International Conference on Very Large Databases(VLDB), pp.487–499, Morgan Kaufmann.

Agrawal, R.C., Aggarwal, C.C. and Prasad, V.V.V. 2001. A Tree Projection Algorithm for Generation of Frequent Itemsets, Journal of Parallel and Distributed Computing 61(3), 350–371.

Alm, E. and Arkin, A.P. 2003. Biological Networks, Current Opinion in Structural Biology 13(2), 193–202.

Aldous, J.M. and Wilson, R.J. 2000. Graphs and Applications, an Introductory Approach, Springer.

Asai, T., Abe, K., Kawasoe, S., Arimura, H., Satamoto, H. and Arikawa, S. 2002. Efficient Substructure Discovery from Large Semi-Structured Data, In Proceedings of the 2nd SIAM International Conference on Data Mining, 158–174.

Asai, T., Arimura, H., Uno, T. and Nakano, S. 2003. Discovering Frequent Substructures in Large Unordered Trees, In Proceedings of the 6th International Conference on Discovery Science, 47–61.

Bayardo Jr., R.J. 1998. Efficiently Mining Long Patterns from Databases, In Proceedings of the 1998 International Conference on Management of Data, 85–93.

Borgelt, C. and Berthold, M. 2002. Mining Molecular Fragments: Finding Relevant Substructures of Molecules, In Proceedings of International Conference on Data Mining, 211–218.

Borgwardt, K.M. and Kriegel, H.P. 2005. Shortest-Path Kernels on Graphs, In Proceedings of the 2005 International Conference on Data Mining, 74–81.

Bomze, I.M., Budinich, M., Pardalos, P.M. and Pelillo, M. 1999. The Maximum Clique Problem, Handbook of Combinatorial Optimization, Kluwer Academic Publishers 4, 1–74.

Brin, S. and Page, L. 1998. The Anatomy of a Large-scale Hyper-textual Web Search Engine, In Proceedings of the 7th International World Wide Web Conference, 107–117.

Bunke, H. and Shearer, K. 1998. A Graph Distance Metric based on the Maximal Common Subgraph, Pattern Recognition Letters 19, 225–259.

Bunke, H. and Allerman, G. 1983. Inexact Graph Matching for Structural Pattern Recognition, Pattern Recognition Letters 1(4), 245–253.

Calders, T., Ramon, J. and van Dyck, D. 2008. Anti-monotonic Overlap-graph Support Measures, In Proceedings of the Eighth IEEE International Conference on Data Mining, 73–82.

Chakrabarti, S., Dom, B., Gibson, D., Kleinberg, J., Kumar, R., Raghavan, P., Rajagopaln, S. and

Tomkins, A. 1999. Mining the Link Structure of the World Wide Web, IEEE Computer 32(8), 60–67.

Chen, C., Yan, X., Zhu, F. and Han, J. 2007a. gApprox: Mining Frequent Approximate Patterns from a Massive Network, In Proceedings of the 7th IEEE International Conference on Data Mining, 445–450.

Chen, C., Yan, X., Yu, P.S., Han, J., Zhang, D. and Gu, X. 2007b. Towards Graph Containment Search and Indexing, Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB'07), 926–937.

Chen, M.S., Han, J. and Yu, P.S. 1996. Data Mining: An Overview from Database Perspective, IEEE Transaction on Knowledge and Data Engineering 8, 866–883.

Chen, C., Lin, C.X., Yan, X. and Han, J. 2008. On Effective Presentation of Graph Patterns: A Structural Representative Approach, In Proceedings of the 17th ACM Conference on Information and Knowledge Management, 299–308.

Chi, Y., Yang, Y., Xia, Y. and Muntz, R.R. 2003. Indexing and Mining Free Trees, In Proceedings of the 2003 IEEE International Conference on Data Mining, 509–512.

Chi, Y., Nijssen, S., Muntz, R. and Kok, J. 2004. Frequent Subtree Mining An Overview, Fundamenta Informaticae, Special Issue on Graph and Tree Mining 66(1-2), 161–198.

Chi, Y., Yang, Y., Xia, Y. and Muntz, R.R. 2004a. HybridTreeMiner: An Efficient Algorithm for Mining Frequent Rooted Trees and Trees using Canonical Forms, In Proceedings of the 16th International Conference on Scientific and Statistical Database Management, 11–20.

Chi, Y., Yang, Y., Xia, Y. and Muntz, R.R. 2004b. CMTreeMiner: Mining both Closed and Maximal Frequent subtrees, In Proceedings of the 8th Pacific Asia Conference on Knowledge Discovery and Data Mining, 63–73.

Chi, Y., Yang, Y., Xia, Y. and Muntz, R.R. 2005. Canonical Forms for labelled Trees and Their Applications in Frequent subtree Mining. Journal of Knowledge and Information Systems 8(2), 203–234.

Christmas, W.J., Kittler, J. and Petrou, M. 1995. Structural Matching in Computer Vision using Probabilistic Relaxation, IEEE Transactions on Pattern Analysis and Machine Intelligence 17(8), 749–764.

Chung, J.C. 1987. O(n

2.5

) Time Algorithm for Subgraph Homeomorphism Problem on Trees, Journal of Algorithms 8, 106–112.

Conte, D., Foggia, F., Sansone, C. and Vento, M. 2004. Thirty Years of Graph Matching in Pattern

Recognition, International Journal of Pattern Recognition and Artificial Intelligence 18(3), 265–298.

Cook, D.J. and Holder, L.B. 1994. Substructure Discovery Using Minimum Description Length and Background Knowledge, Journal of Artificial Intellligence Research 1, 231–255.

Cook, D.J. and Holder, L.B. 2000. Graph-based Data Mining, IEEE Intelligent Systems 15(2), 32–41.

Cordella, L.P., Foggia, P., Sansone, C., Tortorella, F. and Vento, M. 1998. Graph Matching: A Fast Algorithm and its Evaluation, In Proceedings of the 14th Conference on Pattern Recognition, 1582–1584.

28

Cordella, L.P., Foggia, P., Sansone, C. and Vento, M. 2001. An Improved Algorithm for Matching Large Graphs, In Proceedings of the 3rd IAPR-TC15 Workshop on Graph-based Representation in Pattern Recognition, 149–159.

Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C. 2001. Introduction to Algorithms, 2nd Edition, MIT Press and McGraw-Hill.

Cui, J.H., Kim, J., Maggiorini, D., Boussetta, K. and Gerla, M. 2005. Aggregated Multicast-a Comparative Study, Cluster Computing 8(1), 15–26.

Diestel, R. 2000. Graph Theory, Springer-Verlag.

Dehaspe, L., Toivonen, H. and King, R.D. 1998. Finding Frequent Substructures in Chemical Compounds, In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD'98), 30–36.

Deshpande, M., Kuramochi, M., Wale, N., and Karypis, G. 2005. Frequent Sub-Structure-based Approach for Classifying Chemical Compounds, IEEE Transactions on Knowledge and Data Engineering 17(8), 1036–1050.

Fan, W., Zhang, K., Cheng, H., Gao, J., Yan, X., Han, J., Yu, P.S. and Verscheure, O. 2008. Direct Mining of Discriminative and Essential Frequent Patterns via Model-based Search Tree, In Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 230–238, Las Vegas, USA.

Fatta, G.D. and Berthold, M.R. 2005. High Performance Subgraph Mining in Molecular Compounds, In Proceedings of the 2005 International Conference on High Performance Computing and Communications (HPCC'05), 866–877.

Fischer, I. and Meinl, T. 2004. Graph based Molecular Data Mining - An Overview, In Proceedings of the 2004 IEEE International Conference on Systems,Man and Cybernetics, 4578–4582.

Flake, G., Tarjan, R. and Tsioutsiouliklis, K. 2004. Graph Clustering and Minimum Cut Trees, Internet

Mathematics 1, 385–408.

Fortin, S. 1996. The Graph Isomorphism Problem, Technical Report, no. TR06-20, The University of Alberta.

Foggia, P., Genna, R. and Vento, M. 2001. A Performance Comparison of Five Algorithms for Graph Isomorphism, In Proceedings of the 3rd IAPR-TC15 Workshop on Graph-based Representation in Pattern Recognition, 188–199.

Freeman, L. 1979. Centrality in Social Networks Conceptual Clarification, Social Networks 1(3), 215–239.

Garey, M.R. and Johnson, D.S. 1979. Computers and Intractability - A Guide to the Theory of NPCompleteness. W.H. Freeman And Company. New York.

G¨artner, T., Flach, P. and Wrobel, S. 2003. On Graph Kernels: Hardness Results and Efficient Alternatives, In Proceedings of the 16th Annual Conference on Learning Theory (COLT'03), 129–143.

Getoor, L. and Diehl, C. 2005. Link Mining: A Survey, ACM SIGKDD Explorations Newsletter 7(2), 3–12.

Gibbons, A. 1985. Algorithmic Graph Theory, Cambridge University Press.

Greco, G., Guzzo, A., Manco, G., Pontieri, L. and Sacc´a, D. 2005. Mining Constrained Graphs: the Case of Workflow Systems, Constraint based Mining and Inductive Databases, 155–171, LNCS, Springer.

Gudes, E., Shimony, S.E. and Vanetik, N. 2006. Discovering Frequent Graph Patterns using Disjoint Paths, IEEE Transaction on Knowledge and Data Engineering 18(11), 1441–1456.

Gutin, G. 2004. 5.3 Independent Sets and Cliques, In Gross,J.L. and Yellin,J. Handbook of Graph Theory, Discrete Mathematics & Its Applications, CRC Press, 389–402.

Han, J., Pei, J. and Yin, Y. 2000. Mining Frequent Patterns without Candidate Generation, In Proceedings of ACM SIGMOD International Conference on Management of Data, 1–12.

Han, J., Cheng, H., Xin, D. and Yan, X. 2007. Frequent Pattern Mining: Current Status and Future Directions, Journal of Data Mining and Knowledge Discovery 15(1), 55–86.

Han, J. and Kamber, M. 2006. Data Mining Concepts and Techniques, 2nd Edition, San Francisco: Morgan Kaufmann.

Harary, F. and Ross, I.C. 1957. A Procedure for Clique Detection using the Group Matrix, Sociometry 20(3), 205–215.

Hein, J., Jiang, T., Wang, L. and Zhang, K. 1996. On the Complexity of Comparing evolutionary trees. Discrete Applied Mathematics 71(1-3), 153–169.

Hido, S. and Kawano, H. 2005. AMIOT:Induced Ordered Tree Mining in Tree-structured Databases, In

Proceedings of the 5th IEEE International Conference on Data Mining, 170–177.

Hopcroft, J.E. and Tarjan, R.E. 1972. Isomorphism of Planar Graphs, In Miller,R.E. and Thatcher,J.W., Complexity of Computer Computations, 131–152.

Hu, H., Yan, X., Huang, Y., Han, J. and Zhou, X. 2005. Mining Coherent Dense Subgraphs across Massive Biological Networks for Functional Discovery,Bioinformatics 21(1), 213–221.

Huan, J., Wang, W. and Prins, J. 2003. Efficient Mining of Frequent Subgraph in the Presence of Isomorphism, In Proceedings of the 2003 International Conference on Data Mining, 549-552.

Huan, J., Wang, W., Prins, J. and Yang, J. 2004a. SPIN: Mining Maximal Frequent Subgraphs from Graph Databases, In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 581–586.

Huang, X. and Lai, W. 2006. Clustering Graphs for Visualization via Node Similarities, Visual Language and Computing 17, 225–253.

Inokuchi, A., Washio, T. and Motoda, H. 2000. An Apriori-based Algorithm for Mining Frequent Substructures from Graph Data, In Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases, 13–23.

Inokuchi, A., Washio, T., Nishimura, K. and Motoda, H. 2002. A F ast Algorithm for Mining Frequent Connected Subgraphs, Technical Report RT0448, IBM Research, Tokyo Research Laboratory, Japan.

Inokuchi, A., Washio, T. and Motoda, H. 2003. Complete Mining of Frequent Patterns from Graphs: Mining Graph Data, Journal of Machine Learning, 50(3), 321–354.

Jahn, K. and Kramer, S. 2005. Optimizing gSpan for Molecular Datasets, In Proceedings of the 3rd International Workshop on Mining Graphs,Trees and Sequences, 509–523.

Kashima, H., Tsuda, K. and Inokuchi, A. 2003. Marginalized Kernels Between labelled Graphs, In Proceedings of the 20th International Conference on Machine Learning (ICML'03), 321–328.

Kelley, B., Sharan, R., Karp, R., Sittler, E., Root, D., Stockwell, B. and Tdeker, T. 2003. Conserved Pathways within Bacteria and Yeast as Revealed by Global Protein Network Alignment. In Proceedings of the National Academy of Science of the United States of America (PNAS'03) 100(20), 11394–11399.

Ke, Y., Cheng, J. and Ng, W. 2007. Correlated Search in Graph Databases, In Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 390–399.

Ke, Y., Cheng, J. and Yu, J. 2009. Efficient Discovery of Frequent Correlated Subgraph Pairs. In Proceedings of the 9th IEEE International Conference on Data Mining, 239–248.

Kleinberg, J.M. 1998. Authoritative Sources in a Hyper-linked Environment, In Proceedings of ACMSIAM Symposium Discrete Algorithms, 668–677.

Kosala, R. and Blockeel, H. 2000. Web Mining Research: A Survey, ACM SIGKDD Explorations Newsletter 2(1), 1–15.

Koyut¨urk, M., Grama, A. and Szpankowski, W. 2004. An Efficient Algorithm for Detecting Frequent Subgraphs in Biological Networks, Journal of Bioinformatics 20(1), 200–207.

Kudo, T., Maeda, E. and Matsumoto, Y. 2004. An Application to Boosting to Graph Classification, In Proceedings of the 8th Annual Conference on Neural Information Processing Systems, 729–736.

Kuramochi, M. and Karypis, G. 2001. Frequent Subgraph Discovery, In Proceedings of International Conference on Data Mining, 313–320.

Kuramochi, M. and Karypis, G. 2002. Discovering Frequent Geometric Subgraphs, In Proceedings of the IEEE International Conference on Data Mining, 258–265.

Kuramochi, M. and Karypis, G. 2004a. An Efficient Algorithm for Discovering Frequent Subgraphs, IEEE Transactions on Knowledge and Data Engineering 16(9), 1038–1051.

Kuramochi, M. and Karypis, G. 2004b. GREW-A Scalable Frequent Subgraph Discovery Algorithm, In Proceedings of the 4th IEEE International Conference on Data Mining, 439–442.

Kuramochi, M. and Karypis, G. 2004c. Finding Frequent Patterns in a Large Sparse Graph, In Proceedings of the SIAM International Conference on Data Mining, 345–356.

Kuramochi, M and Karypis, G. 2005. Finding Frequent Patterns in a Large Sparse Graph, Data Mining and Knowledge Discovery 11(3), 243–271.

Liu, B. 2008. Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data, Springer.

Liu, T.L. and Geiger, D. 1999. Approximate Tree Matching and Shape Similarity, In Proceedings of 7th International Conference on Computer Vision, 456–462.

Matula, D.W. 1978. Subtree Isomorphism in O(n

5/2

), Annals of Discrete Mathematics 2, 91–106.

McKay, B.D. 1981. Practical Graph Isomorphism, Congressus Numerantium 30, 45–87.

Messmer, B.T. and Bunke, H. 1998. A New Algorithm for Error-Tolerant Subgraph Isomorphism Detection, IEEE Transaction on Pattern Analysis and Machine Intelligenc e 20(5), 493–504.

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D. and Alon, U. 2002. Network Motifs: Simple Building Blocks of Complex Networks, Science 298(5594), 824–827.

Miyazaki, T. 1997. The Complexity of McKay's Canonical labelling Algorithm, Groups and Computation II, DIMACS Series Discrete Mathematics Theoretical Computer Science, American Mathematical Society 28, 239–256.

Newman, M.E.J. 2004. Detecting Community Structure in Networks, The European Physical Journal B - Condensed Matter and Complex Systems 38(2), 321–330.

Nijssen, S. and Kok, J.N. 2003. Efficient Discovery of Frequent Unordered Trees, In Proceedings of the 1st International Workshop on Mining Graphs,Trees and Sequences, 55–64.

Nijssen, S. and Kok, J.N. 2004. A Quickstart in Frequent Structure Mining can Make a Difference, In Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 647–652.

Ozaki, T. and Ohkawa, T. 2008. Mining Correlated Subgraphs in Graph Databases, In Proceedings of the 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'08), 272–283.

Paul, S. 1998. Multicasting on the Internet and Its Applications, Kluwer Academic Publishers.

Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U. and Hsu, M.C. 2001. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth, In Proceedings of 12th IEEE International Conference on Data Engineering (ICDE 01), 215–224, Heidelberg, Germany.

Pei, J., Jiang, D. and Zhang, A. 2005. On Mining Cross-Graph Quasi-Cliques, In Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 228–238, Chicago, USA.

Preiss, B.R. 1998. Data Structures and Algorithms with Object-Oriented Design Patterns in C++, Wiley.

Pr¨ufer, H. 1918. Neuer Beweis eines Satzes ¨uber Permutationen, Archiv f¨ur Mathematik und Physik 27, 742–744.

Raedt, L.D., and Kramer, S. 2001. The Levelwise Version Space Algorithm and its Application to Molecular Fragment Finding. In Proceedings of the 17th International Joint Conference on Artificial Intelligence 2, 853–859.

Read, R.C. and Corneil, D.G. 1977. The Graph Isomorph Disease, Journal of Graph Theory 1, 339–363.

R¨uckert, U. and Kramer, S. 2004. Frequent Free Tree Discovery in Graph Data, In Proceedings of Special Track on Data Mining, ACM Symposium on Applied Computing, 564–570.

Russel, S. and Norvig, P. 2003. Artificial Intelligence: A Modern Approach, 2nd Edition, Prentice Hall, Upper Saddle River, New Jersey.

Schmidt, D.C. and Druffel, L.E. 1976. A Fast Backtracking Algorithm to Test Directed Graphs for Isomorphism using Distance Matrices, Journal of the ACM 23(3), 433–445.

Schreiber, F. and Schw¨obbermeyer, H. 2005. Frequency Concepts and Pattern Detection for the analysis of motifs in Networks, Transactions on Computational Systems Biology 3, 89–104.

Shamir, R. and Tsur, D. 1999. Faster Subtree Isomorphism, Journal of Algorithms 33(2), 267–280.

Shapiro, L.G., and Haralick, R.M. 1981. Structural Descriptions and Inexact Matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 3, 504–519.

Shasha, D., Wang, J. and Zhang, S. 2004. Unordered Tree Mining with Applications to Phylogeny, In Proceedings of the 20th International Conference on Data Engineering (ICDE 04), 708–719.

Shasha, D., Wang, J.T.L. and Giugno, R. 2002. Algorithms and Applications of Tree and Graph Searching, In Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles on Database Systems, 39–52.

Sharan, R., Suthram, S., Kelley, R., Kuhn, T., McCuine, S., Uetz, P., Sittler, T., Karp, R. and Ideker, T. 2005. Conserved Patterns of Protein Interaction in Multiple Species, In Proceedings of the National Academy of Science of the United States of America (PNAS'05) 102(6), 1974–1979.

Tan, H., Dillon, T.S., Hadzic, F., Chang, E. and Feng, L. 2006. IMB3-Miner: Mining Induced/Embedded subtrees by Constraining the Level of Embedding, In Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining, 450–461.

Tan, H., Dillon, T.S., Feng, L., Chang, E. and Hadzic, F. 2005. X3-Miner: Mining Patterns from XML Database, In Proceedings of the 6th International Data Mining, 287–297.

Tatikonda, S., Parthasarathy, S. and Kurc, T. 2006. Trips and Tides: New Algorithms for Tree Mining, In Proceedings of the 15th ACM International Conference on Information and Knowledge Management, 455–464.

Termier, A., Rousset, M.C. and Sebag, M. 2002. Treefinder: a First Step Towards XML Data Mining, In Proceedings of the 2002 IEEE International Conference on Data Mining, 450–457.

Thomas, L.T., Valluri, S.R. and Karlapalem, K. 2006. MARGIN: Maximal Frequent Subgraph Mining, In Proceedings of the 6th International Conference on Data Mining (ICDM 06), 1097–1101, Hong Kong.

Tsuda, K. and Kudo, T. 2006. Clustering Graphs by Weighted Substructure Mining. In Proceedings of the 23rd International Conference on Machine Learning (ICML'06), 953–960.

Ullmann, J.R. 1976. An Algorithm for Subgraph Isomorphism. Journal of the ACM 23(1), 31–42.

Valiente, G. 2002. Algorithms on Trees and Graphs, Springer.

Vanetik, N., Gudes, E. and Shimony, S.E. 2002. Computing Frequent Graph Patterns from Semistructured Data, In Proceedings of the 2nd International Conference on Data Mining, 458–465.

Vanetik, N. 2002. Discovery of Frequent Patterns in Semi-structured Data, Department of Computer Science, Ben Gurion University.

Vanetik, N., Shimony, S.E. and Gudes, E. 2006. Support Measures for Graph Data, Journal of Data Mining and Knowledge Discovery 13(2), 243–260.

Wang, C., Hong, M., Pei, J., Zhou, H., Wang, W. and Shi, B. 2004a. Efficient Pattern-Growth Methods

for Frequent Tree Pattern Mining, In Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining, 441–451.

Wang, C., Wang, W., Pei, J., Zhu, Y. and Shi, B. 2004b. Scalable Mining of Large Disk-based Graph Databases, In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 316–325.

Wang, J., Zeng, Z. and Zhou, L. 2006. CLAN: an Algorithm for Mining Closed Cliques from Large Dense Graph Databases, In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 797–802, Philadelphia, USA.

Washo, T. and Motoda, H. 2003. State of the Art of Graph-based Data Mining, SIGKDD Explorations 5, 59–68.

West, D.B. 2000. Introduction to Graph Theory, 2nd Edition, Prentice Hall.

W¨orlein, M., Meinl, T., Fischer, I., and Philippsen, M. 2005. A Quantitative Comparison of the Subgraph Miners MoFa, gSpan, FFSM and Gaston, In Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, 392–404, Porto, Portugal.

Xin, D., Cheng, H., Yan, X. and Han, J. 2006. Extracting Redundancy Aware Top K Patterns, In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 444–453.

Yan, X. and Han, J.W. 2002. gSpan: Graph-based Substructure pattern mining, In Proceedings of International Conference on Data Mining, 721–724.

Yan, X. and Han, J. 2003. CloseGraph: Mining Closed Frequent Graph Patterns, In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 286–295, Washington D.C., USA.

Yan, X., Yu, P.S. and Han, J. 2004. Graph Indexing: a Frequent Structure-based Approach, In Proceedings of ACM-SIGMOD International Conference on Management of Data, 335–346, Paris, France.

Yan, X., Zhou, X. and Han, J. 2005a. Mining Closed Relational Graphs with Connectivity Constraints, In Proceeding of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, 324–333.

Yan, X., Yu, P.S. and Han, J. 2005b. Sub-Structure Similarity Search in Graph Databases, In Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, 766–777.

Yan, X., Zhu, F., Han, J. and Yu, P.S. 2006. Searching Substructures with Superimposed Distance, In Proceedings of the 22nd International Conference on Data Engineering, 88–97.

Yan, X., Cheng, H., Han, J. and Yu, P.S. 2008. Mining Significant graph patterns by leap search, In

Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, 433–444, Vancouver, Canada.

Zaki, M.J. 2002. Efficiently Mining Frequent Trees in a Forest, In Proceedings of SIGKDD 2002, 71–80, ACM.

Zaki, M.J. and Hsiao, C.J. 2002. CHARM: An Efficient Algorithm for Closed Itemset Mining, In Proceedings of the 2nd SIAM International Conference on Data Mining, 457–473.

Zaki, M.J. and Aggarwal, C.C. 2003. XRules: An Effective Structural Classifier for XML Data, In Proceedings of the 2003 International Conference on Knowledge Discovery and Data Mining, 316–325.

Zaki, M.J. 2005a. Efficiently Mining Frequent Embedded Unordered Trees, Fundamenta Informaticae 66(1-2), 33–52.

Zaki, M.J. 2005b. Efficiently Mining Frequent Trees in a Forest: Algorithms and Applications, IEEE Transactions on Knowledge and Data Engineering 17(8), 1021–1035.

Zeng, Z., Wang, J., Zhou, L. and Karypis, G. 2006. Coherent Closed Quasi-Clique Discovery from Large Dense Graph Databases, In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 797–802, Philadelphia, USA.

Zhang, S. and Wang, J.T.L. 2006. Mining Frequent Agreement Subtrees in Phylogenetic Databases, In Proceedings of the 6th SIAM International Conference on Data Mining, 222–233.

Zhang, S. and Yang, J. 2008. RAM: Randomized Approximate Graph Mining, In Proceedings of the 20th International Conference on Scientific and Statistical Database Management, 187–203.

Zhao, P. and Yu, J. 2006. Fast Frequent Free Tree Mining in Graph Databases, In Proceedings of the 6th IEEE International Conference on Data Mining Workshop, 315–319.

Zhao, P. and Yu, J. 2007. Mining Closed Frequent Free Trees in Graph Databases, In Proceedings of the 12th International Conference on Database Systems for Advanced Applications, 91–102, Thailand.

Zhu, F., Yan, X., Han, J. and Yu, P.S. 2007. gPrune: A Constraint Pushing Framework for Graph Pattern Mining, In Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining, 388–400.