دانشگاه آزاد اسلامی - واحد علوم و تحقیقات

دانشکده مهندسی کامپیوتر

سمینار دوره کارشناسی ارشد مهندسی کامپیوتر – گرایش نرم افزار

بررسی روش ها و الگوریتم های گراف کاوی

نگارش

ایمان رضایی پور

استاد راهنما

دکتر محمد صنیعی

زمستان ۱۳۹۳


بسم الله الرحمن الرحيم

تقدیم

به

پدر فداکارم

کوهی استوار و حامی من در تمام مراحل زندگی

مادر مهربانم

سنگ صبوری که الفبای زندگی به من آموخت.

تشکر و قدردانی

با سپاس فراوان از استاد راهنمای فرهیخته ام جناب آقای دکتر محمد صنیعی که در طول مدت انجام این سمینار از رهنمود های علمی و اخلاقی ایشان بهره مند شدم و درگاه خداوند بزرگ را شاکرم که افتخار شاگردی ایشان را نصیبم نمود.

چکیده

استخراج و مدیریت گراف به خاطر کاربردهای متعددی که در دامنه وسیعی از حوزه های حقیقی از قبیل زیست کامپیوتری، مکان یابی عیب یاب های نرم افزار و ایجاد شبکه های کامپیوتری دارد، به قلمرو تحقیقی متداولی در سال های اخیر تبدیل شده است. برنامه های کاربردی مختلف منجر به پیدایش گراف هایی با اندازه ها و پیچیدگی های گوناگون شده اند. بر همین مقیاس، برنامه های کاربردی دارای الزامات مختلفی برای الگوریتم های استخراج اصلی هستند. در این فصل، ارزیابی از انواع مختلف الگوریتم های استخراج و مدیریت گراف در اختیار شما قرار می گیرد. همچنین، تعدادی از برنامه های کاربردی که وابسته به بازنمایی گراف هستند را مورد بحث قرار خواهیم داد. به این نکته خواهیم پرداخت که چگونه الگوریتم های مختلف استخراج گراف برای کاربردهای مختلف برگزیده و اعمال می شوند. در نهایت، مسیرهای مهم برای تحقیقات آتی در این حوزه را بررسی خواهیم کرد.

این فصل به شکل زیر سازماندهی شده است. در بخش بعدی، انواعی از الگوریتم های مدیریت داده های گراف را بررسی خواهیم کرد. در بخش ۳، الگوریتم های استخراج داده های گراف را مورد بحث قرار خواهیم داد. انواعی از حوزه های کاربردی که در آنها این الگوریتم ها استفاده می شود را در بخش ۴ بررسی خواهیم کرد. بخش ۵ به بررسی نتیجه گیری و خلاصه مطالب می پردازد. راهنمایی های تحقیقی تکمیلی در همین بخش مورد بحث قرار خواهد گرفت.

چکیده

استخراج گراف قلمرو تحقیقی مهمی در حوزه استخراج داده است. زمینه تحقیق بر تعیین زیر گراف های پرتکرار[۱] 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.

واژگان کلیدی: استخراج گراف، مدیریت گراف

[۱] frequent subgraphs

فهرست مطالب

عنوان **صفحه**

Contents

فهرست جدول ها

عنوان **صفحه**

فهرست شکل ها

عنوان **صفحه**

فصل اول: آشنایی با گراف

۱,۱. مقدمه

تمرکز این فصل بر مسئله معرفی گراف و روش های استخراج آن است. ساختار اصلی داده ها می تواند به گراف کلی که وجود حلقه در آن مجاز است، تعلق داشته باشد. اینگونه تصاویر به فرد اجازه می دهد ابعاد پیچیده و دشوار این حوزه مانند ترکیب های شیمیایی، شبکه ها، وب، بیوانفورماتیک و ... را طراحی و مدل سازی کند. به طور کلی، گراف ها با توجه به پیچیدگی های الگوریتمی دارای ویژگی های تئوریک (نظری) نامطلوب فراوانی هستند. در مسئله استخراج گراف، شمارش و بررسی دقیق زیر گراف های یک گراف خاص است که با عنوان مسئله استخراج زیر گراف های پرتکرار شناخته می شود. براساس روش تحلیل گراف موجود، تمرکز خود را بر مسئله فوق که لازمه بررسی پیوند های جالب در میان داده های دارای ساختار گراف است و کاربردهای بسیاری دارد، محدود می کنیم . برای بررسی و ارزیابی همه جانبه استخراج گراف در چارچوب کلی از جمله قوانین مختلف، مولد ها و الگوریتم های داده ها و ... لطفا به [۱, ۲] مراجعه کنید. با توجه به وجود حلقه ها در گراف، مسئله استخراج گراف های پرتکرار بسیار پیچیده تر و دشوارتر از مسئله استخراج زیر مجموعه های درختی است. با وجود اینکه به لحاظ مسئله استخراج گراف یک مسئله NP - کامل است، در عمل تعدادی از رویکردها برای تحلیل داده های گراف حقیقی قابل اجرا است. در ادامه تعدادی از این رویکرد ها و نگرش های مختلف بر مسئله استخراج زیر گراف های پر تکرار و بعضی از رویکردها برای تحلیل کلی داده های گراف را بررسی خواهیم کرد.

۱,۲. مفاهیم و تعاریف کلی گراف

گراف مجموعه ای از گره هاست که هر یک از این گره ها را می توان به گره ای دیگر و حتی خود گره متصل کرد. انواع زیادی گراف وجود دارد از جمله: مستقیم / غیر مستقیم، وزنه ای، متناهی / نامتناهی، منظم (متقارن) و گراف های کامل. این نوع از گراف ها اغلب برای کاربردهای ویژه و تخصصی که در آن رابطه ها یا الزامات خاصی حفظ می شود یا حفظ آنها تقویت و تایید می شود، ایجاد شده اند. با این وجود، اکثر مدارک و اسناد نیمه - سازمان یافته ای که ساختار اصلی آنها نوعی گراف است را می توان بصورت یک گراف با کران های غیر مستقیم و بدون برچسب مدل سازی کرد. از این نظر، گراف را می توان بصورت $G = (V, L, E)$ تعریف کرد، که (۱) V مجموعه رأس ها یا گره ها است; (۲) L یک تابع برچسب زن است که به هر رأس $v \in V$ یک برچسب L(v) نصب می کند; و (۳) $E = \{(V_1, V_2) \mid V_1, V_2 \in V\}$ مجموعه ای از کران ها در مجموعه G است. تعداد گره ها در G به عنوان مرتبه G نامیده می شود در حالی که تعداد کران ها $|E|$ V به عنوان اندازه G معرفی می شود. هر کران معمولا دو گره به همراه دارد ولی ممکن است کران فقط دارای یک گره باشد در حالتی که یک کران از یکی از گره ها به طرف خودش کشیده شود (مانند یک حلقه). اگر دو رأس V_1 , V_2 به یک کران متصل شده باشند؛ انگاه گفته می شود که آنها مجانب یکدیگرند، و در غیر اینصورت این دو رأس غیر مجانب یا مستقل خوانده می شوند. دو کران که در یک رأس به هم می رسند مجانب یکدیگر هستند و در غیر اینصورت غیر مجانب اند. به عبارتی دیگر، اگر دو کران (V_{a1}, V_{a2}) و (V_{b1}, V_{b2}) مجانب یکدیگر باشند و $(V_{a1} \neq V_{a2})$ و $(V_{a1} \neq V_{a2})$، آنگاه یکی از عبارت های دیگر صدق می کند: $(V_{a1} = V_{b1})$ یا $(V_{a1} = V_{b2})$ یا $(V_{a2} = V_{b1})$ یا $(V_{a2} = V_{b2})$.

مجموعه همه رأس های مجانب رأس V با مجانب های V متناظر است. یک مسیر بصورت توالی متناهی از کران ها بین هر دو گره تعریف می شود و در گراف بر خلاف گراف درختی اگر یک مسیر خاص بین دو گره وجود داشته باشد، ممکن است مسیرهای چندگانه ای وجود داشته باشد، طول مسیر P به تعداد کران ها در P گفته می شود. مرتبه یک گره به تعداد گره های صادر شده از یک گره گفته می شود.

گراف G' زیر گراف G خواهد بود، اگر هم ریختی زیر گراف بین G' و G وجود داشته باشد. هم ریختی عبارت است از تناظر یک به یک مجموعه گره های یک گراف با مجموعه گره های گراف دیگر در شرایطی که مجانب یا غیر مجانب بودن محفوظ باشد.

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. گراف های مورد استفاده در زیر گراف های پر تکرار به عنوان گراف های ساده برچسب دار تلقی می شوند. در زیر گراف های بعدی، تعدادی از تعاریف بسیار متداول 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 ϑ که تابع بر چسب است که تناظر $V \to L_V$ ، $E \to L_E$ را تعریف می کند.

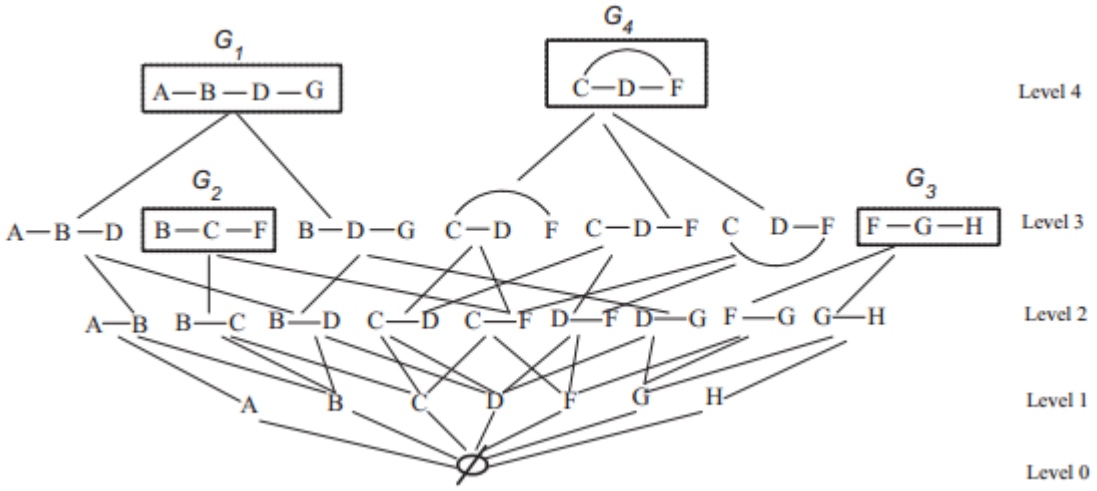
G گراف غیر مستقیم خواهد بود اگر $\forall e \in E$، که e یک جفت رأس نامنظم است. مسیر در G عبارتست از توالی رأس هایی که قابل مرتب کردن هستند به صورتی که دو رأس تشکیل یک کران را می دهند اگر و تنها اگر این دو رأس در لیست به صورت متوالی قرار گرفته باشند. G پیوسته است، اگر شامل مسیری برای هر جفت رأس موجود 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 [۳].

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 $f: V_1 \to V_2$ 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 [۳]. 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 [۴]. 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 مادر رأس q در مشبک است، اگر q زیر گراف p باشد و q دقیقاً در یک کران با p تفاوت دارد. رأس q زاده p است. تمام

زیر گراف های هر گراف $\varsigma \in G_i$ که در پایگاه داده ها موجود باشند در مشبک وجود دارند و هر زیر گراف فقط یک بار در آن دیده می شود.



شکل ۱: مشبک (ς) (تصویر بر مبنای تصویر مشابهی که در [۴] 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 شکل ۱ 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 [۵, ۶].

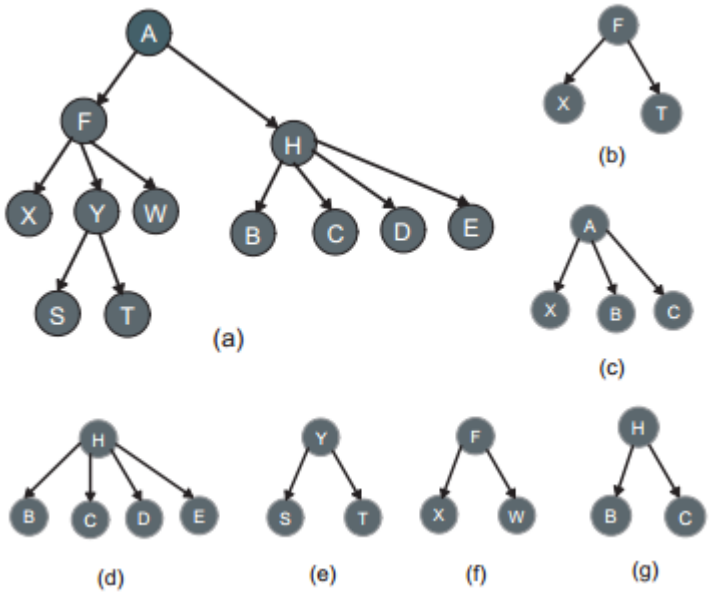
گراف درختی نامنظم برچسب دار: گراف نامنظم برچسب دار (به طور خلاصه گراف درختی نامنظم) یک گراف مستقیم فاقد حلقه است که به صورت T $(V, \emptyset, E, u_r)$ بیان می شود، که V مجموعه ای از رأس های T است؛ $\emptyset$ یک تابع برچسب دار است به صورتی که $\forall u_i \in V, \emptyset(V_i) \rightarrow V_i$؛ و $E \subseteq V \times V$ مجموعه ای از کران های 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 $(u_r, u_1, u_2, \dots, u_i)$ 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، و u_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 می شوند. اندازه T با توجه به تعداد رأس های موجود در T تعیین می شود. هر رأس بدون زاده یک برگ است، در غیر این صورت 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 [۵, ۶].

گراف درختی منظم برچسب دار: گراف درختی منظم برچسب دار 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 [۷, ۸]

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)* اگر $\forall u \in \vec{V}$، u 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 [۶].

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 [۶, ۹]

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.



شکل ۲: انواع مختلف گراف های درختی

جدول ۱: 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
VF₂	مبنای منطقی *VF₂* + ساختارهای پیشرفته داده ها	هم ریختی گراف و زیر گراف

شکل ۲ برای خلاصه کردن جدول ۱، 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*) نشان دهنده 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*.

۱,۳. مسئله هم ریختی گراف

دو گراف $G_1(V_1, L_1, E_1)$ و $G_2(V_2, L_2, E_2)$ هم ریخت یکدیگر گفته می شوند اگر رابطه $f : V_1 \rightarrow V_2$ صدق کند به صورتی که $(V_1, V_2) \in E_1$ اگر $\big(f(V_1), f(V_2)\big) \in E_2$. بنابراین، دو گراف برچسب دار $G_1(V_1, L_1, E_1)$ و $G_2(V_2, L_2, E_2)$ هم ریخت یکدیگرند اگر تناظر یک به یک از V_1 به V_2 وجود داشته باشد به صورتی که رأس ها، برچسب رأس ها و مجانب یا غیر مجانب بودن رأس ها حفظ شود.

به طوری که پیش تر گفته شد، هر گراف می تواند زیر گراف یک گراف دیگر باشد اگر هم ریختی زیر گراف بین آن دو صدق کند. به طور صریح تر، این مسئله را می توان به صورت زیر بیان کرد:

گراف $G_s(V_s, L_s, E_s)$ زیر گراف $G(V, L, E)$ است اگر و تنها اگر $V_s \subseteq V$ و $E_s \subseteq E$.

مسئله استخراج زیر گراف پر تکرار را می توان بطور کلی به این صورت بیان کرد: با ارائه پایگاه داده های G_{DB} و آستانه حفاظت مینیمم (σ)، می توان همه زیر گراف هایی که دست کم σ بار در G_{DB} وجود دارند را استخراج کرد.

علاوه بر تعریف کلی زیر گراف که در بخش قبلی ارائه شده سایر زیر گراف های متداول عبارتند از:

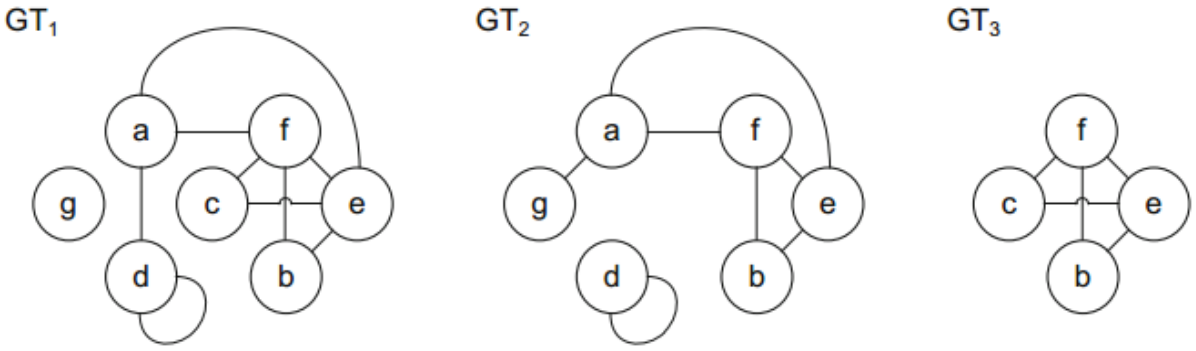
دربرگیرنده، پیوسته و القا شده.

گراف $G_s(V_s, L_s, E_s)$ زیر گراف دربرگیرنده گراف $G(V, L, E)$ است اگر و تنها اگر $V_s = V$ و $E_s \subseteq E$.

گراف $G_s(V_s, L_s, E_s)$ زیر گراف پیوسته گراف $G(V, L, E)$ است اگر و تنها اگر $V_s \subseteq V$ و $E_s \subseteq E$ و تمام رأس های V_s به طور متقابل از طریق کران های E_s قابل دستیابی باشند.

گراف $G_s(V_s, L_s, E_s)$ زیر گراف القا شده گراف $G(V, L, E)$ است اگر و تنها اگر $V_s \subseteq V$ و $E_s \subseteq E$ و تناظر $f : V_1 \rightarrow V_2$ صدق کند به طوری که برای هر جفت رأس $V_x, V_y \in V_s$. اگر کران $\left(f(V_x), f(V_y)\right) \in E$ وجود داشته باشد آنگاه $(V_x, V_y) \in E_s$.

برای روشن شدن این ابعاد، لطفا به شکل ۳ نگاه کنید که یک نمونه پایگاه داده گراف G_{DB} شامل سه فعالیت را نشان می دهد.



شکل ۳: نمونه ای از پایگاه داده گراف G_{DB} شامل سه فعالیت GT_1 , GT_2 , GT_3.

هسته اصلی جستجوی زیر گراف های پرتکرار، 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. وقتی گراف ها را به گراف های درختی محدود می کنیم، ارزیابی هم ریختی (زیر) گراف تبدیل به ارزیابی هم ریختی گراف درخت فرعی می شود. ارزیابی هم ریختی گراف درختی در زمان تک بعدی (خطی) قابل حل است. الگوریتم های سریع تر در زمینه ارزیابی هم ریختی گراف های درختی فرعی، با دشواری بدترین حالت زمان $O(k^{1/5} n)$، پیشنهاد شدند، و پس از آن به صورت $O(\frac{k^{\frac{1}{5}}}{10gk}n)$ ارتقاء یافتند (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 جستجوی زیر گراف پر تکرار 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 [۱۰] یا در مقوله تناظر خطای مجاز گنجاند. اکثر الگوریتم های جستجوی زیر گراف های پر تکرار از تناظر دقیق استفاده می کنند. طبقه بندی مهم ترین الگوریتم های تناظر دقیق برای ارزیابی هم ریختی زیر گراف در جدول ۱ داده شده است. در جدول ۱، 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.

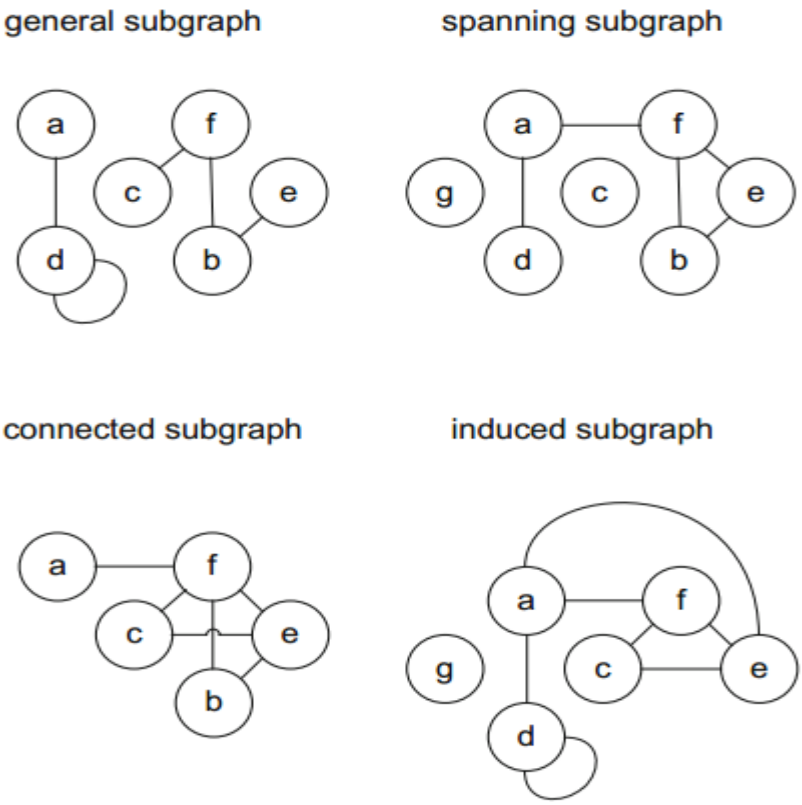
با مراجعه به جدول ۱، الگوریتم اولمان از یک پروسه عقب نشینی با تابع برنامه ریزی برای کاهش اندازه فضای جستجو استفاده می کند. الگوریتم *SD*، در عوض، از یک ماتریکس فاصله معرف یک گراف به همراه پروسه عقب نشینی برای کاهش جستجو بهره می گیرد. الگوریتم نوتی از تئوری گروه برای تغییر شکل و تبدیل گراف ها برای تطابق 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 [۱۱] که ساخت اشکال و فرم

های مجاز می تواند منجر به پیچیدگی تصاعدی در بدترین حالت آن شود. اگر چه الگوریتم نوتی [۱۱] به عنوان سریع ترین الگوریتم هم ریختی گراف شناخته شد، اما مشاهدات نشان داد که بعضی از ویژگی های گراف ها نیازمند زمان فزاینده ای برای ایجاد برچسب مجاز هستند. الگوریتم های *VF* و *VF₂* [۱۰] 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 [۱۲] 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.

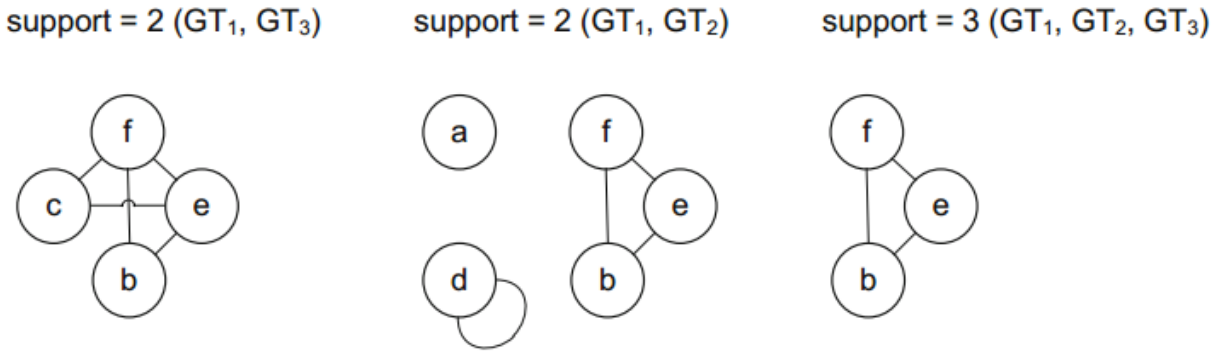
در این بخش، تمرکز خود را بر متداول ترین تعاریف زیر گراف در مسئله استخراج الگوی پر تکرار از دادهای دارای ساختار گراف در حوزه استخراج داده ها محدود می کنیم. تعداد بسیار کمی گونه ها و شکل های مختلف گراف وجود دارد که در مسئله تحقیق گسترده تحلیل داده های گراف قابل بررسی و ملاحظه هستند و همچنین انواع مختلفی از معیارها و الزامات (محدودیت ها) که می توان بر فرایند تحلیل داده ها تحمیل کرد. در بخش بعدی، روش های گوناگونی را که برای مسئله استخراج گراف پر تکرار ایجاد شده اند مورد بررسی قرار می دهیم. در بعضی از این روش ها الزاماتی و محدودیت هایی وضع شده است تا تحلیل را به سمت هدف کاربردی ویژه ای سوق دهد یا تعداد الگوهای شمارش شده را کاهش دهد تا پیچیدگی و دشواری حاصل از استخراج داده های گراف را کم کند.

۱,۴. روش های موجود برای استخراج گراف

تعدادی روش های استخراج داده های گراف – محور طراحی و ایجاد شده اند. در این بخش، عمدتا بر روش هایی که برای انجام عملیات استخراج زیر گراف های پرتکرار طراحی شده اند تمرکز می کنیم. در انتهای این بخش تعدادی از روش های ایجاد شده برای رفع مسائل مرتبط با استخراج زیر گراف های استخراج و به طور کلی تحلیل داده های گراف را ارزیابی می کنیم.



شکل ۴: نمونه هایی از زیر گراف های متعلق به گراف فعالیت GT از پایگاه داده های G_{DB}



شکل ۵: تعدادی از زیر گراف های عمومی پر تکرار از پایگاه داده های G_{DB}

۱,۴,۱. روش های آپریوری – مانند

این بخش بر روش های استخراج گراف که با استفاده از اصول و قواعد استخراج مجموعه آیتم / توالی / درختچه های گراف پرتکرار بر مبنای روش آپریوری تمرکز کرده است. فرایند شمارش و بررسی به شیوه سروته انجام می شود و با زیر گراف هایی که

شامل (مثلا زیر گراف –۱) آغاز می شود. در هر چرخه (تکرار)، زیر گراف داوطلب K از نظر پر تکرار بودن مورد بررسی قرار می گیرند و فقط الگوهای پرتکرار برای ایجاد زیر گراف های (K+1) مورد استفاده قرار می گیرند. تعدادی از واریانس های مختلف در میان الگوریتم های آپریوری - مبنا در خصوص روش ایجاد داوطلب ها (گراف های داوطلب) وجود دارد. رویکرد اتصال که برای استخراج مجموعه آیتم های پرتکرار به خوبی کار می کند ممکن است برای برنامه های کاربردی که ویژگی های ساختاری الگوهای داده در نظر گرفته می شوند، مناسب و کارآمد نباشد. تعداد زیادی داوطلب نامعتبر بی جهت ایجاد و فرض می شوند. در استخراج زیر گراف پرتکرار با توجه به وجود روش های بسیاری برای عملیات اتصال دو زیر ساختار پدیده فوق بیشتر مشاهده می شود.

الگوریتم AGM [۱۳] رویکرد آپریوری سطح – محور را بر می گزینند که گراف ها با استفاده از یک ماتریکس مجانب ارائه می شوند. فرایند مورد نظر از کوچک ترین زیر گراف ها شروع می شود و داوطلب ها با اجرای عملیات اتصال برای ماتریکس های مجانب نشان دهنده زیر گراف های داوطلب بررسی و شمارش می شوند. بر اساس دسته بندی ماتریکس های مجانب که در تعیین دقیق فراوانی نقش دارند، شکل مجاز برای زیر گراف القا شده تعریف می شود. الگوریتم FSG [۱۴] برای استخراج زیر گراف های مستقیم / غیر مستقیم پر تکرار (فراوان) طراحی شد. برای به حداقل رساندن فرایند ذخیره سازی و محاسبه الگوریتم های FSG برای ذخیره موثر فعالیت های ورودی، زیر گراف های داوطلب و زیر گراف های پر تکرار از نمایش گراف کم تراکم (غیر متراکم) استفاده می کند. این الگوریتم ها شکل مجاز ماتریکس مجانب را اجرا می کنند و سپس آن را به نمایش فهرست – مجانب ها تبدیل می کند. برای ایجاد زیر گراف های داوطلب (K+1) زیر گراف پر تکرار K، عملیات اتصال برای زیر گراف های پر تکرار K که شامل زیر گراف (K-1) مشابهی است اجرا می شود. فراوانی گراف (K+1) جدید به صورت اندازه مقطع فهرست های شناساگر اجرایی (فهرست TID) زیر گراف های متصل K تعریف می شود. استفاده از فهرست های TID می تواند فرایند بررسی و شمارش زیر گراف های داوطلب در FSG را ساده تر کند اما لزوم ذخیره تمام فهرست های TID برای داده های گراف بزرگ موجب ایجاد مشکلات حافظه می شود. همین نویسندگان [۱۵] الگوریتم gFSG را پیشنهاد داده اند که دنباله مسئله یافتن الگوهای هندسی پرتکرار در گراف های هندسی است. این گراف های هندسی در واقع گراف هایی هستند که رأس های آنها دارای مختصات دو یا سه بعدی هستند. gFSG از یک رویکرد سطح - محور استفاده می کند که در هر نوبت به وسیله یکی از کران ها، زیر گراف های پر تکرار را امتداد می دهد و تعدادی از الگوریتم ها برای ارزیابی هم ریختی زیر گراف های هندسی تلفیق (ادغام) می شوند (که عبارتند از: دوران، مرتبه، واریانس برگردان).

یکی دیگر از الگوریتم های آپریوری - محور در [۱۶] ارائه شد. این الگوریتم از نمایش های مجاز مسیرها و توالی مسیرها استفاده می کند و نظام واژگان نگاری روی جفت مسیرها بر اساس برچسب های گره و مرتبه گره های درون مسیر تعیین می شود. گراف به عنوان گروهی از مسیرهایی با کران های قطعه قطعه شده بیان می شود که دو مسیر در صورتی دارای کران های قطعه قطعه شده هستند که دارای هیچ کران مشترکی نباشند. رویکرد سروته جایی مورد استفاده قرار می گیرد که در ابتدا، تمام زیر گراف های پر تکراری که دارای مسیر مجزا هستند پیدا شوند و همه آنها ترکیب شوند و هرجا که امکان داشته باشد زیر گراف هایی دارای دو مسیر ساخته شود. الگوریتم به تدریج زیر گراف های دارای K مسیر را با اتصال زیر گراف های پرتکرار با K-1 مسیر معین می کند.

۱,۴,۲. روش های توسعه الگو

اشتراک بین الگوریتم هایی که از رویکردهای شبیه به آپریوری استفاده می کنند این است که زیر گراف ها به طور منظم به روش Bottom-Up معین می شوند در حالی که وقتی داده ها بسیار بزرگ باشند یا رابطه ساختاری نسبتا پیچیده باشد مسائل

مختلفی به وجود خواهد آمد. این مورد به ویژه در شرایطی روی خواهد داد که زیر گراف های داوطلب با استفاده از رویکرد اتصال ایجاد می شوند. در شرایطی که دو زیر گراف قابل اتصال باشند احتمالات زیادی وجود خواهد داشت و شکل های ساختاری یک زیر گراف داوطلب همواره در پایگاه داده وجود نخواهد داشت. هم ریختی زیر گراف یک آزمایش پر هزینه است و به همین خاطر، ایجاد زیر گراف های داوطلبی که می بایست در ادامه از بین بروند، بیهوده است. این مشکلات، انگیزه ها برای طراحی تعدادی از الگوریتم ها که برای به حداقل رساندن زیر گراف های داوطلب غیر ضروری از رویکرد ساختار گراف هدایت شونده استفاده می کنند، را تقویت می کند.

الگوریتم gSpan [۱۷] نخستین رویکردی بود که از جستجوی عمقی برای زیر گراف های پر تکرار استفاده کرد. این جستجو توسط سیستم برچسب گذاری مجاز نوینی پشتیبانی می شود. هر گراف با یک کد زنجیره ای منطبق می شود و تمام کدها بر حسب ترتیب واژگان نگاری افزایشی دسته بندی می شوند. سپس جستجوی عمقی برای درخت هایی که با اولین گره های کد ها جفت می شوند اعمال می شود، و به تدریج داده های پرتکرار بیشتری اضافه می شوند. وقتی پشتیبانی که یک گراف به پایین تر از حداقل پشتیبانی می رسد یا در شرایطی که زیر گراف قبلا شناسایی شده باشد آنگاه این زیر گراف نقاط توقف را افزایش می دهد. الگوریتم gSpan از نظر زمان و حافظه مورد نیاز کارایی زیادی دارد.

الگوریتم ارائه شده در [۱۸]، در شناسایی زیر ساختارهای پرتکرار در ترکیب های مولکولی تمرکز می کند و همچنین از استراتژی جستجوی عمقی برای پیدا کردن زیر گراف های پرتکرار استفاده می کند. در پی فرایند افزایش الگو، وقوع گسستگی درون تمام مولکول ها در تعادل نگه داشته می شود. ترتیب اتم ها و پیوندهای محلی برای از بین بردن گسست های غیر ضروری که سرعت جستجو را پایین می آورند مورد استفاده قرار می گیرند و از ایجاد الگوهای زاید جلوگیری می کنند. الگوریتم FFSM [۳] برای تعیین صریح زیر گراف های پرتکرار از چارچوب گراف های جبری استفاده می کند. هر گراف با استفاده از ماتریکس تجانب بیان می شود و با بهره گیری از تثبیت کننده های هر زیر گراف پر تکرار از هم ریختی زیر گراف جلوگیری می شود. الگوریتم GASTON (گراف، توالی، استخراج نمودارهای درختی) [۱۹] با استفاده از ایده آغاز سریع جستجوی ساختارهای گراف پرتکرار با تلفیق جستجوی مسیرها، درخت ها و گراف های پرتکرار در یک رویکرد خاص طراحی شد. رویکرد سطح - محور مورد استفاده قرار می گیرد که مسیرهای ساده تعیین شود و به دنبال آن ساختارهای درختی و اکثر ساختارهای گراف پیچیده در انتها تعیین شوند. این نوع رویکرد با این تفکر تقویت می شود که در عمل اکثر گراف ها واقعا پیچیده و دشوار نیستند و تعدادی از حلقه ها خیلی بزرگ نیستند. بنابراین، در مرحله تعیین و شمارش، ابتدا توالی ها / مسیرها تعیین می شود پس از آن ساختارهای درختی با اضافه شدن مداوم گره و کران همراه آن به گره های مسیر تعیین می شوند. سپس، ساختارهای گراف با اضافه کردن کران ها در میان گره های ساختار درختی تعیین می شوند.

۱،۴،۳. روش های برنامه ریزی منطقی استقرایی (IPL)

رویکردهایی که در این دسته قرار می گیرند با استفاده از IPL برای بیان داده های گراف با استفاده از جملات برجسته معرفی می شوند. آنها از حوزه استخراج داده های چند رابطه ای می آیند، که در اصل استخراج داده هایی است که در جدول داده های در هم آمیخته چند گانه پایگاه داده های رابطه ای سازماندهی شده اند.

الگوریتم WARMR برای استخراج پرسش های پرتکرار از پایگاه داده های DATALOG طراحی شده است. این الگوریتم در مقایسه با روش های استاندارد استخراج الگوهای پرتکرار از یک پارامتر (کلید) اضافی استفاده می کند که این پارامتر در اصل

خاصیتی است که می بایست در تمام الگوهای استخراج شده گنجانده شود. این الگوریتم به کاربر اجازه می دهد فضای جستجو را به الگوهایی که دارای آیتم مطلوب هستند، محدود کند. الگوریتم مذکور از نظر نوع الگوهای قابل استخراج انعطاف پذیر است (محدودیت ندارد). زبان تسریع کننده ای (WRMODE) طراحی شده است که فضای جستجو را به پرسش های قابل قبول و بالقوه جالب محدود می کند. فرایندهای اصلی الگوریتم عبارتند از ایجاد داوطلب و ارزیابی داوطلب. مرحله ارزیابی داوطلب برای تعیین فراوانی داوطلب های پرتکرار مورد استفاده قرار می گیرد. در مرحله ایجاد داوطلب، الگوهایی که شامل کلیدهای معین هستند پیدا می شوند و به طور فزاینده ای گسترده می شوند، این کار از کوچکترین الگوها آغاز می شود. در هر مرحله، الگوهای پرتکرار، کم تکرار (نادر) تعیین می شوند و الگوهای پرتکرار با اضافه کردن یک گره در هر نوبت گسترده می شوند. الگوهایی که قبلا در مجموعه الگوهای پرتکرار قرار گرفته اند حذف می شوند، یا اگر شکل تخصصی شده الگویی باشند که قبلا در مجموعه الگوهای پرتکرار وجود داشته، حذف می شوند. این روش در مورد تحلیل هشدارهای مخابراتی و سم شناسی شیمیایی مورد استفاده قرار گرفت. نقطه ضعف عمده الگوریتم به کارایی آن بر می گردد چرا که آزمایش تعادل بین بند های ردیف اول بسیار دشوار است. الگوریتم FARMER [۱۹] هنوز هم از نشانه گذاری منطقی ردیف اول استفاده می کند و از بسیاری جهات شبیه به WARMR است، و از لحاظ زمان انجام عملیات پیشرفت قابل ملاحظه ای داشته است. تفاوت اصلی این دو الگوریتم این است که به جای آزمایش های پر هزینه تعادلWARMR برای محاسبه و ایجاد پرسش های کاندیدا به شیوه ای موثر، از ساختار داده های درختی استفاده می کند. روشی را برای یافتن ذرات مولکولی پر تکرار از پایگاه داده های ترکیب های مولکولی طراحی شد [۲۰]، که یک ذره مولکولی به صورت توالی اتم هایی که به صورت خطی به یکدیگرند متصل شده اند تعریف می شود، و پایگاه داده ها شامل اطلاعاتی درباره ویژگی هایی از اتم های یک مولکول و ترتیب پیوند هاست. این رویکرد، تعیین ضابطه کلی را امکان پذیر می کند به گونه ای که تمام ذرات مولکولی اجتماع ضابطه های اولیه را عملی می کنند. علاوه بر پارامتر فراوانی مینیمم سنتی، رویکرد آنها استفاده از ضابطه فراوانی ماکزیمم برای الگو ها را امکان پذیر می کند. این ضوابط قابلیت انعطاف پذیری بیشتری به آن دسته از پرسش ها می دهد که از طریق الگو ها می توان به آن ها پاسخ داد، و در حوزه پیدا کردن ذره های مولکولی، می توان الگو های جالب تری پیدا کرد. رویکرد IPL - محور دیگری ارائه شده است [۲۱] و یک زبان آمیخته و ترکیبی برای حفظ اطلاعات مربوط به ویژگی های ارتباطی و ساختاری در قوانین چند سطحی استخراج شده از روابط چندگانه پیشنهاد شد. برای تعیین ترتیب کلی و اپراتور اصلاح نزولی از رابطه های زیر گروهی پرسش ها استفاده می شود .

۱,۴,۴. روش های جستجوی سختگیرانه

ویژگی های روش های جستجو – محور سختگیرانه این است که آنها از محاسبات مفرط و اضافی مورد نیاز برای جستجوی تمام زیرگراف های پرتکرار اجتناب می کنند و به همین خاطر از رویکرد سختگیرانه برای کاهش تعداد زیرگراف های بررسی شده استفاده می کنند. در ازای از دست دادن تعدادی از زیرگراف های پرتکرار، از پیچیدگی بیش از حد مسئله هم ریختی گراف جلوگیری به عمل می آید.

یکی از نخستین رویکردهای استخراج گراف تحت عنوان سیستم SUBDUE ساخته می شود و بسیاری از اصلاحات و بهینه سازی های این سیستم انجام شده است [۲۲-۲۵]. سیستم SUBDUE بر اصل اندازه مینیمم توصیف (MDL) استوار است، که به صورت تعداد بیت های لازم برای توصیف گراف اندازه گیری می شود. تعاریف مفهومی جایگزین نمونه های زیر گراف شناسایی شده می شود. این رویداد باعث فشرده شدن مجموعه داده های اصلی می شود و مبنایی برای شناسایی ساختارهایی که به صورت طبقاتی تعریف شده اند، فراهم می آورد. انگیزه این گونه روش ها، تعیین زیر ساختارهای به لحاظ ذهنی جالبی است که تفسیر

داده ها را افزایش می دهند. سیستم SUBDUE در کنار استفاده از اصل MDL، امکان تلفیق سایر اطلاعات و یافته های پیشین برای تمرکز بر جستجوی زیرگراف های مناسب تر را به وجود می آورد. رویکرد جستجوی سختگیرانه (حریص) برای شناسایی زیر گراف های داوطلب از داده های موجود است. این رویکرد از گره های مجزا (تکی) شروع می شود و متناوبا زیر ساختارهایی با یک کران مجاور را تعمیم می دهند تا جایی که تمام گسترده های ممکن را پوشش بدهد. هر یک از زیر گراف های داوطلب جدید تعمیم (گسترش) داده می شود و الگوریتم برای شناسایی بهترین زیر گراف ها بر طبق اندازه مینیمم توصیف، تمام زیرگراف های ممکن را مورد بررسی قرار می دهد. هنگامی که تمام زیر گراف های ممکن مورد بررسی قرار بگیرند یا فرایند جستجو به محدودیت محاسبه برسد آنگاه الگوریتم متوقف خواهد شد. برای جلوگیری از تعمیم از زیر گراف هایی که اندازه توصیفی آنها افزایش خواهد یافت از تکنیک حذف انتخابی استفاده می شود. سیستم SUBDUE در جستجوی خود برای زیر گراف های نظیر، از الگوریتم جفت گراف های غیر دقیق استفاده می کند تا تغییرات ناچیز را امکان پذیر کند زیرا زیر ساختارهای جالب ممکن است اغلب اوقات به شکلی تقریبا متفاوت در داده ها دیده شوند.

همچنین، روش القا گراف (GBI) به منظور دستیابی به زیرگراف های جالب و کوچک، داده های گراف را فشرده می کند. این روش در ابتدا برای پیدا کردن مفاهیم جالب از الگوهای پرتکرار یافته شده در نتیجه گیری طراحی شد. تکنیک فشرده کردن در اصطلاح طبقه بندی جفت - محور خوانده می شود، که دو گره درون یک گره با هم جفت می شوند. ارتباط بین گره های جفت شده از بین می رود، و در صورت لزوم رابط های (تکنیک های) بین سایر گره های گراف به خاطر سپرده می شوند به طوری که در هر زمانی در خلال فرایند جستجو بازسازی گراف اصلی امکان پذیر باشد. قطعه بندی جفت - محور را می توان به صورت مکانی مناسب در نظر گرفت و می توان گراف به طور متناوب فشرده کرد. اندازه کوچک انتخاب می شود تا مقدار فشردگی که بیانگر اندازه الگوهای استخراج شده و گراف فشرده است، محدود شود. جستجوی داوطلب های زیر گراف های محلی با استفاده از جستجوی فرصت طلبانه انجام می گیرد.

۱,۴,۵. سایر روش ها

همه روش های بررسی شده در بخش های قبلی به خانواده روش های استخراج زیر ساختار های (زیر گراف های) پرتکرار تعلق دارند. اندازه جستجو که به سمت تحویل اتوماتیک داده های گراف رفته است به طور کلی بزرگ است [۱, ۲]. این بخش بعضی از رویکرد های جایگزین برای استخراج داده های گراف را ارائه خواهد کرد که اهداف مختلف تحلیل داده ها یا نیازهای نرم افزاری مشخص یا محدودیت های خاص عامل اصلی برای روی آوردن به این رویکرد هاست.

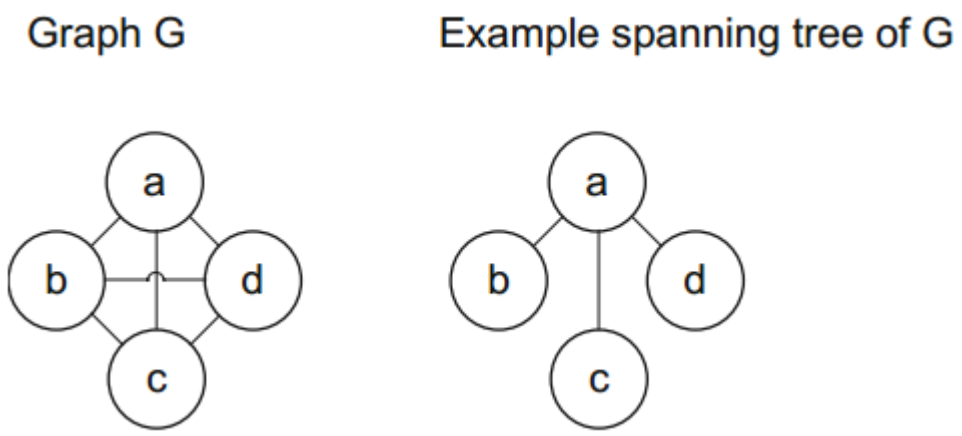
خوشه بندی داده های دارای ساختار گراف عموما در بسیاری از برنامه های کاربردی اهمیت ویژه ای دارند زیرا اغلب خوشه های شناسایی شده اغلب مبنایی برای تحلیل شباهت ها و تفاوت ها به وجود می آورند، و می توان از این خوشه ها برای طبقه بندی داده های گراف بر مبنای ویژگی های خوشه های تشکیل شده استفاده کرد. به عنوان مثال، تکنیک خوشه بندی گراف بر مبنای نظریه جریان شبکه برای تقسیم بندی سلسله تصاویر رزونانس (پژواک) مغناطیسی مغز انسان اعمال شد. داده های مربوط به صورت کران تجانب غیر مستقیم بیان می شود در حالی که به هر کران یک ظرفیت جریان نسبت داده می شود که نشان دهنده شباهت گره های متصل به کران است. خوشه ها با حذف مداوم کران ها از گراف تشکیل می شوند تا وقتی که زیر گراف های متقابلا اختصاصی و منحصر به فرد تشکیل شود. کران ها براساس ظرفیت جریان شان حذف می شوند با این هدف که بزرگترین جریان ماکزیمم در میان زیر گراف های (خوشه های) تشکیل شده به نقطه مینیمم برسد. تعدادی از تکنیک های خوشه بندی برای شناسایی اتوماتیک ساختار قطعه ای یک سیستم نرم افزاری از کد منبع آن طراحی شد. کد منبع به عنوان یک گراف تابع

پیمانه ای بیان می شود و برای شناسایی ساختار سطح بالای سازماندهی سیستم ها از ترکیب خوشه بندی، تپه نوردی و الگوریتم های تکمیلی و ریشه ای استفاده می شود.

تعدادی از الگوریتم های خوشه بندی گراف و بهینه سازی ظابطه خوشه بندی خاصی تمرکز می کنند که در چارچوب خوشه بندی گراف اتفاق می افتد و اغلب اوقات از روش های مربوط به مسائل بهینه سازی گراف های عمومی تر اقتباس می کنند. گراف تشابه تعیین شده و خوشه ها به صورت زیر گراف هایی با قابلیت اتصال بیش از نیمی رأس ها تعریف می شوند و الگوریتم دارای پیچیدگی کمی است. الگوریتم خوشه بندی گراف ارائه شده در [۲۶] بر مبنای ایده کلی تکنیک های جریان ماکزیمم برای به حداکثر رساندن شباهت درون خوشه ای و به حداقل رساندن تشابه میان خوشه ای استوار است. یک سینک مصنوعی در گراف قرار داده می شود که به تمام گره ها متصل است و جریان های ماکزیمم بین تمام گره ها و سینک محاسبه می شوند. برای محدود کردن تعداد کران های مورد استفاده در ایجاد اتصال بین سینک مصنوعی و سایر گره های گراف، یک پارامتر ویژه انتخاب می شود. خوشه بندی بر مبنای گراف های درختی کوتاه شده مینیمم انجام می گیرد. گراف درختی کوتاه شده مینیمم یکی از زیر گراف های گراف اصلی است در شرایطی که تظمین می شود مسیر بین دو گره داده شده در گراف درختی کوتاه شده مینیمم کوتاه ترین مسیر بین این دو گره در گراف اصلی باشد. به همین دلیل، با انتخاب گراف های درختی کوتاه شده مینیمم از گراف های گسترده، الگوریتم حلقه های خاصیت و اجزا به شدت پیوسته (متصل) را شناسایی می کنند. با محاسبه مسیرهای برچسب که در گراف ظاهر می شوند گراف ها به صورت یک بردار اصلی بیان می شوند. مسیرهای برچسب به وسیله تابع های تصادفی روی گراف ایجاد می شوند و هسته اصلی به صورت محصول داخلی بردارهای اصلی بیان می شود که میانگینی از تمام مسیرهای برچسب ممکن است. محاسبه هسته اصلی به مسئله مهمی در یافتن وضعیت ثابت و پایداری از یک سیستم خطی ناپیوسته تبدیل می شود و راه حل به صورت حل معادله های خطی متقارن با یک ماتریکس دارای ضریب پراکنده ظاهر می شود.

اکثر رویکرد های خوشه بندی شرح داده شده قادر به رفع کامل مسائل استخراج زیر گراف های پر تکرار نمی باشند زیرا تظمینی وجود ندارد که زیر گراف های شناسایی شده با استفاده از روش های خوشه بندی تمام زیر گراف های پرتکرار برای پشتیبانی فرض شده را در بر بگیرد. به همین خاطر، این روش ها یک راه حل تقریبی ارائه می دهند و اغلب قادرند بر بعضی از دشواری هایی غلبه کنند که در هنگام نیاز به بررسی کامل تمام زیر گراف های پرتکرار ظاهر شوند.

روش تقریبی دیگری برای استخراج زیر گراف های پرتکرار بر مبنای گراف های درختی دربرگیرنده ارائه شد. همان گونه که گفته شد، زیر گراف دربرگیرنده یک گراف باید شامل تمام رأس های گراف باشد. گراف درختی دربرگیرنده یک زیر گراف دربرگیرنده است که به شکل درخت است (بدون حلقه). به عبارت دیگر زیر گراف درختی دربرگیرنده گراف غیر مستقیم و پیوسته G، منتخبی از کران های G است که همه رأس های G را بر می گیرد و فاقد حلقه است. نمونه ای از زیر گراف های درختی دربرگیرنده در شکل ۶ نشان داده شده است. الگوریتم مانکی [۲۷] بر مبنای این تفکر طراحی شد که در صورت تغییر شکل کران، انعطاف پذیری در بررسی هم ریختی گراف لازم به نظر می رسد به این منظور که الگوهای پر تکرار علیرغم وجود تغییرات در کران ها مورد بررسی قرار بگیرند. انعطاف پذیری در بررسی هم ریختی گراف بر اساس جستجوی گراف های درختی دربرگیرنده پر تکرار مطرح شد. زیرا، در یک زیر گراف درختی دربرگیرنده، تمام رأس ها می بایست نمایش داده شوند در حالی که تا زمانی که زیر گراف مذکور هنوز به شکل گراف درختی است تعدادی از کران ها می توانند غایب باشند. مانکی (Monkey) بر جستجوی عمقی استوار است و در محدوده با هم تلفیق می شوند تا کنترل کران های نامرتب که بر فراوانی الگوهای متناظر اثر می گذارند، امکان پذیر شود.



شکل ۶: نمونه ای از گراف درختی دربرگیرنده از یک گراف *G*

مسئله استخراج پایگاه داده های گراف دیسکت – محور مورد توجه قرار گرفته است. اصل مطلب به ساختار ADI (شاخص تجانب) مربوط است که در شرایطی که نمی توان پایگاه داده های گراف را در حافظه اصلی قرار داد از کارهای اصلی در فرایند استخراج گراف پشتیبانی می شود. الگوریتم gSpan [۱۷] که پیش تر مورد بررسی قرار گرفت برای استفاده از ساختار ADI انتخاب شد، و الگوریتم حاصل یعنی ADI-Mine نشان داده شد تا هنگامی که پایگاه داده های دیسکت - محور بزرگ مورد تردید قرار می گیرند الگوریتم gSpan عملکرد بهتری داشته باشد. الگوریتم ADI-Mine و بعضی از تکنیک های استخراج محدودیت - محور در هم ادغام شدند. تا یک سیستم Graph Miner (استخراج کننده گراف) تشکیل شود [۲۸].

این سیستم یک فاصل گرافیکی کاربر ایجاد می کند به طوری که می توان بعضی از محدودیت ها از جمله محدودیت اندازه الگو، کران / رأس هایی که باید تا نباید در الگو ظاهر شوند، گراف هایی که الگوهایشان باید ابر الگو یا زیر الگو باشند، و محدودیت های ترکیبی را انتخاب کرد . علاوه بر این، سیستم امکان تماشا و تحلیل آسان الگو ها و قابلیت های پرسشی به منظور تمرکز بر الگوهای دارای کاربرد ویژه یا جالب را به وجود می آورند.

روش استخراج گراف متناوبی برای تحلیل اجزا بنیادی پیشنهاد شد [۲۹]، که در آن الگوهای برجسته و مهم به طور فزاینده ای از طریق درخواست های جداگانه برای استخراج گراف جمع آوری می شوند. در هر درخواست استخراج گراف، مقادیر حقیقی به فعالیت های گراف نسبت داده می شوند و الگوهایی که آستانه پشتیبانی مقرر را برآورده می کنند مشخص می شوند. استراتژی جستجو بر مبنای یک الگوریتم شاخه – و - محدوده استوار است که از این گراف درختی کد جستجوی عمقی به عنوان فضای جستجوی مجاز استفاده می کند. الگوها به شکل گراف های درختی سازماندهی می شوند به گونه ای که یک گره کوچک دارای یک زیر گراف الگو از گره مادر (اصلی) است.

با تولید منظم الگوها از ریشه تا برگ های گراف به شیوه ای چرخشی و با استفاده از تعمیم اصلی ترین گره، الگوها مشخص می شوند.

۱,۴,۶. استخراج الگوهای زیر گراف مسدود / بیشینه

استخراج زیر گراف های مسدود / بیشینه پرتکرار نیز مانند سایر مسائل استخراج الگوهای پرتکرار، حوزه مهم در تحقیقات مربوطه است زیرا یکی از روش های کاهش دشواری و پیچیدگی ناشی از تعیین تمام زیر گراف های پرتکرار است.

الگوریتم Close Graph [۳۰]، زیر گراف های پرتکرار مسدود را استخراج می کند و چارچوب کلی آن شبیه به الگوریتم gSpan است با این تفاوت که از تکنیک های انتخاب گراف های مناسب بهره می گیرد تا با کارآمدی بیشتری فضای جستجو را کوچک کند و تنها زیر گراف های مسدود را مشخص کند. این الگوریتم با حذف تمام گره ها و کران های اتفاقی (کم تکرار) آغاز به کار می کند و سپس تمام زیر گراف های پرتکرار شامل یک گره مجزا را تولید می کند. سپس، مجموعه گراف های پرتکرار را با استفاده از جستجوی عمقی و تعمیم گراف اصلی (بیشینه) گسترده می کند.

جستجوی عمق – محور (عمقی) بر مبنای ترکیب واژگان نگاری DFS عمل می کند، یعنی سیستم بر چسب گذاری مجاز نوین ابتدا در الگوریتم gSpan ارائه می شود. نویسندگان براساس هم ارزی وجود زیر گراف های پرتکرار در گراف اصلی یک حالت توقف اولیه تعریف می کنند، به طوری که نیازی به تعمیم یا بررسی تمام گراف ها در راستای دستیابی به ویژگی زیر گراف مسدود نیست. این حالت (شرط) توقف اولیه برای تمام زیر گراف ها معتبر نیست و حالت دیگری برای بررسی این موارد اعمال می شود به این منظور که از مفقود شدن اطلاعات مربوط به زیر گراف های مسدود پرتکرار به طور بالقوه جلوگیری شود. این حالت ها به الگوریتم Close Graph اجازه می دهد حالت هایی را که امکان مسدود شدن تمام زیر گراف های حاصل از گراف پر تکرار وجود ندارد و نیازی به محاسبه وجود ندارد، مورد بررسی قرار دهد.

مسئله استخراج زیر گراف های پرتکرار با محدودیت های اتصال در گراف های ارتباطی در [۳۱] مورد بررسی قرار گرفته است و دو الگوریتم Close Cut و Splat پیشنهاد شده اند. الگوریتم Close Cut یک رویکرد افزایش (گسترش) الگو است، یعنی ابتدا زیر گراف های پر تکرار مشخص می شوند و با افزودن کران های جدید تعمیم داده می شوند. زیر گراف های داوطلب که بر اساس این فرایند ایجاد شده اند باید محدودیت های اتصال تعیین شده و آستانه فراوانی مینیمم را رفع کنند. گراف های داوطلب با اضافه کردن کران های جدید تعمیم داده می شوند و این کار تا زمانی ادامه پیدا می کند که گراف حاصل از افزودن کران ها دیگر پر تکرار نباشد. از سوی دیگر، الگوریتم Splat رویکردی بر مبنای کاهش الگو است که از آن طریق گراف های ارتباطی قطع و تجزیه می شوند تا گراف های دارای پیوستگی (ارتباط) زیاد به دست آید. در هر مرحله، الگوریتم بررسی می کند که آیا گراف جدید تولید شده در مجموعه نتایج وجود دارد، و در این حالت نیازی به ادامه پردازش نیست زیرا نمی توان هیچ یک از گراف های پیوسته مسدود را از گراف داوطلب مشخص کرد.

الگوریتم MARGIN زیر گراف های پرتکرار بیشینه را به شیوه بالا به پایین استخراج می کند. به منظور رکورد گراف از پایگاه داده های گراف، الگوریتم به طور مکرر یکی از کران ها را در هر نوبت حذف می کند بدون اینکه گراف های ناپیوسته (غیر مرتبط) ایجاد کند؛ این کار تا زمانی ادامه پیدا می کند که نمونه پرتکرار گراف مورد نظر پیدا شود. این نمونه پرتکرار احتمالا یک الگوی بیشینه پرتکرار خواهد بود زمانی که تعدادی از کران ها وگره ها اتفاقی حذف شده باشند. به همین خاطر، پروسه ای متناوب بر زیر گراف های پیوسته پرتکرار اعمال می شود که به طور فزاینده ای کران ها و گره های اتفاقی را از زیر گراف تعیین شده حذف می کند و این روند ادامه پیدا می کند تا زمانی که ویژگی های زیر گراف بیشینه پرتکرار برآورده شود.

فصل دوم: الگوریتم ها و کاربردهای گراف

۲,۱. مقدمه

استخراج و مدیریت گراف به واسطه کاربردهای متعددی که در دامنه وسیعی از حوزه های حقیقی از قبیل زیست کامپیوتری، مکان یابی عیب یاب های نرم افزار، و ایجاد شبکه های کامپیوتری دارد، در سال های اخیر به قلمرو تحقیقی متداولی تبدیل شده است، علاوه بر این، انواع جدیدی از داده ها مانند داده های نیمه ـ ساختاری و XML [۳۲] را می توان به شکل گراف ها ارائه کرد.

در حوزه گراف، الزام و نیاز به برنامه های کاربردی مختلف یکنواخت نیست. بنابراین، الگوریتم های استخراج گراف که در یک حوزه عملکرد خوبی دارند ممکن است در حوزه دیگر عملکرد خوبی نشان ندهند. به عنوان مثال، اجازه دهید حوزه های داده های زیر را بررسی کنیم.

- **داده های شیمیایی:** داده های شیمیایی اغلب به صورت گراف هایی بیان می شوند که گره ها متناظر با اتم ها و اتصال ها متناظر با پیوندهای بین اتم ها هستند. زیر ساختارهای داده ها نیز ممکن است به عنوان گره های اختصاصی مورد استفاده قرار بگیرند. در این حالت، گراف های اختصاصی کاملاً کوچک هستند، با این همه، کپی های متعددی در میان گره های مختلف وجود دارند. این موضوع منجر به چالش های هم ریختی در برنامه هایی مثل تناظر گراف می شود. چالش هم ریختی این است که گره ها در یک جفت گراف مشخص از جهات گوناگونی با هم تناظر و شباهت داشته باشند. تعداد تناظرهای موجود کاربردی از جمله استخراج الگوی پرتکرار، تناظر گراف و طبقه بندی گراف، موضوع مهمی به شمار می رود.
- **داده های زیستی:** داده های زیستی به شیوه ای مشابه با داده های شیمیایی طراحی می شوند. با این وجود، گراف های اختصاصی به طور معمول اندازه بسیار بزرگتری دارند. به علاوه، گره ها معمولاً بخش های مدل های زیستی را با دقت طراحی می کنند. آمینواسید می تواند نمونه ای بارز از یک گره در برنامه کاربردی DNA باشد. هر شبکه زیستی مجزا به سادگی می تواند شامل هزاران گره باشد. اندازه پایگاه داده های کلی نیز به حد کافی برای گراف های اصلی بزرگ هست که روی دیسک ذخیره شود. ماهیت ذخیره سازی ـ روی دیسک مجموعه داده ها اغلب به موضوعات منحصر به فردی منتهی می شود که در سایر طرح ها با آن مواجه نمی شویم. به عنوان مثال، ترتیب دستیابی کران ها در گراف در این حالت بسیار جدی تر و مهم تر می شود، هر الگوریتمی که برای دسترسی تصادفی به کران ها طراحی شده باشد کارایی چندانی در این حالت نخواهد داشت.
- **داده های شبکه ای شده کامپیوتری و داده های وب:** در مورد شبکه های کامپیوتری و وب، تعداد گره ها در گراف اصلی ممکن است بسیار انبوه باشد. وقتی تعداد گره ها انبوه باشد موجب پیدایش تعداد بی شماری از کران های متمایز می شود. این پدیده به عنوان مسئله قلمرو انبوه در داده های شبکه ای نیز معرفی می شود. در این گونه موارد، تعداد کران های متمایز ممکن است آن قدر زیاد باشد که نگهداری آن ها در فضای ذخیره سازی موجود دشوار شود. از این رو، باید تکنیک هایی برای فشرده کردن وکار با بازنمایی های فشرده مجموعه داده ها پدیدار شوند. در این گونه موارد، چالش

دیگری از این واقعیت بر می خیزد که ذخیره کران های تازه وارد برای تحلیل در آینده امکان پذیر نخواهد بود. با این وجود، تکنیک های فشرده سازی به ویژه برای این حالت الزامی و حیاتی است. فشرده زنجیره ممکن است برای پردازش گراف های اصلی در آینده ذخیره شود.

- **داده های XML:** داده های XML فرمی طبیعی از داده های گراف نسبتاً کلی هستند. خطر نشان می کنیم که الگوریتم های استخراج و مدیریت برای داده های XML در مورد گراف ها نیز کاملاً کارآمد و سودمند خواهند بود، زیرا داده های XML را می توان به عنوان گراف های برچسب دار در نظر گرفت. به علاوه، ترکیب های خاصیت ـ مقدار همراه با گره ها می تواند این مسئله را چالشی تر سازد. با این وجود، تحقیق در حوزه داده های XML اغلب کاملاً مستقل از تحقیق در حوزه استخراج گراف بوده است. با این حال، در این فصل تلاش خواهیم کرد الگوریتم های استخراج داده های XML را همراه با الگوریتم های استخراج و مدیریت گراف مورد بحث قرار دهیم.

بدیهی است که طراحی یک الگوریتم استخراج ویژه به حوزه کاربردی مرتبط با آن بستگی دارد. به عنوان مثال، یک مجموعه داده قابل ذخیره سازی روی دیسک نیازمند طراحی دقیق یک الگوریتم است که در آن کران های هر گراف به طور تصادفی قابل دستیابی نباشند. همچنین، شبکه های دارای حوزه فراگیر و گسترده نیازمند فشرده سازی دقیق گراف های اصلی به منظور تسهیل فرآیند هستند. از سوی دیگر، مولکول های شیمیایی حاوی کپی های فراوان از گره ـ برچسب ها چالش منحصر به فردی در قالب هم ریختی گراف را به انواعی از برنامه های کاربردی تحمیل می کنند.

در این فصل، انواع مختلفی از برنامه های کاربردی استخراج و مدیریت گراف به همراه کاربردهای متناظر با آن را مورد بررسی و بحث قرار خواهیم داد. به این نکته اشاره می کنیم که مرز بین الگوریتم های استخراج و مدیریت داده ها اغلب اوقات روشن نیست، زیرا بسیاری از الگوریتم ها را می توان در هر دو گروه طبقه بندی کرد. موضوعات مطرح شده در این فصل را می توان اساساً به سه گروه تقسیم کرد. این گروه ها به شکل زیر به بحث گذاشته می شوند:

- **الگوریتم های مدیریت گراف:** این عنوان به الگوریتم هایی برای مدیریت و شاخص گذاری حجم زیادی از داده های گراف باز می گردد. در این بخش، الگوریتم هایی برای شاخص گذاری گراف ها و همین طور پردازش پرس و جوهای گراف ارائه خواهیم کرد. انواع دیگری از پرسش ها از قبیل پرس وجوهای مرتبط با دسترسی را نیز مورد تحقیق قرار خواهیم داد. و به تحقیق درباره الگوریتم های تناظر و تطبیق گراف ها و کاربردهای آنها خواهیم پرداخت.
- **الگوریتم های استخراج گراف:** این به الگوریتم هایی که برای استخراج الگوها، روندها، مسیرها و خوشه ها از گراف به کار می روند، اشاره دارد. در بعضی از موارد، ممکن است لازم باشد الگوریتم ها را بر مجموعه های بزرگی از گراف های موجود در دیسک اعمال کنیم. روش هایی برای خوشه بندی، طبقه بندی و استخراج الگوی پرتکرار را بررسی خواهیم کرد. همچنین، بررسی مشروحی از این الگوریتم ها در آثار این حوزه ارائه خواهیم کرد.
- **کاربردهای مدیریت و استخراج داده های گراف:** حوزه های کاربردی مختلف که در آن الگوریتم های مدیریت و استخراج داده های گراف مورد نیاز است را مورد مطالعه قرار خواهیم داد. این حوزه عبارتند از: داده های وب، شبکه های اجتماعی و کامپیوتری، داده های زیستی و شیمیایی، و مکان یابی عیب یاب های نرم افزار.

۲,۲. الگوریتم های مدیریت داده های گراف

مدیریت داده های گراف چالشی تر از داده های چند بعدی شده است. باز نمایی ساختاری گراف دارای توانایی گویایی بیشتری است، اما این گویایی در ازای هزینه ای به دست می آید. این هزینه در قالب دشواری بازنمایی، دسترسی و پردازش ظاهر می شود زیرا عملیات میانی از جمله ارزیابی شباهت، معدل گیری و محاسبه فاصله و بعد ذاتاً در داده های ساختاری به شیوه داده های چند بعدی تعیین نمی شود. علاوه بر این، پایگاه داده های ارتباطی سنتی با استفاده از قطعه خواندن ـ نوشتن به طور کارآمدی قابل دسترسی است؛ این برای داده های ساختاری که در آن کران ها در یک ترتیب اختیاری قابل دسترسی اند، طبیعی به نظر نمی رسد. با این وجود، پیشرفت های اخیر قادر به کاهش دست کم بعضی از نگرانی ها بوده اند. در این بخش، ارزیابی بسیاری از الگوریتم ها و برنامه های اخیر در زمینه مدیریت گراف ارائه خواهیم داد.

۲,۲,۱. تکنیک های شاخص گذاری و پردازش پرس و جو

مدل های پایگاه داده و زبان های پرسش موجود، شامل مدل ارتباطی و SQL، فاقد پشتیبان طبیعی برای ساختارهای داده های پیشرفته مثل نمودارهای درختی و گراف ها هستند. اخیراً، با توجه به استفاده گسترده از XML به عنوان فرمتی برای تبادل داده ها، تعدادی از مدل های داده ها و زبان های پرسش جدیدی برای ساختارهای درخت ـ مانند پیشنهاد شده اند. در این اواخر، موج جدیدی از برنامه های کاربردی در حوزه مختلفی مثل شبکه، مدیریت هستی شناسی، بیو انفورماتیک، غیره... به دنبال مدل های داده جدید، زبان ها و سیستم های نوین برای داده های دارای ساختار گراف بوده اند.

به طور کلی، این وظیفه ساده خواهد بود اگر به شکل زیر ارائه شود: برای یک الگوی پرس وجو (نمودار درختی یا گراف) نمودارهای درختی یا گراف هایی را در پایگاه داده ها پیدا کنید که دارای یا مشابه الگوی پرس و جو باشند. برای تحقق مؤثر و دقیق این وظیفه، می بایست به چند موضوع مهم بپردازیم: (i) چگونه داده ها و پرس و جو را طراحی کنیم؛ (ii) چگونه داده ها را ذخیره کنیم؛ و (iii) چگونه داده ها را برای پرس و جوی مؤثر شاخص گذاری کنیم.

پردازش پرس و جوی داده هایی به شکل نمودارهای درختی. تحقیقات زیادی در مورد پردازش پرس و جوی XML انجام شده است. در سطح بالا، دو رویکرد برای طراحی داده های XML وجود دارد. یکی از این رویکردها ذخیره مدل ارتباطی موجود پس از انطباق داده های نمودار درختی با طرح ارتباطی است. رویکرد دیگر، ایجاد پایگاه داده طبیعی XML از خط شروع است [۳۳]. برای مثال، بعضی اقدامات با ایجاد نمودار درختی جبر و آنالیز برای داده های XML آغاز به کار کرده اند [۳۴]. گراف جبری پیشنهادی با تعیین اپراتورهای جدید مانند حذف گره و نصب گره برای داده های دارای ساختار گراف، جبر ارتباطی را توسعه داد.

SQL روش دسترسی استانداری برای داده های ارتباطی است. تلاش های زیادی برای طراحی یک بدیل SQL برای داده های نمودار درختی انجام شده است. معیارها عبارتند از: نخست، توان گویایی که انعطاف پذیری بیشتری برای ارائه پرس و جو از طریق داده های نمودار درختی به کاربر می دهد؛ و دوم قابلیت اظهار که به سیستم اجازه می دهد پردازش پرس و جو را بهینه سازی کند. استفاده گسترده از XML، گروه های استاندارد را وادار کرده است مشخصات SQL را به گونه ای تعمیم دهد که تابع های پردازش XML را در برگیرد. XQuery با استفاده از ساختار Flwor مسیر Xpath را برای بیان پرسش گسترش می دهد. ساختار Flwor شبیه به ساختار SELECT – FROM – WHERE است با این تفاوت که پشتیبان اضافی برای داده های نمودار درختی ایجاد می کند و توسط کنسرسیوم شبکه جهان وب (W3C) به عنوان زبان پرسش برای اسناد XML معرفی شده است.

در داده های XML، هسته پردازش پرس و جو در تناظر کارآمد و مؤثر الگوی نمودار درختی نهفته است. اکثر تکنیک های شاخص گذاری XML در [۳۵-۳۸] ارائه شده اند تا از این فعالیت حمایت کند. به عنوان مثال Data Guide یک خلاصه کوتاه از ساختار مسیر در پایگاه داده های نمودار درختی ارائه می دهد. از سوی دیگر، Tindex مجموعه مشخصی از این نظر که تمام مسیرهای برچسب دار که از قسمت ریشه شروع می شوند را حفظ می کند. Index Fabric هر یک از مسیرهای برچسب در هر یک از عامل های XML را با مقادیر داده ها به عنوان یک زنجیره کدگذاری می کند و هر مسیر بر حسب رمزدار و مقدار داده را در یک شاخص برای زنجیره هایی مثل نمودار درختی پاتریشا قرار می دهد. APEX از الگوریتم های استخراج داده برای پیدا کردن مسیرهایی که مکرراً در پروسه پرس و جو ظاهر می شوند، استفاده می کند. در حالی که اکثر تکنیک ها بر عبارات مسیر ساده تمرکز می کنند، FBInder بر انشعاب عبارات مسیر تأکید می کند. با این همه، از آنجایی که محتوای یک نمودار درختی به گره، مسیر یا شاخه های کوچک تجزیه می شود، به هم چسباندن نتایج میانی به فرآیندی زمان بر تبدیل شده است. علامت گذاری XML توالی ـ مبنا [۳۹, ۴۰]، الگوهای نمودار درختی را به شهروند درجه اول در پردازش محتوای XML تبدیل می کند. این عملیات، اسناد XML و متن ها را به توالی ها و زنجیره هایی تبدیل می کند و پردازش محتوای نمودار درختی را از طریق تطابق و تناظر توالی ها (غیر مجاور) انجام می دهد.

پردازش محتوای داده های گراف. یکی از ویژگی های مشترک در دامنه وسیعی از برنامه های کاربردی نوظهور از قبیل شبکه اجتماعی، مدیریت هستی شناسی، شبکه / مسیرهای زیستی و غیره... این است که داده هایی که به آنها مربوط است همگی دارای ساختار گراف هستند. هرچه اندازه و پیچیدگی داده ها افزایش پیدا می کند، مدیریت آن با یک سیستم پایگاه داده ها اهمیت بیشتری پیدا می کند.

چندین روش برای مدیریت گراف ها در پایگاه داده ها وجود دارد. یک احتمال اینست که برای پشتیبانی داده های گراف یک موتور RDBMS تجاری را عرضه کنیم. احتمال دیگر، استفاده از جدول های ارتباطی هدف عمومی برای ذخیره گراف هاست. وقتی این روش ها قادر به ارائه عملکرد مورد انتظار نباشند، تحقیقات اخیر با چالش طراحی یک پایگاه داده گراف با هدف مشخص نیز مواجه خواهد شد. در حال حاضر، اوراکل (Oracle) تنها DBMS تجاری است که پشتیبانی داخلی برای داده های گراف ایجاد می کند. پایگاه داده های ده گیگابایتی جدید آن شامل مدل داده های شبکه فضایی اوراکل است، که به کاربران امکان می دهد داده های گراف را طراحی کنند و به کار بگیرند. مدل شبکه شامل اطلاعات منطقی مانند قابلیت اتصال بین گره ها و لینک ها، دستورالعمل لینک ها، برآوردهای گره ها و لینک ها و غیره است. مدل منطقی عمدتاً از طریق دو جدول قابل فهم است: جدول گره و جدول لینک، که اطلاعات اتصال گراف را ذخیره می کنند. هنوز هم، خیلی ها نگرانند که مدل ارتباطی اساساً برای پشتیبانی داده های گراف مناسب نیست، زیرا حتی بنیادی ترین عملیات مانند پیمودن گراف برای اجرا در DMBS های ارتباطی پر هزینه هستند. به ویژه وقتی که گراف های بزرگی داشته باشیم. اخیراً علاقه به شبکه معنایی، توجه فزاینده ای را به Resource (RDF) Description Framework (چارچوب تشریح منابع) معطوف کرده است. ذخیره سه بعدی (Triple Store) پایگاه داده ای با هدف ویژه برای ذخیره و بازیابی داده های RDF است. حافظه سه بعدی، بر خلاف پایگاه داده های ارتباطی، برای ذخیره و بازیابی تعداد زیادی از عبارت های کوتاه در قالب فاعل ـ گزاره، مفعول که سه بعد خوانده می شوند بهینه سازی شده است. تلاش های زیادی برای پشتیبانی دسترسی مؤثر به داده های ذخیره شده روی حافظه سه بعدی انجام گرفته است. اخیراً، گروه شبکه معنایی چالش سه بعدی را اعلام کرد، که بیش از پیش ضرورت و فوریت برای پشتیبانی استنباط و استنتاج در داده های RDF وسیع و حجیم را آشکار می کند.

تعدادی از زبان های پرس و جوی گراف از اوایل ۱۹۹۰ معرفی شده اند. به عنوان مثال، GraphLog، که ریشه در DataLog دارد، استنتاج قوانین و اصول درباره مسیرهای گرافی که از طریق عبارات متعارف و منظم بیان شده اند را اجرا می کند. GOOD که ریشه هایش در پایگاه داده های مفعول ـ محور قرار دارد، یک زبان تغییر شکل تعریف می کند که حاوی پنج عملیات بنیادی روی گراف است. GraphDB مدل دیگری از داده های مفعول ـ محور در زبان پرس و جوی دیگری برای گراف هست که پرس و جوی گراف را در چهار مرحله پیاده می کند، که هر یک از این مراحل عملیاتی را بر گراف های فرعی مشخص شده به وسیله عبارات منظم اجرا می کند. Graph QL، بر خلاف سایر زبان های پرس و جوی قبلی که روی گره ها، کران ها یا مسیرها عمل می کنند، به طور مستقیم روی گراف ها عمل می کند. به عبارت دیگر، گراف ها به عنوان نوع بازگشتی و بازده تمام فعالیت ها مورد استفاده قرار می گیرد. Graph QL عمل کننده های جبری ارتباطی از قبیل مجموعه، حاصل ضرب دکارتی، و عملیات مجموعه برای ساختار های گراف هستند. به عنوان مثال، عمل کننده مجموعه برای تناظر الگویی گراف جمع بندی شده است. Graph QL به لحاظ ارتباطی کامل است و نسخه غیر متناوب آن معادل جبر ارتباطی است. شرح مفصل Graph QL و مقایسه آن با سایر زبان های پرس و جوی گراف را می توانید در [۴۱] بیابید.

با ظهور برنامه های کاربردی شبکه معنایی، نیاز به داده های RDF کار آمد مورد توجه قرار گرفته است. زبان پرس و جوی SPARQL برای این هدف طراحی شده است. به طوری که پیش تر گفتیم، یک گراف در فرمت RDF به وسیله مجموعه ای از سه بعد توصیف می شود که هر یک از آن ها متناظر با یک کران بین دو گره است. پرس و جوی SPARQL، که SQL ـ مانند نیز هستند احتمالاً متشکل از الگوهای سه بعدی، پیوستگی ها، گسستگی ها، و الگوهای مطلوب است. الگوی سه بعدی از نظر نحوی به سه بعدی RDF نزدیک است با این تفاوت که هر یک از فاعل، گزاره یا مفعول ها ممکن است متغیر باشند. پردازشگر پرس و جوی SPARQL، مجموعه ای از سه بعدی ها را جستجو می کند که با الگوهای سه بعدی متناظر باشند، و متغیرهای موجود در پرس و جوی گراف را به بخش های متناظر هر سه بعدی متصل می کند.

روند کاری دیگری در شاخص گذاری گراف از ویژگی های ساختاری مهم گراف اصلی به منظور تسهیل شاخص گذاری و پردازش پرس و جو استفاده می کند. این ویژگی های ساختاری می تواند به شکل مسیرها یا الگوهای پرتکرار در گراف های اصلی باشند. از این ها می توان به عنوان فیلترهای پیش ـ پردازش که گراف های نامتناسب داده های اصلی را در مرحله ابتدایی حذف می کنند، استفاده کرد. به عنوان مثال، تکنیک GraphGrey [۴۲] از مسیرهای تعیین شده به عنوان ویژگی های شاخص استفاده می کند که به منظور فیلتر کردن گراف های نامتناظر می توان از آن ها استفاده کرد. همچنین، تکنیک GIndex [۴۳] از قطعه های پرتکرار قابل تشخیص به عنوان ویژگی های شاخص استفاده می کند. یک تکنیک کاملاً مرتبط [۴۴]، به منظور تسهیل شاخص گذاری به ذخیره زیر ساختارها در گراف های اصلی می پردازد. روش دیگر شاخص گذاری گراف ها، استفاده از ساختارهای نمودار درختی در گراف های اصلی به منظور تسهیل شاخص گذاری و جستجو است.

موضوع پردازش پرس و جو در داده های گراف سالیان متمادی مورد مطالعه قرار گرفته است، ولی بسیاری از چالش های مرتبط با آن به جای خود باقی است. از یک سو، داده ها به طور فراینده ای در حال افزایش هستند؛ یک احتمال برای کنترل و استفاده از چنین داده های عظیمی از طریق پردزاش موازی با استفاده از، مثلاً چارچوب Map/Reduce است. با این وجود، به خوبی می دانیم که بسیاری از الگوریتم های گراف نیز وجود دارند که متناظر کردن آنها بسیار دشوار است. از سوی دیگر، پرس و جوی گراف ها به طور فراینده ای در حال پیچیده شده است. به عنوان مثال، پرس و جوهایی که در برابر هستی شناسی پیچیده قرار دارند اغلب طولانی هستند، فارغ از اینکه کدام زبان پرس و جوی گراف برای ارئه پرس و جو مورد استفاده قرار می گیرد.

علاوه بر این، در صورت وجود یک گراف پیچیده (مانند هستی شناسی پیچیده)، کاربرها اغلب به جای درک و تعریف درست و روشنی از آنچه که درباره آن به پرس و جو پرداخته اند، فقط ایده ای مبهم وگنگ از این موضوع دارند. این ها موجب فراخوانی روش های جایگزین برای پردازش و بیان پرس و جوهای گراف می شود. به عبارت دیگر، به جای بیان شفاف پرس و جو ها در دقیق ترین چارچوب، ممکن است بخواهیم از جستجوی واژگان کلیدی برای تسهیل پرس و جوها [۴۲]، یا بهره گیری از روش های استخراج داده برای تشکیل پرس و جوی نیمه خودکار استفاده کنیم.

۲,۲,۲. پرس و جوهای دسترسی

پرس و جوهای دسترسی گراف بررسی می کند که آیا مسیری از گره v به گره u در یک گراف مستقیم بزرگ وجود دارد یا نه. پرس و جو برای دسترسی گراف یک عملیات بنیادی است که برای بسیاری از برنامه های کاربردی از جمله برنامه های حوزه شبکه معنایی (سمنتیک وب)، شبکه های زیستی، پردازش پرس و جوهای XML و غیره اهمیت زیادی دارد.

پرس و جو های دسترسی را می توان به دو روش روشن و مشخص پاسخ داد. در روش اول، با استفاده از جستجوی عرض ـ محور یا عمق ـ محور و با شروع از گره v، از گراف عبور می کنیم تا مشاهده کنیم که آیا قادر به دسترسی به گره u هستیم یا نه. زمان پرس و جو برابر با O(n+m) است، که n تعداد گره ها و m تعداد کران های موجود در گراف است. در آن سو، بن بست گذرای کران در گراف را محاسبه و ذخیره می کنیم. با بن بست گذرا، که مستلزم ذخیره $O(n^2)$ است، پرس و جوی دسترسی را می توان در زمان O (۱) و از طریق بررسی وجود (u,v) در بن بست گذرا پاسخ داد. با این وجود، در نمودارهای گسترده، هیچ یک از دو روش شدنی نیست: روش اول از نظر زمان پرس و جو بسیار پر هزینه است، و روش دوم نیازمند فضای بسیار زیادی است.

تحقیق در این حوزه بر یافتن بهترین میانگین بین زمان پرس و جو O(n+m) و هزینه ذخیره $O(n^2)$ تمرکز می کند. این تحقیق از راه شهودی تلاش می کند اطلاعات دسترسی را در بن بست کران فشرده کند و با استفاده از داده های فشرده به پرس و جو ها پاسخ دهد.

رویکردهای مبتنی بر نمودار درختی دربرگیرنده. رویکردهای بسیار، مانند [۴۵-۴۷]، گراف را به دو بخش تقسیم می کنند: i) نمودار درختی در بر گیرنده، و ii) کران هایی که روی نمودار درختی قرار ندارند (کران های غیر درختی). اگر در نمودار درختی در برگیرنده، مسیری بین u و v وجود داشته باشد، دسترسی بین u و v قطعاً به سادگی می تواند باشد. با اختصاص یک کد فاصله (u_{start} و u_{end}) به هر گره u این کار امکان پذیر است، به طوری که v از u قابل دسترس باشد اگر و تنها اگر $u_{start} \leq V_{start} \leq u_{end}$ کل نمودار درختی را می توان با اجرای یک عبور عمق ـ محور ساده از نمودار درختی کد گذاری کرد. با انجام فرآیند کد گذاری، بررسی دسترسی در زمان O (۱) قابل اجرا خواهد بود.

اگر هیچ مسیری روی نمودار درختی در برگیرنده، دو گره را به هم متصل نکند، باید بررسی کنیم که آیا مسیری که کران های غیر درختی را در بر می گیرد، دو گره را به هم متصل می کند یا نه. به این منظور، باید ساختارهای شاخص به علاوه کد فاصله را ایجاد کنیم تا بررسی دسترسی را شتاب بیشتری ببخشیم. ساختارهای شاخص برای این منظور معرفی شدند [۴۶] و رویکرد آن ها به زمان پرس و جویی معادل O (m-n) دست یافت. به عنوان مثال. SSPI (شاخص جایگزین و مازاد پیشین) فهرست پیشین PL (u) را برای هر گره u حفظ می کند، که همراه با کد فاصله موجب بررسی دسترسی کارآمد و مؤثر می شود. بسیاری از گراف های گسترده در برنامه های کاربردی واقعی نامتراکم هستند، که این بدان معناست که تعداد کران های غیر درختی اندک

است. الگوریتم پیشنهادی براساس این فرض با استفاده از ساختار شاخص اندازه O (n + t^2) به پرس و جوی دسترسی در زمان O (۱) پاسخ دادند، که t معادل تعداد کران های غیر درختی است، و $t \ll n$.

رویکردهای مبتنی بر پوشش مجموعه. چندین روش برای استفاده از ساختارهای ساده تر داده ها (مثلاً نمودارهای درختی، مسیرها و غیره) برای پوشش اطلاعات دسترسی که در قالب ساختار گراف بیان شده اند، معرفی می شود. به عنوان مثال، اگر v بتواند به u دسترسی پیدا کند، آنگاه v می تواند به هر یک از گره های نمودار درختی که از u منشأ گرفته، دسترسی پیدا کند. بنابراین، اگر نمودار درختی را در شاخص بگنجانیم آنگاه مجموعه بزرگی از دسترسی در گراف را تحت پوشش قرار داده ایم. سپس، از نمودار درختی چندگانه برای پوشش یک گراف کامل استفاده می کنیم. پوشش نمودار درختی مطلوب به زمان پرس و جوی O (logn) دست پیدا می کند، که n برابر با تعداد گره های موجود در گراف است. روش دیگری به جای استفاده از نمودار های درختی پیشنهاد شد که گراف ها به زنجیره های اتصال جفت محور تقسیم شوند، و سپس از زنجیره ها برای پوشش گراف استفاده شود. ایده استفاده از زنجیره شبیه به استفاده از نمودار درختی است: اگر v بتواند روی یک زنجیره به u دسترسی پیدا کند، آنگاه v می تواند به هر یک از گره هایی که پس از u روی زنجیره قرار می گیرند دسترسی پیدا کند. روش پوشش ـ زنجیره ای به زمان پرس و جوی O (nk) دست می یابد، که k برابر با تعداد زنجیره های موجود در گراف است. روش دیگر یک پوشش ۲ ـ مرحله ای برای پرس و جوهای دسترسی است. گره u به وسیله دو مجموعه از گره ها به نام L_{in} (u) و L_{out} (u) بر چسب زده می شوند، که L_{in} (u) معادل گره هایی است که می تواند به u دسترسی داشته باشد و L_{out} (u) معادل گره هایی است که u می تواند به آنها دسترسی پیدا کند. روش ۲ ـ مرحله ای، برچسب های L_{in} و L_{out} را به هر یک از گره ها نسبت می دهد به گونه ای که u بتواند به v دسترسی پیدا کند اگر و تنها اگر $L_{out}(u) \cap L_{in}(v) \neq \emptyset$. مسئله پوشش ۲ـ مرحله ای مطلوب برای یافتن اندازه مینیمم پوشش ۲ـ مرحله NP دشوار است. یک الگوریتم سخت گیر، پوشش ۲ - مرحله ای را متناوباً پیدا می کند. در هر تناوب، گره ω را انتخاب می کند که مقدار $\frac{S(A_\omega,\omega,D_\omega)\cap TC'}{|A_\omega|+|D_\omega|}$ را به حداکثر می رساند، در حالی که $S(A_\omega, \omega, D_\omega) \cap TC'$ بیانگر دسترسی (غیر پوششی) جدیدی است که خوشه ۲ ـ مرحله ای متمرکز در ω قابل پوشش است، و$|A_\omega| + |D_\omega|$ برابر با اندازه خوشه ۲ ـ مرحله ای متمرکز در ω است. چندین الگوریتم برای ارزیابی کارآمد پوشش های ۲ـ مرحله ای با کیفیت [۴۸, ۴۹] ارائه شده اند. تعمیم های بسیاری برای روش های مبتنی بر پوشش مجموعه پیشنهاد شده اند.

تعمیم هایی به مسئله دسترسی. پرس و جو های دسترسی یکی از اساسی ترین بخش های سازنده بسیاری از فرآیندهای عملیاتی پیشرفته گراف هستند، و بعضی از آن ها مستقیماً با پرس و جو های دسترسی ارتباط دارند. حوزه گراف های برچسب دار یکی از مسائل جالب است. در بسیاری از برنامه های کاربردی، به کران ها برچسب زده می شود تا بیانگر رابطه بین دو گره باشد که به وسیله کران به هم متصل شده اند. گونه جدیدی از پرس و جو های دسترسی این پرسش را مطرح می کند که آیا دو گره به وسیله مسیری که کران های آن با مجموعه معینی از برچسب ها محدود شده، به هم پیوسته اند [۵۰]. در بعضی از برنامه های کاربردی، می خواهیم کوتاه ترین مسیر بین دو گره را پیدا کنیم. مسئله کوتاه ترین مسیر نیز همانند مسئله دسترسی آسان می تواند از طریق روش های نیروی اجباری، مثل الگوریتم دیجکسترا، مرتفع شود اما اینگونه روش ها برای پرس و جو های آنلاین در گراف های گسترده و بزرگ مناسب به نظر نمی رسند.

۲,۲,۳. تناظر گراف

مسئله تناظر گراف، پیدا کردن تناظر تقریبی یا یک به یک بین گره های دو گراف است. این تناظر براساس یک یا چند ویژگی ساختاری زیر در گراف ها استوار است: (۱) بر چسب های روی گره ها در دو گراف می بایست مشابه باشند. (۲) حضور کران ها بین گره های متناظر در دو گراف می بایست متناظر باشند. (۳) بر چسب های روی کران ها در دو گراف باید متناظر باشند.

این سه ویژگی ممکن است برای تعیین تناظر بین دو گراف مورد استفاده قرار بگیرند به طوری که تناظر یک به یک بین ساختارهای دو گراف وجود داشته باشد. اینگونه مسائل، اغلب در چارچوب تعداد برنامه های مختلف پایگاه داده ها از قبیل تناظر چارچوب کلی، تناظر پرس و جو و جاسازی فضای بردار ظاهر شود. در تناظر گراف دقیق تلاش می کنیم تناظر یک به یک بین دو گراف را مشخص کنیم. بنابراین، اگر یک کران بین یک جفت گره در یک گراف وجود داشته باشد، آنگاه آن کران باید بین جفت متناظر در گراف دیگر نیز وجود داشته باشد. این پدیده در برنامه های کاربردی واقعی که در آنها تناظر تقریبی وجود دارد ولی تناظر دقیق شدنی به نظر نمی رسد، چندان عملی نخواهد بود. از این رو، در بسیاری از برنامه های کاربردی، تعریف یک تابع واقعی که تشابه در تناظر بین دو گراف را تعیین کند، امکان پذیر است. تناظر خطای مجاز، یک برنامه کاربردی مهم در حوزه گراف است، زیرا باز نمایی عمومی گراف ها ممکن است دارای گره ها و کران های گمشده بسیاری باشد. این مسئله به عنوان تناظر گراف غیر دقیق نیز شناخته می شود. اکثر متغیرهای مسئله تناظر گراف به عنوان هارد تفکیک نشده[۲] در نظر گرفته می شوند. متداول ترین روش برای تناظر گراف مربوط به تکنیک های جستجوی مبتنی بر نمودار درختی است. در این تکنیک، از مجموعه مناسب کشت گره های متناظر شروع می کنیم و به طور متناوب مجاور تعیین شده توسط همان مجموعه را تعمیم می دهیم. تعمیم متناوب، از طریق افزودن گره ها به مجموعه گره کنونی قابل اجرا است تا زمانی که هیچ یک از محدودیت های کران نقض نشود. اگر مشخص شود که مجموعه گره کنونی قابل تعمیم نیست، آنگاه یک پروسه عقب نشینی را آغاز می کنیم که آخرین مجموعه تناظر را لغو کنیم. تعدادی از الگوریتم هایی که بر مبنای این ایده فراگیر طراحی شده اند در [۵۱] مورد بررسی قرار گرفته اند.

مسئله تناظر دقیق گراف ارتباط نزدیکی با مسئله هم ریختی گراف دارد. در مورد مسئله هم ریختی گراف، تلاش می کنیم تناظر یک به یک دقیقی بین گره ها و کران های دو گراف پیدا کنیم. جمع بندی این مسئله مربوط به یافتن زیر گراف مشترک بیشینه است که در آن سعی می کنیم تعداد بیشینه گره ها بین دو گراف را با هم تطبیق دهیم. توجه داشته باشید که راه حل مسئله زیرگراف (گراف فرعی) مشترک بیشینه می تواند راه حلی نیز برای مسئله تناظر دقیق بین دو زیر گراف ارائه کند، البته در صورتی که اصولاً چنین راه حلی وجود داشته باشد. تعدادی از معیارهای تشابه را می توان بر اساس رفتار تناظر بین دو گراف استخراج کرد. اگر دو گراف مشترکاً در تعداد زیادی از گره ها سهیم باشند، آنگاه تشابه چشمگیرتر است. ایده اصلی و فراگیر در بسیاری از این روش ها تعیین یک معیار فاصله براساس ماهیت تناظر بین دو گراف و استفاده از این معیار فاصله به منظور هدایت الگوریتم ها به سمت راه حلی مؤثر است.

تناظر غیر دقیق گراف عملی تر و اجرایی تر است زیرا خطاهای طبیعی که ممکن است در طی فرآیند تناظر اتفاق بیفتد را توجیه می کند. بدیهی است که روشی برای تعیین این خطاها و نزدیکی بین گراف های مختلف ضروری است. تکنیک مشترکی

---

[۲] NP-Hard

که برای تعیین این خطا مورد استفاده می گیرد، استفاده از تابعی مثل فاصله ویرایش گراف است. این تابع فاصله بین دو گراف را با اندازه گیری میزان ویرایش های لازم برای تبدیل یک گراف به گرافی دیگر تعیین می کند. این ویرایش ها (اصلاحات) ممکن است در قالب افزودن، حذف یا جایگزینی گره ها یا کران ها صورت بگیرد. تناظر غیر دقیق گراف، امکان تناظر بین دوگراف پس از یک سلسله از این ویرایش ها را به وجود می آورد. کیفیت و حالت تناظر با توجه به اندازه ویرایش های متناظر تعیین می شود. لازم به ذکر است که مفهوم فاصله ویرایش گراف کاملاً وابسته به یافتن زیر گراف مشترک بیشینه است؛ زیرا این امکان وجود دارد که یک الگوریتم مبتنی بر ویرایش، فاصله را به سمت پیدا کردن زیر گراف مشترک بیشینه از طریق تعیین فاصله ویرایش مناسب هدایت کنیم.

گونه خاص مسئله زمانی است که مقادیر بر چسب های روی گره ها و کران ها را در طی فرآیند تناظر پیدا می کنیم. در این حالت، لازم است فاصله بین برچسب های گره ها و کران ها را به منظور تعیین میزان جایگزینی برچسب محاسبه کنیم. بدیهی است که میزان جایگزینی برچسب وابسته به برنامه کاربردی مورد نظر است. در صورت وجود برچسب های عددی، ممکن است تعیین فاصله بر مبنای توابع فاصله ای عددی بین دو گراف طبیعی به نظر برسد. به طور کلی، میزان ویرایش ها نیز به برنامه کاربردی بستگی دارد، چرا که برنامه های مختلف ممکن است از مفاهیم مختلف شباهت استفاده کنند. بنابراین، از تکنیک های مشخص شده بر اساس قلمرویی خاص به منظور تعیین میزان ویرایش ها استفاده می شود. در بعضی از موارد، میزان ویرایش حتی ممکن است با استفاده از گراف های نمونه تعیین شود [۵۲, ۵۳]. وقتی با مواردی مواجه می شویم که در آنها گراف های نمونه به طور طبیعی فاصله بین آنها را مشخص می کنند، اندازه و میزان ویرایش به عنوان مقادیری تعیین می شوند که فاصله های تناظر تا جایی که امکان دارد به مقادیر نمونه نزدیکند.

الگوریتم های متعارف برای تناظر غیر دقیق گراف از جستجوی ترکیبی و تلفیقی در فضای ویرایش های ممکن استفاده می کنند به این منظور که تناظر مطلوب را مشخص کند. ایده کلی ادغام ایده های کلیدی و مهم فاصله ویرایش گراف در توابع اصلی است. از آنجایی که توابع اصلی (کرنل) به عنوان تکنیک هایی به شدت قدرتمند برای شناسایی الگو هستند، این طور نتیجه گیری می شود که این تکنیک ها را می توان به مسئله تناظر گراف الحاق کرد. روش های اصلی کلیدی عبارتند از: توابع اصلی دشواری و پیچیدگی، توابع اصلی عبور تصادفی و توابع اصلی انتشار در توابع اصلی عبور تصادفی، تلاش می کنیم تعداد عبورهای تصادفی بین دو گراف که دارای چند برچسب مشترک هستند را تعیین کنیم. توابع اصلی انتشار را می توان به عنوان جمع بندی تابع Gaussim در فضای اقلیدسی در نظر گرفت.

تکنیک بر چسب گذاری تحقیقی (کاهشی) یک گروه بسیار گسترده از روش هایی است که اغلب برای تناظر نمودار مورد استفاده قرار می گیرد. توجه داشته باشید که در مورد مسئله تناظر، ما واقعاً تلاش می کنیم برچسب ها را به گره های هر گراف نسبت دهیم. بر چسب ویژه برای یک گره از مجموعه ناپیوسته احتمالات استخراج می شود. این مجموعه ناپیوسته از احتمالات با گره های تطبیق دهنده در سایر گراف ها متناظر است. احتمال تناظر با استفاده از پراکندگی های احتمال Gaussian تعیین می شود. با برچسب گذاری اولیه بر مبنای ویژگی های ساختاری گراف اصلی شروع می کنیم و پس از آن به طور پی در پی پاسخ را بر اساس شناسایی تکمیلی اطلاعات ساختاری اصلاح می کنیم.

۲,۲,۴. جستجوی واژگان کلیدی

در مسئله جستجوی واژگان کلیدی، ممکن است بخواهیم گروه های کوچکی از گره های دارای لینک های پیوسته را تعیین کنیم که به واژگان کلیدی خاص وابسته اند. به عنوان مثال، یک گراف وب یا شبکه اجتماعی ممکن است به عنوان یک گراف گسترده در نظر گرفته شود که در آن هر گره ممکن است حاوی مقادیر زیادی از داده های متنی باشد. به رغم اینکه جستجوی واژگان کلیدی با توجه به متن درون گره ها تعیین می شود، لازم به ذکر است که ساختار اتصال نیز نقش مهمی در تعیین مجموعه مناسبی از گره ها بازی می کند. به خوبی می دانیم که متن در واحدهای متصل، مثل وب، به هم مرتبط هستند در هنگامی که موضوعات متناظر به هم متصل اند. بنابراین، با پیدا کردن گروه هایی از گره های کاملاً پیوسته که دارای واژگان کلیدی مشترکی هستند، به طور کلی تعیین گره هایی که به لحاظ کیفی مؤثر و کارآمد باشند امکان پذیر است. جستجوی واژگان کلیدی یک سطح مشترک ساده اما کاربر پسند برای بازیابی اطلاعات روی وب ارائه می دهد. همچنین، اثبات شده است که روشی کار آمد برای دسترسی به داده های ساختاری است. از آنجایی که بسیاری از مجموعه داده های واقعی به شکل جدول ها، نمودارهای درختی و گراف ها سازماندهی شده اند، جستجوی واژگان کلیدی در چنین داده هایی دارای اهمیت فزاینده ای است و توجه زیادی به تحقیق در زمینه پایگاه داده ها و جوامع IR معطوف کرده است.

گراف یک ساختار عمومی است و می توان آن را برای طراحی انواعی از داده های پیچیده و دشوار مانند داده های ارتباطی و داده های XML مورد استفاده قرار داد. از آنجایی که داده های اصلی به عنوان یک ساختار گراف تلقی می شوند، جستجوی واژگان کلیدی دشوارتر و پیچیده تر از جستجوی سنتی واژگان کلیدی در اسناد و مدارک است. چالش مذکور از سه منظر زیر قابل بررسی است:

- **معنای پرس و جو:** جستجوی واژگان کلیدی در یک مجموعه از اسناد متنی معنای روشنی دارد: یک سند در صورتی پاسخگوی پرس و جوی واژگان کلیدی است که تمام واژگان کلیدی موجود در پرس و جو را در بر بگیرد. در موردی که ما به آن پرداخته ایم، کل مجموعه داده ها اغلب به عنوان یک گراف مجزا در نظر گرفته می شود بنابراین الگوریتم ها می بایست در یک زمینه بهتر کار کنند و زیر گراف ها را به عنوان پاسخ باز گردانند. باید تصمیم بگیریم که کدام زیر گراف ها حائز شرایطی هستند که به عنوان پاسخ در نظر گرفته شوند.
- **استراتژی رتبه بندی:** در مورد یک پرس و جوی واژگان کلیدی مشخص، این احتمال وجود دارد که بسیاری از گراف فرعی بر مبنای معنای پرس و جوی مورد استفاده خود پاسخگوی پرس و جو باشند. با این وجود، هر گراف فرعی دارای ساختار گراف اصلی مختص به خود است با معنای نامشهودی که آن را از سایر گراف های فرعی متمایز می کند و پرس و جو را پاسخ می گوید از این رو، باید ساختار گراف را در نظر بگیریم و استراتژی های رتبه بندی طراحی کنیم که معنادارترین و به جاترین پاسخ را پیدا کند.
- **کارایی پرس و جو:** اکثر گراف های واقعی به شدت گسترده و بزرگند. کارایی پرس و جو، چالش عمده برای جستجوی واژگان کلیدی در داده های گراف است که تا حد زیادی منوط به معنای پرس و جو و استراتژی رتبه بندی است.

رویکردهای کنونی برای جستجوی واژگان کلیدی بر اساس ساختار اصلی داده ها به سه گروه تقسیم می شوند. در هر گروه معنای پرس و جو، استراتژی های رتبه بندی و الگوریتم های باز نما (نمونه) را به طور خلاصه مورد بررسی قرار می دهیم.

جستجوی واژگان کلیدی در داده های XML. داده های XML عمدتاً دارای ساختار گراف هستند، که هر گروه فقط یک مسیر جدید مجزا دارد. این ویژگی تأثیر شگرفی بر معنای پرس و جو و رتبه بندی پاسخ دارد، و فرصت بهینه سازی فوق العاده ای در طراحی الگوریتم ارائه می دهد.

با تعیین یک پرس و جو که شامل مجموعه ای از واژگان کلیدی است، الگوریتم جستجو قطعه هایی از اسناد XML را می فرستد که مرتبط ترین قطعه ها به واژگان کلیدی هستند. تفسیر "مرتبط و متناسب" متغیر است اما متداول ترین روال، پیدا کردن کوچکترین نمودارهای درختی فرعی است که شامل واژگان کلیدی مورد نظر باشند.

پیدا کردن نمودارهای درختی فرعی که شامل تمام واژگان کلیدی باشند آسان است. فرض کنید L_i مجموعه از گره ها در سند XML باشد که شامل واژگان کلیدی K_i است. اگر یکی از گره های n_i را از هر L_i برداریم و نمودار درختی فرعی از این گره ها تشکیل دهیم، آنگاه نمودار درختی فرعی شامل تمام واژگان کلیدی خواهد بود. بنابراین، پاسخ به پرس و جو را می توان از طریق Lca (n_i, ...، n_n) غیر متداول ترین شکل های قبلی گره ها n_i, ... , n_n در نمودار درختی بیان کرد، که در آن $n_i \epsilon L_i$.

اکثر معنی های پرس و جو فقط به کوچکترین پاسخ ها توجه دارند. شیوه های گوناگونی برای تفسیر مفهوم "کوچکترین" وجود دارد. چندین الگوریتم بر مبنای معنای SLCA (کوچکترین شکل های قبلی غیر متداول) طراحی شده اند که نیازمند آن است که یک پاسخ (دست کم شکل قبلی متداول گره هایی که شامل تمام واژگان کلیدی هستند) دارای هیچ زاده ای نباشد که خود به صورت پاسخ ظاهر شود. XRank معنای پرس و جوی متفاوتی برای جستجوی واژگان کلیدی انتخاب می کند. در XRank، پاسخ متشکل از نمودارهایی درختی فرعی است که شامل حداقل یک پیشامد از تمام واژگان کلیدی پرس و جو باشد، پس از خارج کردن زیر گره هایی که از قبل شامل تمام واژگان کلیدی پرس و جو بوده اند. بنابراین، مجموعه پاسخ ها بر اساس معنای SLCA، زیر مجموعه ای از پاسخ های واجد شرایط برای XRank است.

پرس و جوی واژه کلیدی ممکن است تعداد زیادی پاسخ پیدا کند، اما با توجه به تفاوت در شیوه جاسازی آن ها در ساختار XML همه پاسخ ها مساوی و هم ارز نیستند. بسیاری از رویکردها برای جستجوی واژه کلیدی در داده های XML، مثل XRank و XSEarch، روش رتبه بندی ارائه می دهند. مکانیسم های رتبه بندی چندین فاکتور را در نظر می گیرد. به عنوان مثال، پاسخ های صریح تر در مقایسه با پاسخ هایی که صراحت و دقت کمتری دارند باید در رتبه بالاتری قرار بگیرند. SLCA و معنی هایی که توسط XRank برگزیده شده اند بر این تفکر دلالت دارند. علاوه بر این، واژگان کلیدی در یک پاسخ باید کاملاً نزدیک به یکدیگر ظاهر شوند و نزدیکی به صورت فاصله معنایی تعیین شده در ساختار جاسازی شده XML تفسیر می شود.

جستجوی کلیدی در داده های ارتباطی. SQL یک زبان پرس و جوی بالفعل برای دسترسی به داده های ارتباطی است. با این وجود، برای استفاده از SQL می بایست اطلاعاتی درباره چارچوب کلی داده های ارتباطی داشته باشیم. این به مانعی بر سر راه کاربران بالقوه برای دسترسی به مقادیر قابل توجهی از داده های ارتباطی تبدیل شده است.

جستجوی واژگان کلیدی با توجه به سهولت استفاده می تواند جایگزین خوبی باشد. چالش های استفاده از جستجوی واژگان کلیدی در داده های ارتباطی ناشی از این واقعیت است که در پایگاه داده های ارتباطی، اطلاعات درباره یک واحد مجزا معمولاً میان چندین برچسب مختلف تقسیم می شود. این امر ناشی از اصل نرمال سازی است که روش طراحی چارچوب کلی پایگاه داده های ارتباطی است.

بنابر این، برای پیدا کردن واحدهایی که با پرس و جوی واژه کلیدی مرتبط باشند. الگوریتم جستجو باید داده ها را از جدول های چندگانه به هم متصل کند. اگر هر جدول را به شکل یک گره بیان کنیم، و هر ارتباط کلید خارجی به صورت یک کران بین دو گره تعریف شود، آنگاه گرافی به دست می آوریم که به ما اجازه می دهد مسئله فعلی را به مسئله جستجوی واژگان کلیدی در گراف ها تبدیل کنیم. با این وجود، احتمال خود اتصالی نیز وجود دارد: یعنی، ممکن است یک جدول شامل یک کلید خارجی باشد که خودش را به عنوان مرجع تلقی کند. به نحو گسترده تری، ممکن است حلقه هایی در گراف وجود داشته باشد که به این معناست که فقط اندازه داده می تواند اندازه اتصال را محدود کند. برای پرهیز از این مسئله، الگوریتم جستجو ممکن است از یک محدوده بالایی برای محدود کردن تعداد اتصال ها استفاده کند [۵۴].

شناخته شده ترین الگوریتم های جستجوی واژگان کلیدی برای داده های ارتباطی عبارتند از DBX-Plover و DISCOVER. این دو الگوریتم از پایگاه داده های فیزیکی جدیدی در پایگاه داده استفاده می کنند.

جستجوی واژگان کلیدی در داده های گراف. جستجوی واژگان کلیدی در گراف های بزرگ فاقد چارچوب با این چالش مواجه است که چگونه در ساختار گراف گردش کند و گراف های فرعی را پیدا کند که حاوی تمام واژگان کلیدی موجود در پرس و جو باشند. برای ارزیابی "کیفیت خوب" یک پاسخ، اکثر رویکردها به همه کران ها و گره ها امتیاز می دهند، و سپس امتیازات را به عنوان مقیاس کیفیت خوب در گراف فرعی جمع بندی می کنند. به طور معمول، هر کران با توجه به قدرت پیوستگی و هر گره با توجه به اهمیت آن بر اساس مکانیسم Page Rank امتیاز می گیرد.

الگوریتم های جستجوی واژگان کلیدی گراف را می توان به دو گروه طبقه بندی کرد. الگوریتم های گروه اول، گراف های فرعی تناظر را با بررسی لینک به لینک گراف پیدا می کند، بدون اینکه از هیچ یک از شاخص های گراف استفاده کند. الگوریتم های نمونه در این گروه عبارتند از: BANKS [۵۵] و الگوریتم جستجوی دو سویه [۵۶]. یکی از نقطه ضعف های این رویکردها این است که آنها الگوریتم ها را کورکورانه بررسی می کنند زیرا فاقد یک تصویر کلی از ساختار گراف هستند و از پراکندگی واژگان کلیدی در گراف اطلاعی ندارند. الگوریتم های گروه دوم شاخص ـ محور [۵۷] هستند، و از شاخص برای جهت دادن به بررسی گراف و پشتیبانی از جهش های رو به جلو در جستجو استفاده می کنند.

۲,۲,۵. خلاصه سازی گراف های بزرگ

یک چالش کلیدی که در بسیاری از برنامه های کاربردی که در ادامه مورد بررسی قرار می گیرند، پدید می آید این است که گراف هایی که با آنها سرو کار داریم بسیار بزرگند. در نتیجه، دسترسی به این گراف ها فقط روی دیسک ممکن خواهد بود. اکثر برنامه های استخراج گراف سنتی فرض را بر این می گذارند که داده ها روی حافظه اصلی قرار دارند. با این وجود، وقتی گراف روی دیسک ذخیره شده باشد، برنامه هایی که دسترسی تصادفی به کران ها دارند احتمالاً به شدت گران خواهند بود. به عنوان مثال، مسئله پیدا کردن قطعه ـ مینیمم در الگوریتم های روی حافظه اصلی بسیار کار آمد است، اما در صورتی که گراف های اصلی روی دیسک ذخیره شده باشند به طور غیر منتظره ای پرهزینه خواهد بود. در نتیجه باید الگوریتم ها را به دقت طراحی کرد تا هزینه های دسترسی به دیسک کاهش یابد. یک تکنیک معمول که اغلب مورد استفاده قرار می گیرد طراحی یک تکنیک خلاصه سازی است، که گراف را در یک فضای بسیار کوچکتر فشرده می کند، اما اطلاعات کافی را به منظور پاسخ مؤثر به پرس و جو ها حفظ می کند.

خلاصه سازی به طور معمول از طریق تضادهای گره یا کران تعیین می شود. نکته کلیدی، تعیین خلاصه ای است که تمام ویژگی های ساختاری مرتبط و متناسب گراف اصلی را حفظ کند. گراف فشرده حاصل تمام ویژگی های ساختاری مهم از جمله قابلیت اتصال گراف را حفظ می کند.

یک مسئله کاملاً مرتبط در این زمینه، استخراج زنجیرهای گراف است. در این حالت، کران های گراف به طور مداوم در طول زمان به دست می آیند. این گونه موارد به طور مکرر در برنامه هایی از قبیل شبکه های اجتماعی، شبکه های ارتباطی، و تحلیل لوگ شبکه به چشم می خورد. استخراج زنجیره های گراف به شدت دشوار و چالشی است، زیرا ساختار گراف باید بلادرنگ استخراج شود. از این رو، یک رویکرد معمول ایجاد یک خلاصه از زنجیره گراف و ذخیره آن به منظور تحلیل ساختاری است. بنابراین، از این خلاصه سازی می توان برای برنامه های کاربردی فاصله ـ محور مثل مسئله کوتاه ترین مسیر استفاده کرد. برنامه کاربردی دوم که در چارچوب زنجیره های گراف مورد مطالعه قرار گرفته است مربوط به تناظر گراف است. لازم به ذکر است که این نسخه متفاوتی از مسئله ای است که در بخش های پیشین مورد بحث قرار گرفت. در این شرایط، تلاش می کنیم مجموعه ای از کران ها در یک گراف مجزا را پیدا کنیم به طوری که هیچ دو کرانی دارای نقطه پایانی مشترک نباشند. مطلوب است مقدار پیشینه یا تناظر عددی بیشینه را پیدا کنیم. ایده اصلی حفظ همیشگی یک تناظر داوطلب و به روز کردن آن همزمان با ورود کران های جدید است. وقتی یک کران جدید وارد می شود. فرآیند اضافه کردن آن ممکن است به اندازه کران در نقاط پایانی آن جابه جایی به وجود آورد. اجازه می دهیم یک کران تازه وارد در نقاط پایانی خود کران ها را جابه جا کند، در صورتی که مقدار کران تازه وارد یک فاکتور $(\gamma+1)$ از کران های رفتنی باشد.

اخیراً، بعضی از تکنیک ها نیز برای ایجاد خلاصه هایی که بتوان برای ارزیابی ویژگی های ساختاری تلفیقی گراف های اصلی از آن ها استفاده کرد، طراحی شده اند. تکنیکی برای ارزیابی آمارهای رتبه های موجود در زنجیره گراف اصلی ارائه شده است که از انواعی از تکنیک ها مثل ترسیم نمای کلی، نمونه برداری، شماره گذاری و محاسبه متمایز استفاده می کنند. روش هایی برای تعیین زمان های رتبه ها، تعیین رتبه های پر برخورد، و تعیین دامنه مجموع رتبه ها پیشنهاد شده است. به علاوه، تکنیک هایی معرفی شده اند برای این که خلاصه های مکانی کارآمدی را در زنجیره های داده ها اجرا کنند. این خلاصه برای محاسبه مثلث ها در زنجیره داده ها مورد استفاده قرار گرفته است. یک برنامه کاربردی مفید در زنجیره گراف مربوطه به مسئله PageRank است. در این مسئله، تلاش می کنیم صفحه هایی مهم در مجموعه را با استفاده از ساختار اتصال اسناد اصلی تعیین کنیم. بدیهی است که اسنادی که به وسیله تعداد زیادی از اسناد به هم متصل شده اند اهمیت بیشتری دارند. در واقع، مفهوم رتبه صفحه را می توان به صورت احتمال مشاهده یک گره از طریق موج سوار تصادفی در شبکه جهانی وب طراحی کرد. الگوریتم های طراحی شده برای گراف های استاتیک (ثابت) به کار می روند. وقتی گراف ها دینامیک باشند، مانند شبکه های اجتماعی، مسئله چالشی تر نیز خواهد شد. یک تکنیک خلاصه سازی طبیعی که از آن می توان برای چنین مواردی استفاده کرد روش نمونه گیری است. در [۵۸]، چگونگی استفاده از تکنیک نمونه برداری به منظور ارزیابی رتبه صفحه برای زنجیره های گراف نشان داده شده است. ایده اصلی این است که گره ها را در گراف به طور جداگانه نمونه برداری کنیم و فرآیند گردش تصادفی را با حرکت از این گره ها اجرا کنیم. این گردش های تصادفی را می توان به منظور برآورد احتمال حضور یک موج سوار تصادفی در یک گره مشخص مورد استفاده قرار داد. این پروسه لزوماً هم ارز با رتبه بندی صفحه است.

۲,۳. الگوریتم های استخراج گراف

اکثر برنامه های استخراج سنتی برای گراف ها نیز مورد استفاده قرار می گیرند. برنامه های کاربردی استخراج نیز مانند برنامه های کاربردی مدیریت به واسطه محدودیت های سنتی که از ماهیت ساختاری گراف اصلی نشأت می گیرد، برای اجرا شدن با چالش های فراوانی مواجهند، علیرغم این چالش ها، برخی تکنیک ها برای مسائل استخراج سنتی از قبیل استخراج الگوی پرتکرار، خوشه بندی و طبقه بندی الگوی پرتکرار طراحی شده اند. در این بخش، یک نمای کلی از بسیاری از الگوریتم های ساختاری استخراج گراف ارائه می دهیم.

۲,۳,۱. استخراج الگو در گراف ها

مسئله استخراج الگوی پرتکرار در چارچوب استخراج داده های اجرایی به طور گسترده ای مورد مطالعه قرار گرفته است. اخیراً تکنیک هایی برای استخراج الگوی پرتکرار برای داده های گراف نیز تعمیم داده شده اند. تفاوت عمده در مورد گراف ها این است که فرآیند تعیین پشتیبان کاملاً متفاوت است. مسئله برحسب حوزه کاربردی آن به شیوه های مختلفی قابل تعریف است:

- در مورد اول، گروهی از گراف ها را در اختیار داریم و مایلیم تمام الگوهایی که بخشی از گراف های متناظر را تأیید می کنند تعیین کنیم [۱۳, ۵۹].
- در مورد دوم، یک گراف بزرگ مجزا داریم و مایلیم تمام الگوهایی که حداقل در چند نوبت معین در این گراف بزرگ مورد تأیید قرار می گیرند را تعیین کنیم [۶۰, ۶۱].

در هر دو مورد، لازم است در تعیین تأیید یک گراف توسط دیگری، موضوع هم ریختی را نیز در نظر بگیریم. با این وجود، مسئله تعیین پشتیبان چالشی تر است اگر همپوشی بین تثبیت کننده های مختلف امکان پذیر باشد. این موضوع به این خاطر است که اگر چنین همپوشی هایی را مجاز کنیم آنگاه ویژگی یکنواختی اکثر الگوریتم های استخراج الگوی پرتکرار نقض می شود.

برای مورد اول، جایی که مجموعه ای از داده ها شامل گراف های چند گانه داریم می توان اکثر داده ها را به سادگی تعمیم داد. به عنوان مثال، الگوریتم هایی به سبک آپریوری را می توان با استفاده از استراتژی طبقه ـ محور مشابهی برای ایجاد داوطلب های (K+۱) از الگوهای K به سادگی تعمیم داد. تفاوت عمده در این است که باید فرآیند اتصال را به شیوه ای متفاوت تعریف کنیم. اندازه این ساختار بر حسب گره ها یا کران قابل تعریف است. در مورد الگوریتم AGM [۱۳]، این ساختار عمومی بر حسب تعداد رأس های مشترک قابل تعریف خواهد بود. از این رو دو گراف با K رأس به هم متصل می شوند تنها اگر دارای یک گراف فرعی مشترک با دست کم (K_۱) رأس باشند.

راه دوم برای اجرای فرآیند استخراج، اتصال دو گراف است که دارای یک گراف فرعی هستند که دست کم شامل (K_۱) کران مشترک است. از الگوریتم FSG [۵۹] می توان به منظور اجرای اتصال کران ـ محور بهره گرفت. همچنین، تعیین اتصالات برحسب ساختارهای اختیاری نیز امکان پذیر است. به عنوان مثال، بیان گراف ها برحسب مسیرهای تجزیه ـ کران امکان پذیر است. در این گونه موارد، گراف های فرعی با مسیرهای تجزیه کران (K+۱) را می توان از دو گراف که دارای K مسیر تجزیه کران هستند ایجاد کرد که از میان آنها (K_۱) مسیر باید مشترک باشند. استراتژی دیگری از اغلب مورد استفاده قرار می گیرد مربوط به تکنیک های افزایش الگو است که در آن الگوهای گراف پرتکرار با استفاده از کران های اضافی تعمیم داده می شوند [۱۷, ۶۲]. مانند مسئله

استخراج الگوی پرتکرار، از ترتیب واژگان نمایی میان کران ها استفاده می شود تا فرآیند جستجو سازماندهی شود به طوری که یک الگوی مشخص تنها یک بار رؤیت شود.

در مورد دوم که یک گراف بزرگ مجزا داریم، ممکن است از چند تکنیک مختلف برای تعیین پشتیبان در حضور همپوشی ها استفاده شود. یک استراتژی عمومی استفاده از اندازه مجموعه مستقل بیشینه ای از گراف های همپوشان برای تعیین پشتیبان است. به این استراتژی پشتیبان مجموعه مستقل بیشینه نیز گفته می شود. در [۶۳]، دو الگوریتم HSIGRAM و VSIGRAM برای تعیین گراف های فرعی پرتکرار درون یک گراف بزرگ مجزا معرفی شدند. در مورد پیشین، از یک رویکرد جستجوی وسعت ـ محور به منظور تعیین گراف های فرعی پرتکرار استفاده می شود، در حالی که از یک رویکرد عمق ـ محور برای مورد دوم استفاده می شود. در [۶۱]، نشان داده شده است که مقیاس مجموعه مستقل بیشینه همچنان به تأمین ویژگی ضد یکنواختی ادامه می دهد. مسئله اصلی در مورد این مقیاس این است که محاسبه آن به شدت پرهزینه است. بنابراین، تکنیک معرفی شده در [۶۰] یک مقیاس متفاوت برای محاسبه پشتیبان الگو تعریف می کند. محاسبه پشتیبان تصویر ـ محور بیشینه یک الگوی مشخص مد نظر است. در این حالت، تعداد گره های منحصر بفرد گراف که یک گره از الگوی مشخص با آن متناظر می شود را محاسبه می کنیم. این مقیاس همچنان به تأمین ویژگی ضد یکنواختی ادامه می دهد و به همین دلیل می توان از آن به منظور تعیین الگوهای پرتکرار اصلی استفاده کرد.

درمورد استخراج الگوی پرتکرار استاندارد، تعدادی از متغیرها برای یافتن الگوهای گراف محتمل است که عبارتند از تعیین الگوهای بیشینه [۶۲]، الگوهای مسدود [۳۰]، یا الگوهای مهم [۶۴]. لازم به ذکر است که الگوهای گراف مهم را می توان بسته به کاربرد آن به شیوه های مختلفی تعیین کرد. در [۶۴]، گراف های مهم با تبدیل نواحی گراف ها به خاصیت ها و ارزیابی ارزش متناظر بر حسب مقادیر ـ p تعیین می شوند. در [۳۰]، الگوهای مهم برحسب توابع واقعی اختیاری تعیین می شوند. یک فرا چارچوب در [۳۰] معرفی شده است تا الگوهای مهم بر مبنای توابع واقعی اختیاری تعیین شوند. یکی از رویکردهای جالب برای شناسایی الگوهای مهم، ساخت نمودار درختی جستجوی مدل ـ محور یا mbT [۶۵] است. استفاده از تقسیم و دستیابی برای شناسایی الگوهای مهم، ساخت نمودار درختی جستجوی مدل ـ محور یا mbT است. استفاده از تقسیم و دستیابی برای استخراج مهم ترین الگوها در یک فضای فرعی از نمونه ها، ایده اصلی این رویکرد است. این رویکرد یک درخت تصمیم می سازد که داده ها را به گره های مختلف تقسیم بندی می کند. پس از آن در هر گره، مستقیماً یک الگوی متمایز برای تقسیم تکمیلی نمونه ها به زیر مجموعه های واضح تر و محض تر شناسایی می کند. از آنجایی که تعداد نمونه های نزدیک به سطح برگ نسبتاً کم است، این رویکرد قادر است الگوهای دارای پشتیبان کلی کاملاً ضعیف که روی مجموعه داده های کلی قابل شمارش نیستند را بررسی می کند. برای بعضی از مجموعه داده های گراف که در برنامه های کشف مواد مخدر وجود دارند، می تواند الگوهای گراف مهم را استخراج کنند که برای بسیاری از رویکرد های دیگر دشوار است. این الگوریتم به خاطر استفاده از الگوی تکنیک mbT به گراف ها محدود نمی شود بلکه برای مجموعه داده ها و توالی ها نیز قابل اجرا است و مجموعه الگوی استخراج شده کوچک و مهم است.

یکی از چالش های مهم که در چارچوب تمام الگوریتم های استخراج الگوی پرتکرار پدید می آید مربوط به تعداد زیاد الگوهایی است که می توان از پایگاه داده ها استخراج کرد. این مسئله به ویژه در مورد گراف ها شدیدتر است زیرا اندازه خروجی می تواند به شدت بزرگ باشد. یک راه حل برای کاهش تعداد الگوهای بازنما (نمونه) اعلام الگوهای پرتکرار بر حسب راست گوشه بودن (قائم بودن) آنهاست. مدلی تحت عنوان ORIGAMI ارائه شده است، که الگوهای گراف پرتکرار را تنها در صورتی اعلام می کند که شباهت کمتر از آستانه $\propto$ باشد. اینگونه الگوها به عنوان الگوهای قائم ـ $\propto$ نیز شناخته می شوند. مجموعه الگوی ρ در صورتی

بازنمای ـ β گفته می شود که به ازای هر الگوی اعلام نشده g دست کم یک الگو در ρ پیدا شود که شباهت آن به g حداقل معادل آستانه β باشد. این دو محدودیت به بسیاری از جنبه های الگوهای ساختاری مورد توجه قرار می دهند. روش معرفی شده مجموعه ای از تمام الگوهای قائم ـ $\propto$ و باز نمای ـ β را مشخص می کند. در اینجا، کاهش موارد زائد در مجموعه الگوهای اصلی مد نظر است به این منظور که درک بهتری از الگوهای اعلام شده به دست آید.

بعضی از شکل های به ویژه چالشی مسئله در چارچوب مجموعه داده های بسیار بزرگ یا گراف های داده های بزرگ ظاهر می شود. اخیراً، تکنیکی ارائه شد که از خلاصه سازی تصادفی به منظور کاهش مجموعه داده ها به یک اندازه ی بسیار کوچکتر استفاده می کند [۶۶]. سپس این خلاصه سازی برای تعیین الگوهای نمودار فرعی پرتکرار از داده ها به کارگرفته می شود. محدوده ها بر مبنای مثبت های نادرست و منفی های ناردست و با استفاده از چنین رویکردی استخراج می شود. یک شکل چالشی دیگر زمانی ظاهر می شود که الگوهای پرتکرار در یک گراف خیلی بزرگ نادیده گرفته می شوند به این دلیل که این الگوها ممکن است خودشان گراف های فرعی بسیار بزرگی باشند. الگوریتمی به نام TSMiner معرفی شد تا ساختارهای پرتکرار در گراف های بسیار بزرگ تعیین شوند.

استخراج الگوی گراف دارای کاربردهای فراوانی برای انواع گوناگونی از برنامه های کاربردی است. به عنوان مثال، در مورد داده های برچسب دار از این گونه تکنیک های استخراج الگو می توان به منظور تعیین قوانین طبقه بندی ساختاری استفاده کرد. به عنوان مثال، تکنیک ارائه شده در [۶۷] از این رویکرد برای طبقه بندی داده های XML استفاده می کند. در این حالت، مجموعه داده های متشکل از گراف های (XML) چندگانه داریم که هر یک از آنها یک برچسب طبقه به همراه دارد. روش معرفی شده در [۶۷] قوانین و اصولی را تعیین می کند که ضلع سمت چپ معرف یک ساختار است و ضلع سمت راست معرف یک برچسب طبقه است. از این روش برای اهداف طبقه بندی استفاده می شود. یکی از دیگر کاربردهای استخراج الگوی پرتکرار در [۶۸] مورد مطالعه قرار گرفته است؛ که در آن الگوهای مذکور به منظور ایجاد gBoost مورد استفاده قرار می گیرند. که طبقه بندی کننده ای است که به عنوان یک برنامه کاربردی تقویت کننده طراحی شده است. اثبات شده است که استخراج الگوی پرتکرار به ویژه در حوزه داده های شیمیایی و زیست شناسی مفید است [۶۹-۷۱]. تکنیک های استخراج الگوی پرتکرار برای اجرای توابع مهم در این حوزه از جمله طبقه بندی و تعیین مسیرهای متابولیک مورد استفاده قرار گرفته اند.

استخراج الگوی گراف پرتکرار برای ایجاد شاخص های گراف نیز سودمند است. در [۴۳]، ساختارهای پرتکرار در مجموعه گراف استخراج می شوند به طوری که بتوان از آن ها به عنوان خاصیت هایی برای فرآیند شاخص گذاری استفاده کرد. از تشابه رفتار عضویت الگوی پرتکرار در سراسر گراف ها برای تعیین یک تابع شباهت ابتدایی به منظور انجام فرآیند فیلترینگ استفاده می شود. یک بازنمایی معکوس بر اساس این بازنمایی خاصیت ـ محور ساخته می شود. به این منظور که گراف های نامربوط برای فرآیند جستجوی شباهت فیلتر شوند. تکنیک ارائه شده به واسطه رویکرد خاصیت ـ محور خود کارآمدتر از معیار تکنیک های رقابتی است. به طوری که، الگوریتم های استخراجی الگوی پرتکرار برای هرگونه کاربردی که بر پایه ویژگی های تلفیقی به نحو مؤثر قابل تعیین باشد، مفید خواهند بود. در کل، تکنیک های استخراج الگوی گراف دامنه کاربردی مشابه با آنچه که در مورد استخراج الگوی پرتکرار عادی انجام می دهند، دارند.

۲,۳,۲. الگوریتم های خوشه بندی برای داده های گراف

در این بخش، انواعی از الگوریتم های خوشه بندی داده های گراف را مورد بحث قرار می دهیم. این شامل الگوریتم های سنتی خوشه بندی گراف و الگوریتم های خوشه بندی داده های XML می شود. الگوریتم های خوشه بندی کاربردهای قابل ملاحظه ای در انواع گوناگونی از نقشه های گراف از جمله ارزیابی تراکم، جهت یابی سهولت، و تلفیق داده های XML دارند [۷۲]. در چارچوب الگوریتم های گراف، خوشه بندی می تواند به دو شکل وجود داشته باشد:

- **الگوریتم های خوشه بندی گره:** در این حالت، یک گراف بزرگ داریم و تلاش می کنیم با استفاده از مقادیر فاصله (یا شباهت) نزدیک کران ها، گره های اصلی را خوشه بندی کنیم. در این شرایط، کران های گراف با مقادیر فاصله ای عددی بر چسب زده می شوند. این مقادیر فاصله عددی به منظور ایجاد خوشه هایی از گره ها مورد استفاده قرار می گیرند. یک حالت ویژه این است که در صورت وجود یک کران به آن مقدار تشابه برابر با ۱ نسبت داده می شود، در حالی که در صورت نبود یک کران به آن مقدار تشابه برابر با ۰ داده می شود. لازم به یادآوری است که مسئله به حداقل رساندن تشابه درون ـ خوشه ای برای تعداد ثابتی از خوشه ها لزوماً به مسئله تصمیم بندی گراف یا مسئله برش چند سویه مینیمم تنزل پیدا می کند، و این به مسئله استخراج گراف های متراکم و دسته های ساختگی نیز گفته می شود. اخیراً، مسئله در آثار حوزه پایگاه داده ها به عنوان تعیین دسته های ظاهری (ساختگی) مورد مطالعه قرار گرفته است. در این مسئله، گروه هایی از گره ها را تعیین می کنیم که تقریباً دسته هستند. به عبارت دیگر، یک کران با احتمال بالا بین هر جفت گره مجموعه وجود دارد. گروه های مختلف الگوریتم های خوشه بندی گره را در بخش دیگری مورد مطالعه قرار خواهیم داد.
- **الگوریتم های خوشه بندی گراف:** در این حالت، یک تعداد (احتمالاً برگ) از گراف ها که باید بر اساس رفتار ساختاری خود خوشه بندی شوند، در اختیار داریم. این مسئله از آن جهت چالشی است که تناظر ساختارهای گراف های اصلی و استفاده از این ساختارها برای خوشه بندی ضروری است. هر دو این الگوریتم ها در چارچوب مجموعه داده های گراف سنتی و داده های نیمه ساختاری مورد بررسی قرار می گیرند. بنابراین، هر دوی این شکل ها را بررسی خواهیم کرد.

در زیر بخش های بعدی، هر یک از الگوریتم های خوشه بندی گراف فوق الذکر را مورد بحث قرار خواهیم داد.

الگوریتم های خوشه بندی گره. تعدادی از الگوریتم های خوشه بندی گره گراف در [۲۶] بررسی شده اند. در [۲۶]، مسئله خوشه بندی گره با مسائل برش مینیمم و تقسیم بندی گراف مرتبط شده است. در این حالت، فرض بر این است که گراف های اصلی دارای بارهایی روی کران ها هستند. تقسیم بندی گراف به شیوه ای که بارهای کران ها در عرض تقسیمات به حداقل برسد، مطلوب است. ساده ترین حالت، مسئله برش مینیمم دو ـ راهی است، که در آن می خواهیم گراف را به دو خوشه تقسیم کنیم به گونه ای که بارهای کران ها در عرض تقسیمات به حداقل برسد. این نسخه از مسئله به طور مؤثر قابل حل است و با استفاده از برنامه های کاربردی پی در پی مسئله جریان بیشینه می توان آن را مرتفع کرد؛ زیرا جریان پیشینه بین منبع S و سینک t عامل تعیین برش مینیمم s ـ t خواهد بود. با استفاده از ترکیب متفاوت منبع و سینک، پیدا کردن برش مینیمم کلی امکان پذیر است. روش دوم برای تعیین برش مینیمم با استفاده از رویکرد نمونه برداری کران انجام می شود. این یک تکنیک احتمال گراست که در آن کران ها را به طور توالی نمونه برداری می کنیم به این منظور که گره ها را به مجموعه های بزرگتری از گره ها تجزیه کنیم. با نمونه برداری پی درپی توالی های مختلف از کران ها و انتخاب مقدار مطلوب، تعیین یک برش مینیمم کلی امکان پذیر است. هر دو تکنیک فوق کاملاً کارآمد هستند و دشواری زمانی بر حسب تعداد گره ها و کران های چند فرمولی خواهد بود [۲۶].

مسئله تقسیم بندی چند سویه گراف به طور قابل ملاحظه ای دشوار تر است و NP-hard به شمار می آید. در این مورد، می خواهیم یک گراف را به مؤلفه های K>۲ تقسیم کنیم به طوری که بار کلی کران هایی که انتهای آنها در قسمت های مختلفی قرار گرفته به حداقل برسد. الگوریتم Kerninghan-Lin رویکرد شناخته شده ای برای تقسیم بندی گراف است. این الگوریتم سنتی بر مبنای تپه نوردی (یا تکنیک جستجوی ـ مجاور) برای تعیین تقسیم بندی مطلوب گراف استوار است. در ابتدا، با برش تصادفی گراف شروع می کنیم. در هر تناوب، یک جفت از رأس ها در دو قسمت را مبادله می کنیم تا ببینیم که آیا مقدار کلی برش کاهش می یابد. در صورتی که مقدار برش کاهش یابد، آنگاه جابه جا کردن در دستور کار قرار می گیرد. در غیر این صورت، جفت رأس دیگری را برای جابه جایی انتخاب می کنیم. لازم است یادآوری کنیم که این مطلوب نمی تواند مطلوب کلی باشد، بلکه ممکن است فقط یک مطلوب محلی از داده های اصلی باشد. شکل اصلی در نسخه های مختلف الگوریتم Kerninghan-Lin سیاستی است که برای اجرای جابه جایی رأس ها اتخاذ می شود. باید خاطر نشان کنیم که استفاده از تعداد بیشتری از استراتژی های پیشرفته موجب پیشرفت بهتر در عملکرد ملموس برای هر جابه جایی می شود، اما نیازمند زمان بیشتری برای هر جابه جایی است. این یک تعادل طبیعی است که بسته به نوع برنامه کاربردی که در دست داریم ممکن است به گونه ای متفاوت عمل کند. لازم به ذکر است که مسئله تقسیم بندی گراف به طور گسترده در آثار این حوزه مورد مطالعه قرار گرفته است.

تعیین گراف فرعی متراکم در گراف های بزرگ و گسترده، مسئله ای کاملاً مرتبط است. در مجموعه داده های گراف بزرگ به طور مکرر با این مسئله مواجه می شویم. به عنوان مثال، مسئله تعیین گراف های فرعی بزرگ در گراف های وب در [۷۳] مورد مطالعه قرار گرفت. در این مقاله، از رویکرد عدد ـ مینیمم برای تعیین پلاک هایی که نشان دهنده گراف های فرعی متراکم هستند، استفاده می شود. ایده فراگیر، بیان لینک های خارجی یک گره مشخص به شکل مجموعه هاست. دو گره مشابه هستند که اگر و تنها اگر تعداد زیادی لینک خارجی مشترک داشته باشند. از این رو، گره A با مجموعه لینک های خارجی S_A و گره B با مجموعه لینک های خارجی S_B را در نظر بگیرید. آنگاه، شباهت بین دو گره با استفاده از ضریب Gaccard تعریف می شود، که به شکل $\frac{S_A \cap S_B}{S_A \cup S_B}$ تعیین می شود. باید اشاره کنیم که شمارش علنی تمام کران ها به منظور محاسبه این شباهت کاملاً غیر کارآمد خواهد بود. در عوض، به منظور اجرای برآورد شباهت از رویکرد عدد ـ مینیمم استفاده می شود. این رویکرد به صورت زیر اجرا می شود. فضای گره ها را با یک ترتیب تصادفی تفکیک می کنیم. برای هر یک از مجموعه گره هایی که به صورت تصادفی مرتب شده اند گره نخست یعنی First(A) را تعیین می کنیم که برای آن یک لینک خارجی از A به First(A) وجود دارد. همچنین گره نخست First(B) را تعیین می کنیم که برای آن یک لینک خارجی از B به First(B) وجود دارد. می توان نشان داد که ضریب Jaccard یک برآورد غیر جهت دار این احتمال است که First(A) و First(B) گره مشابهی باشند. با تکرار این فرآیند در جایگشت های مختلفی در فضای گره ها، برآورد دقیق ضریب Jaccard عملی خواهد بود. این کار با استفاده از تعداد ثابتی از جایگشت های مختلفی در فضای گره ها، برآورد دقیق ضریب Jaccard عملی خواهد بود. این کار با استفاده از تعداد ثابتی از جایگشت های C در ترتیب گره انجام می شود. بنابراین، برای هر گره یک اثر انگشت از اندازه C قابل تولید است. با مقایسه اثر انگشت های دو گره می توان ضریب Jaccard را برآورد کرد. این رویکرد با استفاده از هر مجموعه عامل S که فقط دارای S_A و S_B باشد قابل تعمیم است. با استفاده از مقادیر مختلف S و C، طراحی یک الگوریتم که بتواند دو مجموعه بالاتر یا پایین تر از آستانه تعیین شده شباهت را تمیز دهد امکان پذیر خواهد بود.

تکنیک کلی در [۷۳]، ابتدا مجموعه ای از C پلاک با اندازه S برای هر گره می سازد. فرآیند ایجاد پلاک C بسیار ساده است. هر گره به طور جداگانه پردازش می شود. از تابع عدد مینیمم برای ایجاد زیر مجموعه هایی با اندازه S از لینک های خارجی در هر گره استفاده می کنیم. این کار موجب ایجاد C زیر مجموعه برای هر گره می شود. بنابر این، برای هر گره مجموعه ای از پلاک

های C خواهیم داشت. از این رو، اگر گراف دارای n گره باشد، اندازه کل اثر انگشت های این پلاک عبارت است از $n \times C \times SP$، که SP فضای مورد نیاز برای هر پلاک است. به طور معمول، SP به صورت O (S) است، زیرا هر پلاک شامل S گره است. برای هر پلاک مجزا که به این شکل ایجاد شده، می توانیم فهرستی از گره ها را ایجاد کنیم که دارای آن پلاک هستند. به طور کلی ممکن است مایل باشیم گروهی از پلاک ها را تعیین کنیم که حاوی تعداد زیادی از گره های مشترک باشند. به این منظور، روش مطرح شده در [۷۳] یک پلاک سازی درجه دوم را اجرا کنیم که در آن فرا پلاک هایی از پلاک ها ایجاد شوند. از این رو، این کار موجب فشردگی بیش از پیش گراف در یک ساختار داده ها با اندازه ی C×C می شود. این در اصل یک ساختار داده های دارای اندازه ثابت است. لازم به ذکر است که این گروه از فرا پلاک ها دارای این ویژگی هستند که تعداد زیادی از گروه های مشترک را شامل می شوند. آنگاه می توان گراف های فرعی را از این فرا پلاک ها استخراج کرد.

تعیین به ظاهر ـ دسته ها (دسته های ساختگی) در داده های اصلی می تواند مسئله ای مرتبط باشد. به ظاهر ـ دسته ها در اصل تخفیف هایی (کاهش هایی) بر مفهوم دسته ها هستند. در مورد یک دسته، گراف های فرعی که بر مجموعه ای از گره ها القاء شده اند، کامل است. از سوی دیگر، در مورد یک به ظاهر ـ دسته ـ γ، هر رأس در آن زیر مجموعه از گره ها دارای یک رتبه K ـ γ است، که γ یک قطعه است، و K معادل تعداد گره های موجود در آن مجموعه است. اولین قدم در تعیین به ظاهر ـ دسته های ـ γ این است که در آن یک الگوریتم تصادفی به منظور تعیین به ظاهر ـ دسته دارای بزرگترین اندازه مورد استفاده قرار می گیرد. مسئله کاملاً مرتبط با آن، یافتن دسته های پرتکرار در مجموعه داده های چندگانه است. به عبارت دیگر، وقتی گراف های چندگانه از مجموعه داده های مختلفی به دست می آید بعضی از گراف های فرعی متراکم مکرراً در مجموعه داده های مختلف ظاهر می شوند. این گراف ها به تعیین الگوهای متراکم مهم رفتاری در منابع داده های مختلف کمک می کنند. این گونه تکنیک ها در استخراج الگوهای مهم در تصاویر گرافیکی مشتریان کارایی خود را نشان می دهند. تشریح کاربرد این تکنیک در مسئله داده های بیان ـ ژن در [۷۴] قابل مشاهده است.

الگوریتم های سنتی برای خوشه بندی داده های گراف و XML. در این بخش، انواعی از الگوریتم های خوشه بندی داده های گراف و XML را مورد بحث قرار خواهیم داد. باید یادآوری کنیم که داده های XML از نظر چگونگی سازماندهی ساختاری کاملاً مشابه داده های گراف هستند. در [۳۲, ۷۵] نشان داده شده است که استفاده از این رفتار ساختاری اهمیت زیادی در پردازش مؤثر داده ها دارد. دو تکنیک عمده برای خوشه بندی اسناد XML وجود دارد. این تکنیک ها به قرار زیر هستند:

- **رویکرد ساختاری فاصله ـ محور:** این رویکرد به محاسبه فاصله ساختاری بین اسناد مبادرت می کند و برای محاسبه خوشه های اسناد از آن ها استفاده می کند. این گونه رویکردهای فاصله ـ محور، تکنیک هایی کاملاً عمومی و کارآمد هستند که برای انواع گسترده ای از حوزه های غیر عددی مانند داده های زنجیره ای و صریح مورد استفاده قرار می گیرند. به همین خاطر، شناسایی این تکنیک در چارچوب داده های گراف طبیعی به نظر می رسد. یکی از کارهای اولیه در خوشه بندی داده های دارای ساختار نمودار درختی متعلق به الگوریتم XClust است، که برای خوشه بندی چارچوب کلی XML برای تلفیق مؤثر تعداد زیادی از تعاریف اسناد گونه (DTDs) مجزا آغاز می کند و به تدریج دو خوشه ای را که دارای بیشترین تشابه هستند را در یک خوشه بزرگتر ادغام می کند. شباهت بین دو DTDs بر اساس شباهت میانی آنها تعیین می شود، که می توان آن را بر طبق اطلاعات معنایی، ساختاری و مفهومی عامل های موجود در DTDs متناظر محاسبه کرد. یکی از کمبودهای الگوریتم XClust این است که از اطلاعات ساختاری DTDs به طور کامل استفاده نمی کند، که در خوشه بندی ساختارهای درخت - مانند اهمیت فوق العاده ای دارد.

تکنیک خوشه بندی دیگری که در این گروه عمومی از روش ها قرار می گیرد، الگوریتم S-GRACE است. ایده اصلی آن استفاده از رابطه عامل ـ زیر عامل در تابع فاصله به جای استفاده آسان از فاصله ویرایش ـ نمودار درختی است. S-GRACE یک الگوریتم خوشه بندی طبقاتی است. در [۷۵]، یک سند XML به ساختار گراف (یا گراف فرعی) تبدیل شده است، و فاصله بین دو سند XML بر حسب تعداد روابط عامل ـ زیر عامل مشترک تعیین شده است که می تواند در بعضی از موارد روابط شباهت ساختاری بهتری نسبت به فاصله ویرایش نمودار درختی به دست آورد.

- **رویکرد ساختاری خلاصه ـ محور:** در اکثر موارد، ایجاد خلاصه هایی از اسناد اصلی امکان پذیر است. این خلاصه ها برای ایجاد گروه هایی از اسناد که مشابه این خلاصه ها هستند مورد استفاده قرار می گیرند. نخستین رویکرد خلاصه ـ محور برای خوشه بندی اسناد XML در [۷۶] ارائه شده است. در [۷۶]، اسناد XML به عنوان نمودارهای درختی برچسب دار منظم ریشه دار طراحی می شوند. چارچوبی برای خوشه بندی اسناد XML با استفاده از خلاصه های ساختاری ارائه شده است. هدف این چارچوب افزایش کارایی الگوریتم بدون به خطر انداختن کیفیت خوشه بندی است.

رویکرد دوم برای خوشه بندی اسناد XML تحت عنوان XProj شناخته می شود. این تکنیک یک الگوریتم تقسیم بندی است. ایده اولیه در این رویکرد، استفاده از الگوریتم های استخراج الگوی پرتکرار به منظور تعیین خلاصه هایی از ساختارهای پرتکرار در داده هاست. این تکنیک از رویکرد میانگین های ـ K استفاده می کند که در آن هر یک از مرکز های خوشه متشکل از مجموعه ای از الگوهای پرتکرار است که به همان قسمت مشخص شده برای خوشه تعلق دارند. الگوهای پرتکرار با استفاده از اسنادی که در آخرین تناوب به مرکز خوشه اختصاص داده شده اند، استخراج می شوند. آن گاه، براساس شباهت میانگین بین سند و مراکز خوشه جدیداً ایجاد شده از الگوهای پرتکرار محلی، مجدداً اسناد به یک مرکز خوشه تخصیص داده می شوند، تا زمانی که مراکز خوشه و تقسیمات سند در یک وضعیت نهایی تلاقی پیدا کنند. در [۳۲] نشان داده شده است که یک رویکرد ساختاری خلاصه ـ محور برتری چشمگیری بر رویکرد شباهت ـ محور دارد. این روش بر رویکرد ساختاری معرفی شده در [۷۶] نیز برتری دارد زیرا از بازنمایی های خلاصه های ساختاری اصلی بیشتر استفاده می کند.

۲،۳،۳. الگوریتم های طبقه بندی برای داده های گراف

طبقه بندی، کاری محوری در استخراج داده ها و یادگیری دستگاه است. از آن جایی که گراف ها برای بیان واحدها و ارتباط بین آن ها در انواع فزاینده ای از برنامه های کاربردی مورد استفاده قرار می گیرند، موضوع طبقه بندی گراف توجه زیادی را در دانشگاه و صنعت به خود معطوف کرده است. به عنوان مثال، در دارو سازی و طراحی دارو ها، مایلیم به رابطه بین فعالیت یک ترکیب شیمیایی و ساختار آن که توسط یک گراف بیان می شود، پی ببریم. در تحلیل شبکه اجتماعی، ارتباط بین سلامت جامعه (مثلاً در حال گسترش یا کاهش است) و ساختار آن که به صورت گراف ارائه می شود، مورد مطالعه قرار می گیرد.

طبقه بندی گراف مستلزم دو فعالیت یادگیری متفاوت اما مرتبط است.

- **انتقال برچسب.** زیر مجموعه ای از گره ها در گراف بر چسب زده می شوند. فعالیت مورد نظر، یادگیری یک نمونه گره های برچسب دار و استفاده از آن برای طبقه بندی گره های بدون برچسب است.
- **طبقه بندی گراف.** زیر مجموعه ای از گراف ها در یک مجموعه داده های گراف برچسب زده می شوند. این فعالیت به صورت یادگیری یک نمونه از گراف های برچسب دار و استفاده از آن برای طبقه بندی گراف های بدون برچسب انجام می شود.

انتقال برچسب. مفهوم انتقال برچسب یا عقیده [۷۷] یک تکنیک بنیادی است که به منظور به کار بردن ساختار گراف در طبقه بندی داده ها در بعضی از داده های ارتباطی مورد استفاده قرار می گیرد. طرح انتقال برچسب در بسیاری از برنامه های کاربردی یافت می شود. به عنوان مثال، تحلیل شبکه اجتماعی به عنوان ابزاری برای بازاریابی هدفمند مورد استفاده قرار می گیرد. فروشنده های خرده پا به دنبال مشتری هایی هستند که تبلیغات آنها را دریافت کرده اند. این مشتری ها که (با خرید کردن) به تبلیغات پاسخ می دهند به عنوان گره های مثبت در گراف شبکه اجتماعی برچسب زده می شوند و مشتری هایی است که بیشترین احتمال پاسخ گفتن به تبلیغات از جانب آن هاست. فرآیند مذکور را این طور خلاصه می کنیم: یادگیری یک نمونه از مشتری هایی که تبلیغات را دریافت کرده اند و پیش بینی واکنش های سایر مشتری های بالقوه ای که در شبکه اجتماعی حضور دارند. از راه شهودی، می خواهیم به چگونگی انتقال برچسب های مثبت و منفی موجود در گراف به گره های فاقد برچسب پی ببریم.

بر مبنای این فرض که گره های "مشابه" باید برچسب های مشابهی داشته باشند، چالش کلیدی برای انتقال برچسب در طراحی تابع فاصله ای که شباهت بین دو گره در گراف را اندازه گیری کند، نهفته است.

روش های طبقه بندی گراف هسته – محور. این روش بر مبنای عبور تصادفی طراحی شده است. برای هر گراف، مسیرهای آن را شمارش می کنیم و احتمال های این مسیرها را استخراج می کنیم. هسته اصلی گراف به مقایسه مجموعه مسیرها و احتمال های آن ها بین دو گراف می پردازد. مسیر تصادفی (که به صورت توالی بر چسب های گره و کران بیان می شود) از طریق عبور تصادفی ایجاد می شود. نخست، به طور تصادفی یک گره از گراف انتخاب می کنیم. در طی و بعد از هر یک از مراحل بعدی، یا توقف می کنیم و یا یک گره مجانب را برای ادامه عبور تصادفی انتخاب می کنیم، انتخاب های ما در معرض احتمال توقف و احتمال انتقال گره هستند. با تکرار عبورهای تصادفی، جدولی از مسیرها به دست می آوریم که هر یک از آن یک احتمال به همراه دارد.

به منظور اندازه گیری شباهت بین دو گراف، باید شباهت بین گره ها، کران ها و مسیرها را اندازه بگیریم.

- **هسته اصلی گره / کران.** هسته مرکزی هویت، یکی از هسته های گره / کران است. اگر دو گره / کران دارای برچسب مشابهی باشند، آنگاه هسته اصلی عدد ۱ را ارسال می کند و در غیر اینصورت عدد ۰ را می فرستد. اگر برچسب های گره / کران دارای مقادیر حقیقی باشند، آنگاه یک هسته Gaussian قابل استفاده خواهد بود.
- **هسته مرکزی مسیر.** هر مسیر یک توالی از برچسب های گره / کران است. اگر دو مسیر دارای طول یکسان باشند، هسته مرکزی مسیر را می توان به صورت فرآورده هسته مرکزی گره / کران طراحی کرد. اگر دو مسیر دارای طول های متفاوتی باشند، هسته مرکزی مسیر به سادگی عدد ۰ را ارسال می کند.
- **هسته مرکزی گراف.** از آن جایی که هر مسیر با یک احتمال همراه است، می توانیم هسته مرکزی گراف را به صورت احتمال وقوع هسته مرکزی در تمام مسیرهای ممکن در دو گراف تعریف کنیم.

تعریف بالا از هسته مرکزی گراف کاملاً روشن است. با این وجود، شمارش تمام مسیر ها به لحاظ محاسباتی عملی نیست. به ویژه، در گراف های حلقه ای طول یک مسیر نامحدود است که شمارش را غیر ممکن می کند. از این رو، به رویکرد های کارآمد تری برای محاسبه هسته مرکزی نیاز داریم. معلوم می شود که تعریف هسته مرکزی را می توان برای نشان دادن یک ساختار ذخیره ای مجدداً تدوین کرد. در مورد گراف های مستقیم فاقد حلقه، می توان گره ها را ازنظر مکانی مرتب کرد به طوری که هیچ مسیری از گره j به i وجود نداشته باشد اگر j>i، و هسته مرکزی را می توان به صورت یک تابع تناوبی تعریف کرد، و برنامه ریزی

دینامیک می تواند این مسئله را در $O(|X|.|X'|)$ کنترل کند، که X و 'X بیانگر مجموعه گره ها در دو گراف هستند. در مورد گراف های حلقه ای، فضای خاصیت هسته مرکزی (توالی های برچسب) به خاطر لوپ ها (حلقه ها) احتمالاً نامحدود است. محاسبه هسته مرکزی گراف حلقه ای با نظریه سیستم خطی و ویژگی های همگرایی هسته مرکزی قابل اجرا خواهد بود.

روش های طبقه بندی گراف تشدید ـ محور. هر چند روش مبتنی بر هسته مرکزی یک راه حل دقیق برای طبقه بندی گراف ارائه می دهد، صراحتاً مشخص نمی کند که کدام خاصیت های (زیر ساختارهای) گراف متناسب و مرتبط با طبقه بندی هستند. برای پرداختن به این موضوع، رویکرد جدیدی از طبقه بندی گراف بر مبنای استخراج الگو معرفی می شود. اجرای طبقه بندی گراف بر زیر ساختارهای مهم گراف مد نظر است. می توانیم یک بردار خاصیت دوتایی بر اساس حضور یا عدم حضور یک زیر ساختار مشخص ایجاد کنیم و یک طبقه بندی کننده غیر قفسه ای را به کار بگیریم.

از آن جایی که، کل مجموعه گراف های فرعی اغلب بسیار بزرگ است، باید بر زیر مجموعه کوچکی از خاصیت های مرتبط و متناسب تمرکز کنیم. ساده ترین رویکرد برای پیدا کردن خاصیت های جالب از طریق استخراج الگوی پرتکرار انجام می گیرد. با این وجود، الگوهای پرتکرار لزوماً الگوهای مرتبط و متناسب نیستند. به عنوان مثال، در داده های شیمیایی، الگوهای فراگیر مانند C-C یا C-C-C پر تکرارند، اما هیچ اهمیتی در پیش بینی ویژگی های مهم ترکیب های شیمیایی از جمله فعالیت، سم زدایی و غیره ندارند.

از تشدید برای انتخاب خودکار مجموعه ای از گراف های فرعی مانند خاصیت هایی برای طبقه بندی استفاده می شود. LPBoost (تشدید برنامه خطی) یک تابع تشخیص خطی برای انتخاب خاصیت بیان می کند. برای به دست آوردن یک قانون قابل تفسیر، لازم است یک بردار ارزش نا متراکم به دست آوریم که در آن فقط تعدادی از ارزش ها مقادیری غیر از صفر دارند. در [۷۸] نشان داده شده است که تشدید گراف می تواند دقت بهتری از هسته های مرکزی گراف به دست آورد، و دارای مزیت شناسایی زیر ساختارهای کلیدی به طور همزمان است.

مسئله طبقه بندی گراف کاملاً مرتبط با مسئله طبقه بندی XML است؛ به این خاطر که داده های XML را می توان به عنوان نمونه ای از گراف های ارزشمند در نظر گرفت، که در آن گره ها و کران ها دارایی خاصیت هایی همراه با خود هستند. در نتیجه، اکثر روش های طبقه بندی XML برای طبقه بندی گراف های ساختاری نیز قابل اجرا خواهند بود. در [۶۷]، یک طبقه بندی کننده قانون ـ محور (به نام XRules) معرفی شد که در آن خاصیت های ساختاری در ضلع سمت چپ را با برچسب های طبقه روی ضلع سمت راست همراه می کنیم. خاصیت های ساختاری روی ضلع دست چپ با محاسبه خاصیت های ساختاری در گراف که هم پرتکرار و هم تشخیص دهنده برای اهداف طبقه بندی هستند، تعیین می شوند. این خاصیت های ساختاری به منظور ایجاد یک لیست اولویت بندی شده از قوانین مورد استفاده در طبقه بندی به کار گرفته می شوند. قوانین و اصول دارای K بیشینه بر مبنای رفتار تشخیصی تعیین می شوند و اکثریت برچسب های طبقه روی ضلع دست راستی این قوانین K به عنوان نتیجه نهایی گزارش می شوند.

سایر کارهای مرتبط. مسئله طبقه بندی گره در تعدادی از چارچوب های کاربردی مختلف از جمله طبقه بندی داده های ارتباطی، طبقه بندی شبکه اجتماعی و طبقه بندی بلاک دیده می شود. تکنیکی در [۷۹] پیشنهاد شده است که از شباهت اتصال ـ محور برای طبقه بندی گره در چارچوب داده های ارتباطی استفاده می کند. این رویکرد خاصیت های اتصال را از ساختار اصلی ایجاد می کند و به منظور ایجاد یک مدل کار آمد طبقه بندی از این خاصیت ها استفاده می کند. اخیراً، از این تکنیک در طبقه

بندی اتصال ـ محور بلاگ ها نیز استفاده شده است [۶۶]، با این وجود، تمام این تکنیک ها فقط از روش های اتصال ـ محور بهره می گیرند. از آن جایی که اکثر این تکنیک ها در زمینه داده های متنی مطرح می شوند، طبیعی است بررسی کنیم که آیا می توان از این محتوا برای بهبود دقت طبقه بندی استفاده کرد. روشی برای اجرای طبقه بندی گروهی قوانین گفتار ایمیل معرفی شده است که نشان داده شده است که تحلیل جنبه های ارتباطی ایمیل (مانند ایمیل ها در یک زنجیره مشخص) به طور چشمگیری باعث افزایش دقت طبقه بندی می شود. همچنین، در [۸۰] نشان داده شده است که استفاده از ساختار های گراف در طی گروه بندی می تواند دقت طبقه بندی صفحات وب را افزایش دهد.

۲،۳،۴. فعالیت های گراف های تکامل ـ زمان

بسیاری از شبکه ها در برنامه های کاربردی واقعی در چارچوب واحد های شبکه ای مانند وب، شبکه های موبایل، شبکه های نظامی، شبکه های اجتماعی پدیدار می شوند. در این گونه موارد، بررسی جنبه های مختلف فعالیت های تکاملی شبکه های واقعی مانند وب یا شبکه های اجتماعی می تواند مفید واقع شود. از این رو، این گستره تحقیقی بر طراحی ویژگی های تکاملی عمومی گراف های بزرگ که به طور واقعی با آن ها روبه رو می شویم، تمرکز می کند. تحقیق های فراوانی به بررسی ویژگی های تکامل کلی پرداخته اند، ویزگی هایی که در شبکه های فراگیری مانند شبکه های وب، شبکه های فراخوانی و شبکه های اجتماعی معتبر هستند. برخی نمونه های این ویژگی ها عبارتند از:

تراکم. اکثر شبکه های واقعی مثل وب و شبکه های اجتماعی با گذشت زمان متراکم تر می شوند [۸۱]. این قانون اعلام می کند که تعداد گره های موجود در شبکه به طور خطی با تعداد گره ها در طول زمان افزایش می یابد. به عبارت دیگر، اگر n(t) و e(t) بیانگر تعداد کران ها و گره های شبکه در زمان t باشند، آنگاه داریم:

معادله ۱

$$e(t) \propto n(t)^{\propto}$$

مقدار توان $\propto$ بین ۱ و ۲ قابل قبول است.

قطرهای انقباضی. پدیده دنیای کوچک گراف ها به خوبی شناخته شده است. به عنوان مثال، نشان داده شده است که طول میانگین مسیر بین دو کاربر مسنجر MSN حدوداً ۶۰۶ است. این را می توان تأیید قانون شناخته شده "درجه جداسازی" در شبکه های اجتماعی در نظر گرفت. در [۸۱] نشان داده شده است قطرهای شبکه های فراگیر و گسترده ای مانند وب در طول زمان کاهش می یابند. ممکن است تعجب کنید، زیرا انتظار می رود که قطرهای شبکه با افزودن گره ها مرتباً رشد کنند. با این وجود، باید به خاطر داشته باشید که کران ها با سرعت بیشتری از گره ها به شبکه افزوده می شوند (که معادله ۱ آن را بیان می کند). هر چه کران های بیشتری به گراف اضافه می شود عبور از یک گره به گره دیگر با استفاده از تعداد کمتری از کران ها امکان پذیر خواهد بود.

در حالی که تفکر فوق الذکر درک بهتری از بعضی جنبه های کلیدی تکامل دراز مدت گراف های گسترده و فراگیر ارائه می دهد، درباره این که چگونه تکامل در شبکه های اجتماعی را می توان به شیوه ای همه جانبه و فراگیر طراحی کرد هیچ ایده ای مطرح نمی کند. روشی پیشنهاد شده است که از اصل احتمال بیشینه برای ترسیم رفتار تکاملی شبکه های اجتماعی فراگیر استفاده می کند. این روش از استراتژی های داده های ـ استخراج شده برای طراحی رفتار آنلاین شبکه ها استفاده می کند. روش

مذکور به مطالعه رفتار ۴ مدل مختلف شبکه می پردازد، و از اطلاعات این شبکه ها استفاده می کند تا یک مدل تکامل اصلی به وجود آورد. همچنین نشان می دهد که موقعیت کران نقش مهمی در تکامل شبکه های اجتماعی بازی می کند.

یک گستره تحقیق دیگر در این حوزه به مطالعه روش هایی برای توصیف تکامل گراف های خاص باز می گردد. به عنوان مثال، در یک شبکه اجتماعی، ممکن است تعیین جوامع تازه تشکیل شده یا تازه منقرض شده در شبکه های اصلی سودمند باشد [۸۲]. در گزارشی نشان داده شده است که چگونه جوامع در حال گسترش یا در حال انقباض در شبکه اجتماعی ممکن است از طریق بررسی رفتار نسبی کران ها به گونه ای که در یک زنجیره گراف دینامیک (پویا) دریافت شده اند، ترسیم شوند. تکنیک مطرح شده در این گزارش، رفتار ساختاری گراف رشد یابنده درون یک بازه زمانی معین را ترسیم می کند و از آن برای تعیین تولد و مرگ جوامع در زنجیره گراف بهره می گیرد. این نخستین بخش کار است که مسئله تکامل در زنجیره های سریع گراف ها را مورد مطالعه و تحقیق قرار می دهد. به خاطر پیچیدگی و دشواری ترکیبی ذاتی تحلیل ساختاری گراف، که طرح زنجیره را در خود جای نمی دهد، در مطالعه زنجیره با چالش روبه رو خواهیم شد.

روش موجود در [۸۲]، استخراج بدون ـ پارامتر گراف های بزرگ تکاملی در طول زمان را اجرا می کند. این تکنیک می تواند جوامع در حال تکامل در شبکه ها و تغییرات جدی در طول زمان را تعیین کند. یکی از ویژگی های کلیدی این روش این است که فاقد پارامتر است و این قابلیت استفاده از آن در بسیاری از طرح ها را امکان پذیر می کند. با استفاده از اصل MDL در فرآیند استخراج می توان به این هدف دست یافت. یک تکنیک مرتبط نیز قادر است تحلیل بدون پرامتر تکامل در شبکه های گسترده و فراگیر را با استفاده از اصل MDL اجرا کند. این تکنیک می تواند تعیین کند که کدام جوامع در طول زمان منقبض شده اند، منشعب شده اند یا پدید آمده اند.

مسئله تکامل در گراف ها معمولاً در چارچوب خوشه بندی مورد مطالعه قرار می گیرد، زیرا خوشه ها یک خلاصه طبیعی برای درک گراف اصلی و تغییرات در طی فرآیند تکامل در اختیار ما می گذارند. نیاز به این گونه توصیف ها در شرایط شبکه های فراگیر از جمله گراف های برهم کنش، ارزیابی جامعه در شبکه های اجتماعی، و تغییرات کلی خوشه بندی در شبکه های اطلاعاتی اتصال یافته دیده می شود. تحقیق انجام شده توسط [۸۳]، یک چارچوب رویداد ـ محور ارائه می دهد که درک بهتری از رویدادهای معمولی که در شبکه های واقعی در هنگام تشکیل، تکامل یا فروپاشی جوامع رخ می دهد، در اختیار ما می گذارد. از این رو، این روش می تواند روش ساده ای در اختیار ما بگذارد تا به سرعت تعیین کنیم که آیا تغییرات خاصی در یک شبکه خاص روی داده است یا نه. تکنیک مهمی که توسط بسیاری از روش ها مورد استفاده قرار می گیرد تحلیل جوامع در داده ها در برش های زمانی خاص و سپس تعیین تغییر بین برش های زمانی مختلف به منظور تشخیص ماهیت تکامل زیر بنایی است.

استفاده از تکنیک های استخراج قانون ـ محور [۶۶]، رویکردی متفاوت محسوب می شود. الگوریتم مورد نظر یک سلسله عکس از یک گراف در حال تکامل می گیرد، و پس از آن تلاش می کند قوانین و اصولی مشخص کند که تغییرات در گراف اصلی را تعیین کنند. توالی های پرتکرار تغییرات در گراف اصلی به عنوان راهنماهای مهم برای تعیین قانون در نظر گرفته می شوند. به علاوه، الگوهای پرتکرار تجزیه می شوند. به منظور بررسی این اطمینان که یک توالی خاص از گام ها در گذشته منجر به یک جابجایی ویژه می شود. احتمال چنین جابجایی، "اطمینان" نامیده می شود. سپس قوانین در گراف اصلی به منظور توصیف تکامل کلی شبکه مورد استفاده قرار می گیرند.

شکل دیگری از تکامل در شبکه ها در قالب گردش پیام ها (اطلاعات) اصلی ظاهر می شود. از آنجایی که گردش پیام ها و اطلاعات به طور ضمنی یک گراف (زنجیره) را تعیین می کند، فعالیت های این رفتار ممکن است برای بسیاری از برنامه های کاربردی مختلف جالب توجه باشد. این گونه رفتارها اغلب در انواعی از شبکه های اطلاعات از جمله شبکه های اجتماعی، بلاگ ها، یا گراف های نقل قول نویسنده پدید می آیند. در بسیاری از موارد، تکامل ممکن است شکل اطلاعات سرریز شده از گراف های اصلی به خود بگیرد. هدف این است که اطلاعات به وسیله شبکه ای اجتماعی از طریق ارتباط بین واحدهای مختلف شبکه انتقال پیدا کند. تکامل این گردش اطلاعات همزمان با انتشار عیب ها در شبکه، تعدادی شباهت را به اشتراک می گذارد. در بخش بعدی بیشتر درباره این موضوع سخن خواهیم گفت. این تکامل در [۸۴] مورد مطالعه قرار گرفته است، که بررسی می کند چگونه باید رفتار تکامل در گراف های بلاگ را توصیف کرد.

۲,۴. کاربردهای گراف

در این بخش، کاربرد بسیاری از الگوریتم های استخراج پیش گفته در انواع مختلفی از برنامه های کاربردی حوزه گراف را بررسی خواهیم کرد. بسیاری از حوزه های داده ها از قبیل داده های شیمیایی، داده های زیستی، و وب معمولاً به شکل گراف سازماندهی می شوند. بنابراین، طبیعی است که بسیاری از برنامه های کاربردی که پیش تر به بررسی آن ها پرداختیم قابل استفاده در این حوزه ها هستند. در این بخش، برنامه های کاربردی گوناگونی که از کمک تکنیک های استخراج گراف بهره مند می شوند را بررسی خواهیم کرد. همچنین خواهیم دید به رغم این که این برنامه های کاربردی از حوزه های مختلفی به دست آمده اند، رگه های مشترکی وجود دارند که می توان از آن ها برای ارتقاء کیفیت نتایج زیر ساختی بهره گرفت.

۲,۴,۱. برنامه های کاربردی شیمیایی و زیستی

کشف دارو فعالیتی زمان بر و به شدت پرهزینه است. گراف ها، بازنمایی های طبیعی برای ترکیب های شیمیایی هستند. در گراف های شیمیایی، گره ها نشان دهنده اتم ها و کران ها بیانگر پیوند بین اتم ها هستند. گراف های زیستی معمولاً در سطح بالاتری قرار دارند که گره ها بیانگر آمینو اسید ها و کران ها معرف پیوستگی یا ارتباط بین آمینو اسیدها هستند. یک فرایند مهم، که به عنوان اصل رابطه فعالیت ساختار شناخته می شود، این است که ویژگی ها و فعالیت های زیستی ترکیبات شیمیایی به ساختار آن ها بستگی دارد. از این رو، استخراج گراف ممکن است به شناسایی ویژگی های شیمیایی و زیستی مانند فعالیت، سم زدایی، جذب، سوخت و ساز، و غیره کمک کند، و فرآیند ساخت دارو را تسهیل کند. به همین دلیل، تحقیقات دانشگاهی و صنعت داروسازی تلاش هایی در زمینه استخراج گراف ها انجام داده اند به این امید که زمان و هزینه کشف دارو را به طور چشمگیری کاهش دهند.

اگر چه گراف ها بازنمایی طبیعی ساختارهای شیمیایی و زیستی هستند، هنوز هم به بازنماهای کارآمدتری تحت عنوان "معرف" نیاز داریم که منشأ فعالیت هایی است که از جستجوی شباهت تا پیش بینی های مختلف ساختارهای استخراج شده را در بر می گیرد. تنها تعداد اندکی معرف تاکنون معرفی شده اند. به عنوان مثال، اثر انگشت های نشانه [۴۳] نوعی بازنمای برداری هستند. با در نظر گرفتن یک گراف شیمیایی، یک اثر انگشت نشانه را با شمارش انواع معین ساختارهای پایه ای (مانند حلقه ها و مسیرها) در گراف ها و تجزیه آن ها به یک زنجیره کوچک ایجاد می کنیم. در یک تحقیق دیگر، محققان از روش های استخراج داده ها برای پیدا کردن گراف های فرعی پرتکرار در یک پایگاه داده گراف شیمیایی استفاده می کنند و هر گراف شیمیایی را به شکل یک بردار در فضای خاصیت با مجموعه ای از گراف های فرعی پرتکرار بیان می کنند.

جستجوی شباهت یکی از فعالیت های زیر بنایی در ترکیب های شیمیایی است. الگوریتم های تناظر گراف مختلفی برای موارد زیر به خدمت گرفته شده اند: i) بازیابی ـ رتبه، یعنی جستجوی پایگاه داده های بزرگ برای پیدا کردن ترکیب های شیمیایی که دارای فعالیت زیستی مشابهی مثل یک ترکیب مورد سوال هستند؛ و ii) جهش ـ سکو، یعنی پیدا کردن ترکیب های دارای فعالیت زیستی مشابه ولی ساختار متفاوت از ترکیب مورد سوال هستند. جهش ـ سکو برای شناسایی ترکیب هایی به کار می رود که جایگزین خوبی برای ترکیب مورد سوال (مورد مطالعه) هستند، و یا دارای چند ویژگی نامطلوب هستند (مثلاً سم زدایی)، یا از فضای شیمیایی انحصاری موجود آمده اند. از آن جایی که ساختار شیمیایی فعالیت زیستی را تعیین می کند (اصل SAR). "جهش ـ سکو" چالشی پیش روی کاربر قرار می دهد زیرا ترکیب های شیمیایی باید به لحاظ ساختاری آنقدر شبیه باشند که فعالیت زیستی مشابهی به نمایش بگذارند اما در عین حال باید تفاوت آن ها به قدری باشد که یک گونه شیمیایی جدید به شمار آیند. رویکردهای کنونی برای تناظر شباهت را می توان در گروه طبقه بندی کرد. یکی از این گروه ها، تناظر شباهت را مستقیماً در فضای معرف انجام می دهد. گروه دیگر، تناظر غیر مستقیم را اجرا می کند: اگر یک ترکیب شیمیایی C از نظر ساختاری مشابه ترکیب مورد سوال q باشد، و ترکیب شیمیایی 'C از نظر ساختاری مشابه ترکیب C باشد آنگاه 'C و q متناظرهای غیر مستقیم هستند. یعنی به طور غیر مستقیم با یکدیگر تناظر دارند، بدیهی است که تناظر غیر مستقیم مستعد شناسایی ترکیب هایی است که به لحاظ عملکرد مشابه هستند ولی از نظر ساختاری متفاوتند، که این موضوع در "جهش ـ سکو" اهمیت دارد.

پیش بینی ساختار استخراج شده یکی دیگر از حوزه های کاربردی مهم برای استخراج گراف های شیمیایی و زیستی است. هدف این فرآیند، پیش بینی فعالیت یا عدم فعالیت یک ساختار شیمیایی است، یا پیش بینی این که ساختار مورد نظر دارای ویژگی های معین مثل سمی بودن یا غیر سمی بودن، غیره است. ثابت شده که روش های مبتنی بر SVM (سیستم های برداری پشتیبان) در این حوزه کارایی زیادی دارند. توابع هسته ای بر مبنای فضاهای برداری گوناگون از قبیل تابع رایج شعاعی و تابع هسته ای Min-Max برای اندازه گیری شباهت بین ترکیب های شیمیایی که توسط بردارها بیان می شوند، مورد استفاده قرار می گیرند. به جای فعالیت در فضای بردار، گروه دیگری از روش های SVM از هسته های مرکزی گراف برای مقایسه ی دو ساختار شیمیایی بهره می گیرند.

در اواخل ۱۹۸۰، صنعت داروسازی، یک الگوی جدید کشف دارو به نام کشف داروی هدف ـ محور را پذیرفت. هدف آن ساختن دارویی است که اثرات ژن بیماری زا یا فرآورده ژن را بدون تأثیر بر سایر ژن ها یا مکانیسم های مولکولی در موجود زنده کم کند. این هدف احتمالاً استفاده از تکنیک نمایش ظرفیت بالا (HTS) امکان پذیر شده است چرا که تکنیک HTS قادر است تعداد زیادی از ترکیب ها را براساس فعالیت آن ها در مقابل یک هدف معین به سرعت مورد آزمایش قرار دهد. با این وجود، HTS به جای افزایش بازدهی ساخت دارو آن را تا میزان زیادی کاهش می دهد. یکی از دلایل این است که تعداد زیادی از داوطلب های نمایش ممکن است اثرات فنوتیپی غیر قابل قبولی مانند سمی بودن و تعداد روابط داشته باشند که این موارد می توانند در مراحل بعدی کشف دارو هزینه تأیید را افزایش دهند. تکنیک"صید هدف" با استفاده از تکنیک های محاسباتی برای نمایش مستقیم مولکول ها از نظر اثرات فنوتیپ نامطلوب می تواند نقص های فوق الذکر را بر طرف کند.

۲,۴,۲. برنامه های کاربردی Web

وب جهانی طبیعتاً به شکل یک گراف سازماندهی شده است که در آن صفحات وب همان گره ها هستند و لینک ها معادل کران ها هستند. ساختار اتصال وب حجم زیادی از اطلاعات که قابل استفاده در فعالیت های گوناگون استخراج داده ها هستند را در خود نگه می دارد. مشهورترین برنامه کاربردی که از ساختار اتصال وب استفاده می کند الگوریتم PageRank است. این الگوریتم

یکی از اسرار کلیدی در موفقیت موتور جستجوی گوگل است. ایده اصلی پشت الگوریتم PageRank این است که اهمیت یک صفحه روی وب با توجه به تعداد و اهمیت هایپر لینک های معرف آن قابل ارزیابی است. هدف مستقیم الگوریتم طراحی یک موج سوار تصادفی است که لینک های روی صفحات را با احتمال برابر دنبال می کند. بدیهی است که موج سوار به صفحاتی که دارای مسیرهای متعددی هستند بیشتر از سایر صفحات می رسد. تفسیر مستقیم رتبه بندی صفحه عبارتست از: احتمال اینکه یک موج سوار تصادفی در عبور تصادفی به یک صفحه مشخص برسد. بنابراین، رتبه بندی صفحه در اصل یک توزیع احتمال روی صفحات وب تشکیل می دهد، به طوری که مقدار رتبه بندی صفحه روی تمام صفحات وب معادل ۱ است. به علاوه، گاهی اوقات معبر مخابراتی اضافه می کنیم، که در آن می توانیم هر یک از صفحات وب را کاملاً تصادفی جابه جا کنیم.

فرض کنید A مجموعه ای از کران ها در گراف باشد. فرض کنید π_i بیانگر احتمال حالت پایای گره i در عبور تصادفی باشد و $p=(p_{ij})$ نیز معرف ماتریکس انتقال برای فرآیند عبور تصادفی باشد. فرض کنید $\propto$ بیانگر احتمال معبر مخابراتی در یک مرحله معین و q_i بیانگر مقدار i ام بردار احتمال برای تمام گره هایی باشد که احتمال حضور معبر مخابراتی در گره i در هر مرحله معین را مشخص می کنند. در حال حاضر، فرض را بر این می گذاریم که همه مقادیر q_i یکسان و برابر با 1/n باشند، که n معادل تعداد کلی گره ها است. آن گاه برای گره معین i می توانیم رابطه حالت پایای زیر را استخراج کنیم:

معادله ۲

$$\pi_i = \sum_{j:(j,i)\epsilon A} \pi_j . \rho_{ji} . (1-\propto) + \propto . q_i$$

لازم به ذکر است که می توانیم معادله ۲ را برای هر یک از گره ها استخراج کنیم؛ که موجب یک سیستم خطی از معادله های مربوط به احتمال جابه جایی می شود. راه حل ها برای این سیستم باعث ایجاد بردار رتبه بندی صفحه π می شود. این سیستم خطی دارای n متغیر و n محدودیت متفاوت است و به همین خاطر می توان آن را در فضای n^2 در بدترین حالت ممکن تعریف کرد. راه حل برای این سیستم خطی نیازمند عملیات ماتریکس است که در تمام گره ها دست کم درجه ۲ (و حداکثر درجه ۳) است. این می تواند در عمل بسیار پرهزینه باشد. البته، از آن جایی که رتبه بندی صفحه باید در مرحله گروهی فقط یک بار محاسبه و ارزیابی شود، اجرای آن با استفاده از تعداد اندکی تکنیک های ماتریکس دارای طراحی دقیق امکان پذیر خواهد بود. الگوریتم PageRank از یک رویکرد تناوبی استفاده می کند که بردارهای اصلی ماتریکس اتصال متعارف شبکه را محاسبه می کند.

لازم به ذکر است که الگوریتم PageRank در طی فرآیند رتبه بندی فقط به ساختار اتصال توجه می کند و فاقد هرگونه اطلاعاتی درباره محتوی صفحات اصلی وب است. یک مفهوم کاملاً نزدیک مربوط به رتبه بندی حساس به موضوع است که در آن از موضوعات صفحات وب در فرآیند رتبه بندی استفاده می کنیم. ایده اصلی در این گونه روش ها، ایجاد امکان معبر مخابراتی اختصاصی (یا جهش های اختصاصی) در طی فرایند گردش تصادفی است. در هر مرحله از عبور تصادفی مجاز به یک جابه جایی به مجموعه S صفحات که مرتبط با موضوع جستجو هستند، خواهیم بود. در غیر این صورت، عبور تصادفی به شیوه استاندارد خود با احتمال $(\propto -1)$ ادامه پیدا می کند. این عمل به سادگی با تعدیل بردار $q=(q_1 \ldots q_n)$ محقق می شود، به طوری که اجزاء و مؤلفه های مناسب در این بردار را برابر با ۱ و سایر مؤلفه ها را برابر با ۰ قرار می دهیم. آخرین احتمالات حالت پایا با این عبور تصادفی تعدیل شده، رتبه بندی صفحه حساس به موضوع را تعیین می کند. هر چه احتمال $\propto$ بیشتر باشد، آخرین رتبه بندی صفحه بیشتر به سمت مجموعه S سوق پیدا می کند. از آن جایی که هر بردار خصوصی سازی حساس به موضع نیازمند ذخیره یک بردار رتبه بندی بسیار بزرگ است، ممکن است با استفاده از چند نمونه یا صفحات معتبر اقدام به پیش محاسبه بردار به شیوه ای

محدود کند. هدف این است که از تعداد محدودی از این بردارهای خصوصی سازی q استفاده کنیم و بردارهای رتبه بندی صفحه اختصاصی متناظر π را برای این صفحات معتبر تعیین کنیم. ترکیب سنجیده ای از این بردارهای مختلف رتبه بندی اختصاصی (برای صفحات معتبر) به منظور تعیین پاسخ به مجموعه پرس و جوی معین مورد استفاده قرار می گیرد. البته، این رویکرد از نظر سطح غیر یکنواخت که می تواند در آن اختصاصی سازی را اجرا کند با محدودیت هایی مواجه است. در [۸۵] نشان داده شده است که رتبه بندی کاملاً اختصاصی که در آن می توانیم عبور تصادفی را به سمت مجموعه ای دلخواه از صفحات وب سوق دهیم، همواره نیازمند فضای دست کم درجه دوم در بدترین حالت است. بنابراین، رویکرد معرفی شده در [۸۵] نشان می دهد که استفاده از نمونه برداری Monte-Carlo می تواند الزامات و محدودیت های فضا را به شدت کاهش دهد، بدون این که کیفیت را تحت تأثیر قرار دهد. روش مطرح شده در [۸۵] نمونه های Monte-Carlo عبورهای تصادفی ویژه گره را پیش ذخیره می کند، که به عنوان اثر انگشت نیز شناخته می شوند. در [۸۵] نشان داده شده است که در فضای محدود می توان با استفاده از این اثر انگشت ها به سطح بالایی از دقت دست پیدا کرد. کار بعدی که در [۸۶] ارائه شده بر مبنای این ایده در بسیاری از طرح ها بنیان گذاری شده و نشان داده که این گونه تکنیک های رتبه بندی اختصاص دنیامیک را می توان کارآمدتر و مؤثرتر کرد.

سایر رویکردهای مرتبط شامل استفاده از مقیاس هایی مثل زمان برخورد برای تعیین و رتبه بندی مجاورت متن ـ محور گره ها است. زمان برخورد بین گره i و گره j به صورت تعداد جهش های مورد انتظار که یک موج سوار نیاز دارد تا از گره j به گره i برسد، تعریف می شود. بدیهی است که زمان برخورد فقط تابعی از طول کوتاه ترین مسیر نیست، بلکه تابعی از تعداد مسیرهای ممکن که از گره i تا گره j وجود دارند نیز هست. از این رو، زمان برخورد در مقایسه با استفاده از فواصل کوتاه ترین مسیر مقیاس بهتری برای تعیین شباهت بین موارد اتصال یافته، است. نسخه کوتاه شده زمان برخورد، تابع حقیقی را به محدود کردن آن به فاصله هایی که در آن ها زمان برخورد کمتر از آستانه تعیین شده است، مشخص می کند. هرگاه زمان برخورد بالاتر از آستانه تعیین شده باشد، تابع را به سادگی در مقدار آستانه قرار می دهیم. الگوریتم های سریع برای محاسبه شکل کوتاه شده زمان برخورد در [۸۷] مورد بررسی قرار گرفته اند. موضوع قابلیت محاسبه در الگوریتم های عبور تصادفی بسیار جدی و با اهمیت است زیرا این گراف ها بزرگ و دینامیک هستند و ما تمایل داریم قابلیت و توانایی رتبه بندی سریع انواع ویژه ای از پرس و جو را کسب کنیم. روش ارائه شده در [۸۸] تکنیک باز ـ رتبه بندی دینامیک سریعی معرفی می کند برای زمانی که واکنش کاربر به ثبت رسیده است. مسئله مرتبط با آن، بررسی رفتار عبورهای تصادفی از طول های ثابت و معین است.

پرس و جوی ترکیبی را می توان "نسخه وارونه" زمان برخورد در نظر گرفت، که تعداد جهش ها را ثابت می کنیم و سعی می کنیم به جای تعداد جهش های برخورد تعداد برخوردها را تعیین کنیم. یکی از مزیت های این کار این است که به طور خودکار فقط عبورهای تصادفی کوتاه شده را که در آن طول عبور پایین تر از آستانه تعیین شده h است را در نظر می گیرد؛ همچنین، به لحاظ بررسی عبورهای مختلف به شیوه ای یکپارچه؛ تعیین بهتری در مقایسه با زمان برخورد کوتاه شده است. روش ارائه شده در [۸۹] گره هایی را مشخص می کند که با استفاده از یک چارچوب ترکیب مجاورهای محلی به نام (Local Neigborhood Aggregation) LONA، دارای بالاترین مقادیر K - بیشینه در تمام مجاورهای جهش - h هستند. این چارچوب از ویژگی های مکانی در فضای شبکه برای ایجاد یک شاخص کار آمد برای پرس و جو بهره می گیرد.

تکنیک رتبه بندی صفحه با استفاده از رفتار اتصال به صورت نشانه مرجع به تعیین مرجع می پردازد. روش دیگری پیشنهاد می کند که صفحات وب یکی از دو نوع زیر خواهند بود:

- **قطب ها** صفحاتی هستند که به صفحات معتبر متصل می شوند.

- **مرجع ها** صفحاتی هستند که به وسیله قطب های معتبر متصل می شوند.

هر امتیاز، قطب ها و مرجع ها را مطابق با اعتبار آنها برای قطب بودن یا مرجع بودن به همراه دارد. امتیازات قطب ها بر امتیازات مرجع ها اثر می گذارد و بالعکس. از رویکرد متناوبی به منظور محاسبه امتیازات قطب ها و مرجع ها استفاده می شود. الگوریتم HITS از این دو امتیاز برای محاسبه قطب ها و مرجع های موجود در گراف وب استفاده می کند.

بسیاری از این برنامه های کاربردی در گراف های دینامیک که گره ها و کران های گراف در طول زمان دریافت می شوند، مطرح می شوند. به عنوان مثال، در مورد شبکه اجتماعی که در آن لینک های جدید به طور پی در پی و مداوم ایجاد می شوند، ارزیابی رتبه بندی صفحه ذاتاً یک مسئله دینامیک (پویا) است. از آن جایی که الگوریتم رتبه بندی صفحه به شدت وابسته به رفتار گردش های تصادفی است، الگوریتم رتبه بندی زنجیره ای صفحه [۵۸]، گره ها را به طور جداگانه نمونه برداری می کند به این منظور که گردش های تصادفی کوتاه از هر یک از گره ها ایجاد کند. این گردش ها را می توان بعداً در هم ادغام کرد تا گردش های تصادفی طولانی تری ایجاد شود. با راه اندازی چندین گردش تصادفی به این شکل، می توان رتبه بندی صفحه را با کارایی بالا ارزیابی کرد؛ زیرا رتبه بندی صفحه صرفاً احتمال مشاهده یک گره در گردش تصادفی است و الگوریتم نمونه برداری می تواند این فرآیند را به خوبی شبیه سازی کند. چالش جدی پیش روی این الگوریتم این است که ممکن است در طی فرآیند گردش تصادفی گیر کند. این به این خاطر است که فرآیند نمونه برداری، هم گره ها و هم کران ها را به عنوان نمونه انتخاب می کند و احتمال دارد یک کران به گونه ای پیموده شود که نقطه انتهایی آن در نمونه گره موجود نباشد. علاوه بر این، مجاز به عبور تکراری از گره ها به منظور حفظ و تداوم تصادفی بودن نیستیم. این گره های گیر افتاده را می توان با نگه داشتن حساب مجموعه S از گره های نمونه برداری شده که گردش آن ها قبلاً برای تعمیم گردش تصادفی به کار گرفته شده، کنترل کرد. کران های جدید از گره های گیر افتاده و گره های مجموعه S نمونه برداری می شوند. از این نمونه ها استفاده می شود به این منظور که تا جایی که امکان دارد گردش ها تعمیم داده شوند. اگر نقطه ـ انتهایی جدید به شکل یک گره نمونه برداری شده ظاهر شود، آن گاه فرآیند ادغام گره ها را ادامه می دهیم. در غیر این صورت، فرآیند نمونه برداری کران ها از مجموعه S و تمام گره های گیر افتاده ای که از آخرین گردش تا به حال مشاهده شده اند را تکرار می کنیم.

تحلیل لوگ های جریان پرس و جو، رویکرد دیگری است که در چارچوب استخراج گراف عموماً با آن مواجه می شویم لازم به ذکر است که شیوه ای متداول که بسیاری از کاربرها برای جهت یابی در وب به کار می گیرند استفاده از موتورهای جستجو برای شناسایی صفحات وب و پس از آن کلیک روی بعضی از هایپر لینک های موجود در نتایج جستجو است. از رفتار گراف های به دست آمده می توان برای تعیین پراکندگی موضوع و ارتباط های معنایی بین موضوعات مختلف استفاده کرد.

در بسیاری از برنامه های کاربردی وب، تعیین خوشه های صفحات وب یا بلاگ ها سودمند خواهد بود. به این منظور، کمک گرفتن از ساختار اتصال وب می تواند مفید واقع شود. تکنیک متداولی که اغلب برای خوشه بندی اسناد وب مورد استفاده قرار می گیرد تکنیک تخته پوش کردن است [۷۳]. در این حالت، از رویکرد نشانه ـ مینیمم برای تعیین نواحی به شدت پیوسته در وب مورد استفاده قرار می گیرد. علاوه بر این، از هر یک از تکنیک های ایجاد به ظاهر ـ دسته ها [۷۴] می توان برای تعیین نواحی متراکم گراف بهره گرفت.

شبکه اجتماعی. شبکه های اجتماعی در اصل گراف های بسیار بزرگی هستند که با توجه به افرادی که به شکل گره ها ظاهر می شوند و لینک هایی که متناظر با ارتباطات یا روابط بین این افراد هستند تعریف می شوند. از لینک های موجود در شبکه

اجتماعی می توان برای تعیین جوامع مرتبط، اعضاء با مجموعه های تخصصی ویژه، و جریان اطلاعات در شبکه اجتماعی استفاده کرد. این کاربردها را یک به یک مورد بررسی قرار خواهیم داد.

مسئله بررسی جامعه در شبکه های اجتماعی به مسئله خوشه بندی گره در گراف های خیلی بزرگ مرتبط است. در این حالت مایلیم خوشه های متراکم گره ها را بر اساس ساختار اصلی اتصال مشخص کنیم [۹۰]. شبکه های اجتماعی به واسطه اندازه بزرگ گراف های اصلی به چالش ویژه در مسئله خوشه بندی تبدیل شده اند. مانند گراف های وب، هر یک از روش های ایجاد تخته پوش یا به ظاهر ـ دسته ها [۷۳, ۷۴] را می توان برای تعیین جوامع مرتبط در شبکه به کار گرفت. تکنیکی در [۹۱] به منظور استفاده از محرک های جریان تصادفی در تعیین خوشه ها در گراف های اصلی معرفی شده است. روش برای تعیین ساختار خوشه بندی با استفاده از ساختار ـ ایجن ماتریکس اتصال به منظور تعیین ساختار اجتماع، در [۸۷] ارائه شده است. یکی از ویژگی های مهم شبکه های بزرگ این است که اغلب می توان آن ها را با توجه به ماهیت گراف های فرعی زیر بنایی توصیف کرد، در [۹۲]، تکنیکی برای محاسبه تعداد گراف های فرعی گونه خاصی از شبکه های بزرگ معرفی شده است. نشان داده شده است که این توصیف در خوشه بندی شبکه های بزرگ مفید است. این دقت و وضوح با استفاده از ویژگی های مکانی محقق نمی شود. بنابراین، این رویکرد برای بررسی اجتماع در شبکه های فراگیر و بزرگ نیز قابل استفاده است. مسئله بررسی اجتماعی به ویژه در مورد تحلیل دنیامیک شبکه های در حال تکامل که در آن تلاش می کنیم چگونگی تغییر جوامع شبکه در طول زمان را مشخص کنیم. بسیاری از تکنیک ها به ارزیابی مسئله بررسی اجتماعی و بررسی تغییر در یک چارچوب می پردازند. با این کار قادر خواهیم بود تغییرات در شبکه اصلی را به شیوه کوتاه و خلاصه ارائه دهیم.

الگوریتم های خوشه بندی گره کاملاً مرتبط با مفهوم تحلیل مرکزیت در شبکه ها است. به عنوان مثال، تکنیکی که در [۹۰] مورد بحث واقع شده، از یک رویکرد K-medoids استفاده می کند که K نقطه مرکزی شبکه را نشان می دهد. این نوع رویکرد ها در شبکه های مختلف مفید واقع می شوند، حتی اگر چارچوب شبکه ها نیز با هم متفاوت داشته باشند. در مورد شبکه های اجتماعی، این نقطه های مرکزی به طور معمول اعضا کلیدی در شبکه هستند که به خوبی به سایر اعضاء اجتماع متصل شده اند. از تحلیل مرکزیت می توان برای تعیین نقطه های مرکزی در جریان های اطلاعات نیز بهره گرفت. از این رو، بدیهی است که گونه مشابهی از الگوریتم تحلیل ساختاری می تواند منجر به دیدگاه های متفاوت در شبکه های مختلف شود.

ارزیابی مرکزیت ارتباط نزدیکی با مسئله جریان اطلاعات منتشر شده در شبکه های اجتماعی دارد. مشاهده شده است که اکثر تکنیک های تحلیل جریان ویروسی که اخیراً طراحی شده اند در چارچوب انواع مختلفی از برنامه های کاربردی مرتبط با جریان اطلاعات شبکه اجتماعی قابل استفاده اند. این کاربرد به این خاطر است که برنامه های کاربردی مرتبط با جریان اطلاعات با مدل های رفتاری مشابه با انتشار ویروس ها قابل درک هستند. این برنامه های کاربردی عبارتند از: (۱) ما تمایل داریم مؤثرترین اعضاء شبکه اجتماعی را مشخص کنیم؛ یعنی اعضایی که باعث جریان بیشینه اطلاعات به خارج می شود. (۲) اطلاعات در رفتار اجتماعی اغلب به شیوه ای مشابه اپیدمی سرریز می کند و منتشر می شود. ما تمایل داریم میزان سر ریز اطلاعات از طریق شبکه اجتماعی را اندازه گیری کنیم و اثر منابع مختلف بر اطلاعات را تعیین کنیم. هدف این است که نظارت موجب ارتقاء ارزیابی اولیه از جریان های اطلاعات شود و این برای فردی که بتواند آن را ارزیابی کند مفید خواهد بود. رفتار سرریز و انتشار به ویژه در مورد گراف بلاگ که در آن رفتار سرریز و انتشار به شکل لینک های افزوده در طول زمان بازتاب می یابد، قابل مشاهده است. از آن جایی که کنترل و نظارت بر تمام بلاگ ها به طور همزمان امکان پذیر نیست، به حداقل رساندن هزینه نظارت بر بلاگ های مختلف با فرض یک هزینه ثابت به ازای هر گره مطلوب خواهد بود، این مسئله NP ـ دشوار است، زیرا مسئله پوشش ـ رأس را می توان به آن

تقلیل داد. ایده اصلی در [۸۴]، استفاده از اکتشاف تقریبی به منظور به حداقل رساندن هزینه است. این رویکرد به طرح بلاگ محدود نمی شود بلکه برای سایر طرح ها از جمله نظارت بر تبادل اطلاعات در شبکه های اجتماعی، و نظارت بر قطع جریان در شبکه های ارتباطی نیز قابل استفاده است. (۳) ما می خواهیم شرایطی که منجر به نیاز فوری برای انتقال غیر کنترل شده اطلاعات می شوند را مشخص کنیم.

سایر برنامه های شبکه کامپیوتری. اکثر این تکنیک ها را می توان برای انواع دیگری از شبکه ها مانند شبکه های ارتباطی نیز مورد استفاده قرار داد. تحلیل ساختاری و توان شبکه های ارتباطی تا حد زیادی به طراحی گراف اصلی شبکه بستگی دارد. طراحی دقیق گراف اصلی می تواند در جلوگیری از نقص های شبکه، تراکم، و سایر نقاط ضعف در کل شبکه مؤثر واقع شود. به عنوان مثال، تحلیل مرکزیت [۹۰] در مورد یک شبکه اجتماعی برای تعیین نقاط حساس نقص و ناکارآمدی مورد استفاده خواهد بود. همچنین، تکنیک هایی برای انتشار جریان اطلاعات در شبکه های اجتماعی برای مدل انتشار ویروس در شبکه های اجتماعی نیز قابل استفاده خواهد بود. تفاوت عمده در این است که احتمال آلودگی به ویروس را در امتداد یک کران در شبکه اجتماعی طراحی می کنیم به جای اینکه تلاش کنیم احتمال جریان اطلاعات را در امتداد یک کران در شبکه اجتماعی مدل سازی کنیم.

اکثر تکنیک های قابلیت دسترسی [۴۷-۴۹] برای تعیین تصمیمات استخراج بهینه در شبکه های کامپیوتری نیز کاربرد خواهد داشت. این به مسئله تعیین اتصال ـ گره جفت محور [۹۳] در شبکه های کامپیوتری نیز مرتبط است. تکنیک ارائه شده در [۹۳] از خلاصه سازی بر مبنای فشرده کردن برای ایجاد یک شاخص اتصال مؤثر در گراف های فراگیر ذخیره شده روی دیسک استفاده می کند. این کار می تواند در شبکه های ارتباطی مفید واقع شود که در آن ها نیاز است تعداد مینیمم کران هایی که باید به منظور قطع ارتباط یک جفت ویژه از گره ها حذف شوند را مشخص کنیم.

۲,۴,۳. مکان یابی حفره های نرم افزاری

یکی از کاربردهای طبیعی الگوریتم های استخراج گراف مربوط به مکان یابی حفره های نرم افزاری از نقطه نظر اعتبار و دشواری نرم افزار یکی از برنامه های کاربردی مهم به شمار می آید. جریان کنترل برنامه ها را می توان در قالب گراف های پیام (خبر) مدل سازی کرد. هدف تکنیک های مکان یابی حفره های نرم افزاری، استخراج این گراف های پیام به منظور تعیین حفره ها در برنامه های نرم افزاری است. گراف های پیام به دو گروه تقسیم می شوند:

- **گراف های پیام استاتیک (ثابت)** از کد منبع یک برنامه مشخص قابل استنباط است. تمام روش ها، پروسه ها، و عملکردها در برنامه به شکل گره ها وجود دارند و رابطه بین روش های مختلف به صورت کران ها تعریف می شوند. همچنین، تعیین گره ها برای عناصر داده ها، و طراحی روابط بین عناصر داده های مختلف و کران ها نیز امکان پذیر است. در مورد گراف های پیام استاتیک، اغلب استفاده از نمونه های واقعی ساختار برنامه به منظور تعیین قسمت هایی از نرم افزار که ممکن است در آن پیشامدهای غیر واقعی روی دهد، امکان پذیر است.
- **گراف های پیام دینامیک (پویا)** در طی اجرای برنامه ایجاد می شوند و بیانگر ساختار تقاضا هستند. به عوان مثال، یک پیام از پروسه ای به پروسه دیگر موجب کرانی می شود که معرف رابطه تقاضا بین دو پروسه است. این گونه گراف های پیام می توانند در برنامه های نرم افزاری فراگیر به شدت بزرگ و گسترده باشند، زیرا این برنامه ها شامل هزاران تقاضا بین پروسه های مختلف می باشند. در این گونه موارد، از تفاوت در رفتار ساختاری، فراوانی توالی تقاضاهای موفق و ناموفق

می توان برای مکان یابی حفره های نرم افزاری استفاده کرد. این گراف های پیام به ویژه در مکان یابی حفره های تصادفی که ممکن است در بعضی تقاضاها و نه در همه آن ها روی دهند، سودمند هستند.

متذکر می شویم که مکان یابی حفره ها از نظر انواع خطاهایی که می تواند به دام اندازد جامع و فراگیر نیست. به عنوان مثال، خطاهای منطقی در یک برنامه که ناشی از ساختار برنامه نباشند و توالی یا ساختار اجرای روش های مختلف را تحت تأثیر قرار ندهند قابل مکان یابی با این تکنیک ها نیستند. علاوه بر این، مکان یابی حفره نرم افزاری مهارتی دقیق محسوب نمی شود. تا اندازه ای می توان از این تکنیک برای ایجاد تست نرم افزاری که متخصص حفره های موجود باشد استفاده کرد، و این تست ها می توانند از این تکنیک برای تصحیح ها و اصلاحات مرتبط استفاده کنند.

یکی از موارد جالب، حالتی است که در آن اجراهای مختلف برنامه منجر به ساختار، توالی و فراوانی تکنیک هایی می شود که مختص ناکامی ها و موفقیت های اجرای نهایی برنامه می شود. این ناکامی ها و موفقیت ها ممکن است نتیجه خطاهای منطقی باشد که موجب تغییراتی در ساختار و توالی پیام ها می شود. در این گونه موارد، مکان یابی حفره نرم افزاری به صورت یک مسئله طبقه بندی قبل طراحی است. مرحله اول شامل ایجاد گراف های پیام از تکینک های اجرایی است. این پروسه با جستجوی تکنیک های اجرای برنامه در فرآیند تست و آزمایش محقق می شود. لازم به ذکر است که این دسته از گراف های پیام ممکن است برای کار با الگوریتم های استخراج گراف بیش از اندازه غول پیکر و گسترده باشند. اندازه های بزرگ گراف های پیام، چالش برای پروسه های استخراج گراف پدید می آورد؛ زیرا الگوریتم های استخراج گراف اغلب برای گراف های نسبتاً کوچک طراحی شده اند، در حالی که اندازه گراف های پیام ممکن است بسیار بزرگ باشد. بنابراین، کاهش اندازه گراف های پیام با استفاده از رویکرد مقایسه ـ محور می تواند راه حل طبیعی برای چالش فوق باشد. این کاهش به طور طبیعی منجر به مفقود شدن اطلاعات می شود و در بعضی از موارد که صدمه به اطلاعات همه جانبه و گسترده باشد باعث ضعف و ناتوانی در استفاده مؤثر از رویکرد مکان یابی می شود.

گام بعدی شامل استفاده از الگوریتم های استخراج گراف فرعی در داده های آموزشی به منظور تعیین الگوهایی است که به طور مکرر در تکنیک های اجرایی پر اشتباه ظاهر می شوند. باید خاطر نشان کنیم که این تا حدودی شبیه به تکنیکی است که اغلب در طبقه کننده های اصل ـ محور که سعی می کنند الگوهای خاص و شرایط ویژه را به برچسب های طبقه خاص متصل کنند، به کار گرفته می شود. سپس این الگوها با روش های مختلفی همراه می شوند و برای ایجاد فرآیند رتبه بندی روش ها و تابع های گوناگون در برنامه های دارای حفره های نرم افزاری مورد استفاده قرار می گیرند. این کار به آشنایی و درک حفره های موجود در برنامه های زیر بنایی نیز منجر خواهد شد.

باید اشاره کنیم که فرآیند فشرده سازی در ایجاد توانایی برای پردازش مؤثر گراف های اصلی از اهمیت ویژه ای برخوردار است. یکی از روش های طبیعی برای کاهش اندازه گراف های متناظر این است که گره های چندگانه در گراف پیام را با یک گره مجزا انطباق دهیم. به عنوان مثال، در فرآیند کاهش کلی، تمام گره های موجود در گره پیام را با هم تطبیق می دهیم که با همان روش در گره گراف فشرده متناظر است. از این رو، تعداد کلی گره ها در نمودار حداکثر برابر با تعداد روش هاست. به منظور کاهش اندازه گراف پیام از چنین تکنیکی در [۹۴] استفاده شده است. دومین روشی که ممکن است مورد استفاده قرار گیرد فشرده سازی ساختارهای اجرایی تناوبی مانند لوپ ها (حلقه ها) در یک گراف مجزا است. این یک رویکرد طبیعی است، در حالی که ساختار اجرایی تناوبی یکی از متداول ترین بخش های گراف های پیام است. تکنیک دیگر، تبدیل نمودارهای درختی فرعی به گره های

مجزاست. انواع مختلفی از استراتژی های مکان یابی که از این گونه تکنیک های تبدیلی و کاهشی استفاده می کنند در [۹۵-۹۷] بررسی شده اند.

در نهایت، گراف های تبدیل شده استخراج می شوند به این منظور که ساختارهای تشخیص دهنده برای امکان یابی حفره ها تعیین شوند. روش ارائه شده در [۹۷] بر اساس تعیین نمودارهای درختی فرعی از داده ها عمل می کند. به بیان دقیق تر، این روش تمام گراف های درخت فرعی که در اجرا های ناموفق پرتکرارند ولی در اجراهای صحیح پرتکرار نیستند را پیدا می کند. سپس، از آن ها برای ایجاد قوانینی که ممکن است برای نمونه های خاصی از راه اندازی های برنامه های طبقه بندی مورد استفاده قرار گیرند، استفاده می شود. مهم تر این که، این قوانین شناخت و درک بهتر از تصادفی بودن حفره ها در اختیار ما می گذارد و این شناخت می تواند برای تأیید اصلاح خطا های زیر بنایی مورد استفاده قرار گیرد.

تکنیک فوق برای پیدا کردن ویژگی های ساختاری تکنیک اجرایی که در ایزوله کردن حفره های نرم افزاری مورد استفاده قرار می گیرد طراحی شده است. با این وجود، در اکثر موارد ویژگی های ساختاری ممکن است تنها خاصیت هایی باشند که با مکان یابی حفره ها ارتباط دارند. به عنوان مثال، خاصیت مهمی که ممکن است در تعیین حفره های به کار گرفته شود "فراوانی نسبی" تقاضای روش های مخلف است. مثلاً، تقاضاهایی که دارای حفره اند ممکن است روش ویژه ای که پر تکرارتر از سایرین است را فرا بخوانند. شیوه ای معمول برای یادگیری این روش این است که بارهای کران را به گراف پیام مربوط کنیم. این بارها با فراوانی تقاضا متناظرند. سپس، از این بارهای کران برای تحلیل تقاضاهایی که بیشترین تناسب را با متمایز کردن تکنیک های اجرایی موفق و ناموفق دارند، استفاده می کنیم.

لازم به ذکر است که ساختار و فراوانی دو جنبه متفاوت از داده ها هستند که برای اجرای مکان یابی حفره ها می توان آن ها را در هم ادغام کرد. بنابراین، تلفیق این رویکردها به منظور بهبود فرآیند مکان یابی کاملاً منطقی به نظر می رسد. تکنیک های ارائه شده در [۹۵, ۹۶] امتیازی برای خاصیت های ساختار ـ محور و فراوانی ـ محور به وجود می آورد. آن گاه، ترکیب این امتیاز ها برای فرآیند مکان یابی حفره مورد استفاده قرار می گیرد. نشان داده شده است که این رویکرد مؤثرتر و کار آمدتر از استفاده از یکی از دو خاصیت (یا ساختار و یا فراوانی) است.

ویژگی مهم دیگری که در کارهای آینده قابل بررسی و پژوهش است، تحلیل توالی پیام های برنامه است به جای اینکه به سادگی به تحلیل ساختار پیام دینامیک یا فراوانی پیام های روش های مختلف بپردازیم. برخی کارهای اولیه در این راستا نشان می دهد که استخراج توالی می تواند اطلاعات فوق العاده ای را برای مکان یابی نرم افزار حتی با بهره گیری از روش های ساده به رمز در آورد.

با این وجود؛ این تکنیک های استخراج گراف پیشرفته برای تلفیق این توالی اطلاعاتی بهره نمی گیرد. بنابراین، این می تواند مسیری پربار و پرفایده برای تحقیق های آینده در زمینه ادغام اطلاعات زنجیره ای و متوالی با تکنیک های استخراج گراف موجود باشد.

تحلیل کد منبع استاتیک به جای گراف های پیام دینامیک یکی دیگر از خطوط کلی تحلیل است. در چنین شرایطی، توجه به گروه های خاصی از حفره های نرم افزاری به جای سعی در ایزوله کردن منشأ خطای اجرایی منطقی تر خواهد بود. به عنوان مثال، حالت های فراموش شده در برنامه های نرم افزار [۹۸] می تواند موجب ضعف و کمبود شود. به عنوان مثال، گزارش وضعیت در برنامه نرم افزاری به همراه یک حالت مفقود شده یکی از مداول ترین حفره های نرم افزاری است. در این شرایط، طراحی

تکنیک های مختص به این حوزه برای مکان یابی حفره ها طبیعی به نظر برسد. برای این هدف، تکنیک هایی بر مبنای گراف های استاتیک وابسته به برنامه به کار گرفته می شوند. این گراف ها متفاوت از گراف های دنیامیکی هستند که در بالا مورد بررسی قرار گرفتند؛ به تعبیری مورد دومی نیازمند اجرای برنامه برای ایجاد گراف هاست، در حالی که در مورد اول گراف های استاتیک به شیوه ای استاتیک ساخته می شوند. گراف های وابسته به برنامه در اصل یک بازنمایی گرافیکی از رابطه های بین روش های مختلف و عناصر داده های یک برنامه ایجاد می کنند. انواع مختلف و متفاوتی از کران ها برای توصیف کنترل و وابستگی های داده ها مورد استفاده قرار می گیرند. گام اول، تعیین قوانین شرطی [۹۸] در برنامه ای است که وابستگی های پرتکرار در پروژه را شرح می دهد. سپس به دنبال تحریک های استاتیکی درون پروژه می گردیم که این قوانین را نقض می کنند. در اکثر موارد، این تحریک ها متناظر با حالت های نادیده گرفته شده در برنامه نرم افزاری هستند.

حوزه مکان یابی حفره نرم افزاری با تعدادی چالش کلیدی روبروست. یکی از چالش های عمده این است که کار در این حوزه عمدتاً بر پروژه های نرم افزاری کوچک تر تمرکز می کند. برنامه بزرگتر و فراگیرتر چالش محسوب می شوند زیرا ممکن است گراف های پیام متناظر بسیار بزرگ و غول پیکر باشند و فرآیند فشرده سازی گراف ممکن است حجم زیادی از اطلاعات را از دست بدهد. در حالی که بعضی از این چالش ها ممکن است با طراحی تکنیک های استخراج کارآمد تا اندازه ای کم اثر شوند، چندین مزیت نیز ممکن است با استفاده از بازنمایی بهتر "سطح مدل سازی" کسب شوند. به عنوان مثال، گره های گراف را می توان در سطح پایین تری از غیر یکنواختی در فاز مدل سازی بیان کرد. از آن جایی که فرآیند مدل سازی با درک بهتر احتمالات بروز حفره های نرم افزاری محقق می شود (که با فرآیند فشرده سازی خودکار قابل مقایسه است). فرض بر این است که این رویکرد اطلاعات کمتری را در فرآیند مکان یابی حفره ها از دست می دهد. هدف دوم، تلفیق تکنیک های گراف ـ محور با سایر تکنیک های آماری مؤثر به منظور ایجاد طبقه بندی کننده های منسجم تر و قوی تر است. در آینده، منطقی است انتظار داشته باشیم که تحلیل پروژه های نرم افزاری بزرگتر فقط با استفاده از این تکنیک های تلفیقی که قادرند از ویژگی های مختلف داده های اصلی استفاده کنند، امکان پذیر شود.

۲,۵. نتیجه گیری

در این فصل، یک نمای کلی از برنامه های کاربردی در حوزه استخراج و مدیریت گراف در اختیار شما قرار دادیم. همچنین، ارزیابی از برنامه های کاربردی متداول که از کاربردهای استخراج گراف نشأت می گیرند، ارائه دادیم. بخش اعظم فعالیت ها در سال های اخیر بر گراف های کوچک و قابل ذخیره سازی در حافظه تمرکز کرده اند. بسیاری از چالش های آینده در مورد گراف های بسیار بزرگی که روی دیسک ذخیره می شوند پدید می آید. برنامه های کاربردی دیگری در چارچوب زنجیره های گراف فراگیر و گسترده مطرح می شوند. زنجیره های گراف در چارچوب بعضی از برنامه های کاربردی از قبیل شبکه اجتماعی پدید می آیند که ارتباطات بین گروه های بزرگی از کاربران به شکل گراف ها ذخیره می شوند. این گونه برنامه ها بسیار چالشی هستند زیرا نمی توان کل داده ها را به منظور تحلیل ساختاری روی دیسک مکان یابی کرد. از این رو، به تکنیک های جدیدی برای خلاصه سازی رفتار ساختاری زنجیره های گراف نیاز داریم، و می توان از این تکنیک ها برای انواع مختلفی از نقشه های تحلیلی بهره گرفت. انتظار داریم که تحقیق آینده بر طرح های مقیاس ـ بزرگ و زنجیره ـ محور برای استخراج گراف تمرکز کند.

فصل سوم: استخراج زیرگراف های پرتکرار

۳,۱. مقدمه

هدف اولیه استخراج داده ها، استخراج کلان و مفید (به لحاظ آماری) دانش و اطلاعات از داده هاست. 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 این ها 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. 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 ۲ فهرست شده اند.

جدول ۲: زیر مجموعه های حوزه های تحقیقی متداول در استخراج گراف

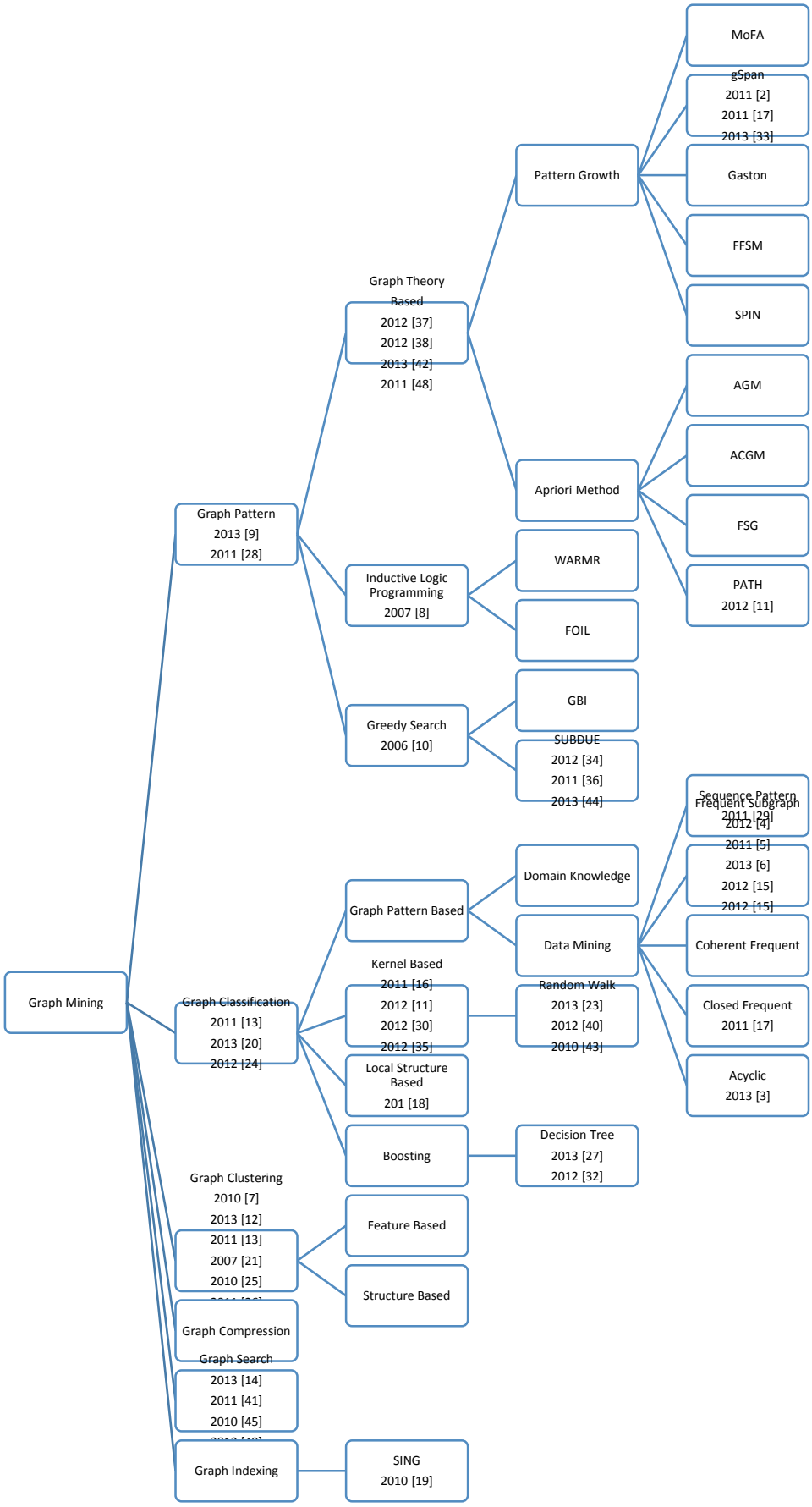
استخراج زیر گراف پرتکرار	Frequent Subgraph Mining
استخراج الگوی گراف متناظر [۹۹-۱۰۱]	Correlated Graph Pattern Mining
استخراج الگوی گراف مطلوب [۶۵, ۱۰۲]	Optimal Graph Pattern Mining
استخراج الگوی گراف تقریبی [۱۰۳, ۱۰۴]	Approximate Graph Pattern Mining
خلاصه سازی و جمع بندی الگوی گراف [۱۰۵, ۱۰۶]	Graph Pattern Summarization
طبقه بندی گراف [۶۲, ۷۰, ۱۰۷]	Graph Classification
خوشه بندی گراف [۲۶, ۱۰۸, ۱۰۹]	Graph Clustering
شاخص گذاری گراف [۴۳]	Graph Indexing
جستجوی گراف [۴۴, ۱۱۰, ۱۱۱]	Graph Searching
هسته اصلی گراف [۱۱۲-۱۱۴]	Graph Kernels

استخراج لینک [۱۱۵]	Link Mining
استخراج ساختار وب	Web Structure Mining
استخراج جریان کار [۱۱۶]	Work-Flow Mining
استخراج شبکه زیستی [۱۱۷]	Biological Network Mining

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 آن ها بالاتر از آستانه تعیین شده است.

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 این که آیا زیر گراف های داوطلب مشخص شده به قدر کافی در داده های گراف پرتکرار هستند که بتوان آن ها 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. 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.


Graph Mining
Graph Pattern
2013 [9]
2011 [28]
Graph Classification
2011 [13]
2013 [20]
2012 [24]
Graph Clustering
2010 [7]
2013 [12]
Graph Compression
Graph Search
2013 [14]
2011 [41]
2010 [45]
Graph Indexing
Graph Theory Based
2012 [37]
2012 [38]
2013 [42]
2011 [48]
Inductive Logic Programming
2007 [8]
Greedy Search
2006 [10]
Graph Pattern Based
Kernel Based
2011 [16]
2012 [11]
2012 [30]
2012 [35]
Local Structure Based
201 [18]
Boosting
2011 [13]
2007 [21]
2010 [25]
Feature Based
Structure Based
SING
2010 [19]
Pattern Growth
Apriori Method
WARMR
FOIL
GBI
SUBDUE
2012 [34]
2011 [36]
2013 [44]
Domain Knowledge
Data Mining
Random Walk
2013 [23]
2012 [40]
2010 [43]
Decision Tree
2013 [27]
2012 [32]
MoFA
gSpan
2011 [2]
2011 [17]
2013 [33]
Gaston
FFSM
SPIN
AGM
ACGM
FSG
PATH
2012 [11]
Sequence Pattern
2011 [29]
Frequent Subgraph
2012 [4]
2011 [5]
2013 [6]
2012 [15]
2012 [15]
Coherent Frequent
Closed Frequent
2011 [17]
Acyclic
2013 [3]

شکل ۷: درختواره مقالات

۳,۲. 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 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 حالی که 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 آن ها وجود دارد تعریف می شود. هر شمارش فارغ از اینکه g یک بار یا بیشتر از یک بار 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:

معادله ۳

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

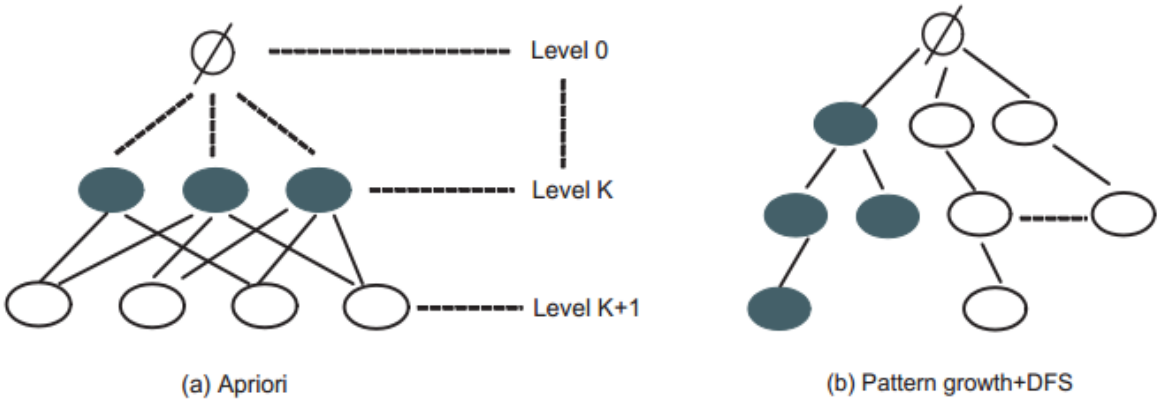
که $|\sigma g(g)|$ بیانگر مقدار $\sigma g(g)$ و T بیانگر تعداد گراف های موجود در G است. بر این اساس، g پرتکرار خواهد بود اگر و تنها اگر $SUPg(g) \geq \sigma$. در شمارش پیشامد ـ محور به راحتی تعداد پیشامد g در مجموعه ورودی ها را محاسبه می کنیم.

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 [۶۳, ۱۱۸] که ممکن است برای FSM بر مبنای گراف مجزا اعمال شود.

۳,۳. ارزیابی FSM

این بخش یک ارزیابی کلی از فرآیند *FSM* (استخراج زیر گراف های پرتکرار) ارائه می دهد. این نکته به طور گسترده ای پذیرفته شده است که تکنیک های *FMS* را می توان 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 و الگوریتم تعمیم *FP* (الگوی پرتکرار) شباهت دارد. 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. تفکیک بین دو رویکرد در تصویر ۴ نشان داده شده است.



شکل ۸: 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* 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 می توان 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 آن جایی 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 بین استخراج زیر گراف پرتکرار و استخراج گراف درختی فرعی پرتکرار تمایز قائل می شود. در ادامه، به استفاده از واژه اختصاری *FSM* برای هر دو حوزه استخراج زیرگراف 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 زیرگراف و گراف درختی فرعی، 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.

۳,۳,۱. 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 برای ارزیابی هم ریختی استفاده نمی شود، زیرا بسته به اینکه رأس ها (و کران ها) چگونه مشخص شوند می توان گراف ها را به شیوه های مختلفی ارائه کرد. با توجه به آزمایش هم ریختی، اتخاذ 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).

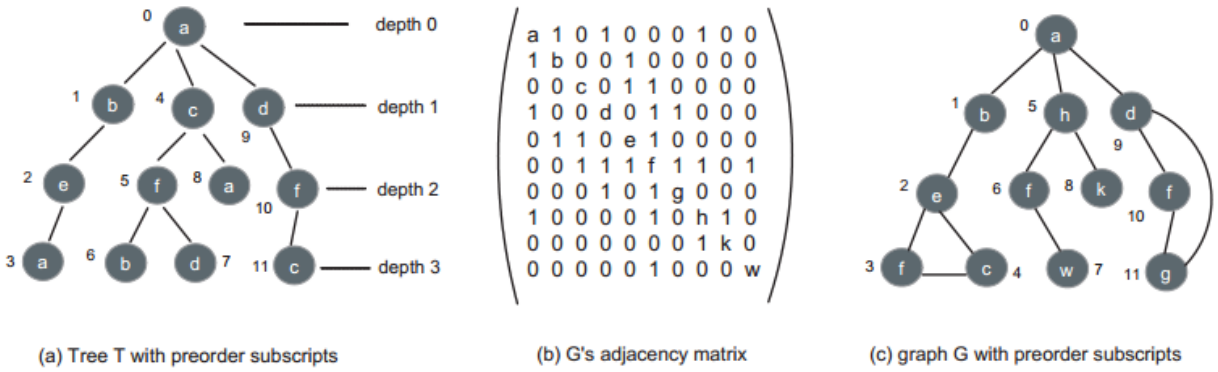
استراتژی برچسب گذاری مجاز یک کد منحصر بفرد برای هر گراف تعیین می کند. برچسب گذاری مجاز موجب سهولت بررسی هم ریختی می شود زیرا تضمین می کند که اگر یک جفت گراف هم ریخت باشند آنگاه بر چسب گذاری مجاز آن ها نیز یکسان باشد. روش ساده برای ایجاد برچسب مجاز، باز کردن ماتریکس تجانب مربوطه با ردیف ها یا ستون های به هم پیوسته برای تولید یک کد متشکل از فهرستی از اعداد صحیح با ترتیب واژگان نمایی مینیمم (یا ماکزیمم) است. برای کاهش بیشتر محاسبات حاصل از جایگشت های ماتریکس، بر چسب گذاری مجاز معمولاً با استفاده از طرح نامتغیر رأس فشرده می شود 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 صورت یک برچسب مجاز g تعریف کرد [۱۷]. کدهای *DFS* برای دست چپی ترین شاخه و دست راستی ترین شاخه گراف ارائه شده در شکل ۹ (c) به ترتیب $\{(0, 1, a, 1, b), (1, 2, b, 1, e), (2, 3, e, 1, f), (3, 4, f, 1, c), (4, 2, c, 1, e)\}$ و $\{(0, 9, a, 1, d), (9, 10, d, 1, f), (10, 11, f, 1, g), (11, 9, g, 1, d)\}$ 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 آن جایی 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 مجاز *CAM* را تعیین می کند [۳, ۱۳, ۵۹]. کد گذاری گراف که در شکل ۹ (c) داده شده که توسط ماتریکس گراف شکل ۹ (*b*) بازنمایی می شود، به این صورت است:

$$\{a1b00c100d0110e00111f\ 000101g1000010h00000001k000001000w\}$$



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

شکل ۹

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.

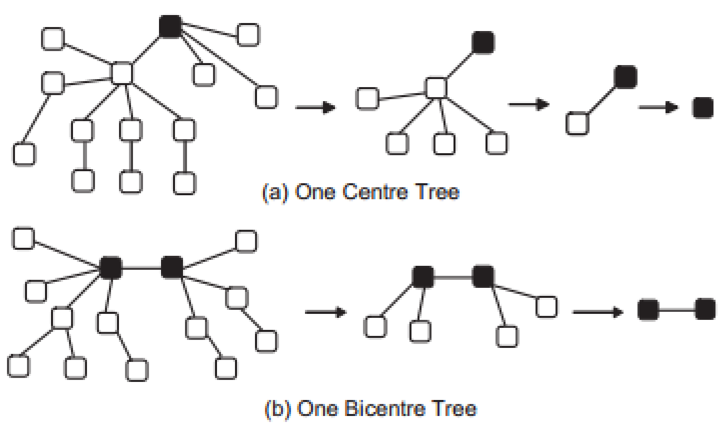
توالی برچسب *DFS* (*DFS – LS*): با در نظر گرفتن یک گراف درختی منظم برچسب دار T، بر چسب های $\forall V_i \in V$ در خلال ایجاد برش *DFS* در T به زنجیره S اضافه می شوند. هرگاه فرآیند عقب نشینی اتفاق بیفتد یک سمبل منحصر بفرد مانند "۱ـ" یا "S" یا "/" به زنجیره S اضافه می شود [۱۱۹-۱۲۱]. کد *DFS-LS* برای گراف درختی داده شده در شکل ۹ (*a*) 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 $\{(V_k))$

L، *d* (*V_k*) و... و((*V_1*) *L* و (*V_1*) *d*)}= *S* تعریف می شود [۷, ۱۲۲]. کد *DLS* برای گراف درختی ارائه شده در شکل ۹ (*a*) به این صورت است: { *(0, a), (1, b), (2, e), (3, a), (1, c), (2, f), (3, b), (3, d), (2, a), (1, d), (2, f), (3, c)*}

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 بر این 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 آن ها 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 مینیمیم این کد گذاری هاست و گراف درختی منظم ریشه ای متناظر می تواند فرم مجاز گستره ـ محور *T* را تعیین کند [۱۲۳]. بنابراین، کد گذاری زنجیره *BFS* از گراف درختی نمونه در شکل ۹ (*a*) 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 [۱۲۳]. کد گذاری زنجیره ای *DFS* گراف درختی نمونه در شکل ۹ (*a*) به صورت زیر است: *abea$$$cfb$d$$dfc$$$#* .



شکل ۱۰: 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 باقی مانده به عنوان مرکز خوانده می شود، و گراف درختی نامنظم ریشه ای با مرکز به عنوان ریشه به دست می آید. پروسه در شکل ۱۰ (*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 شکل ۱۰ (*b*) 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 [۱۲۴]. 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.

۳,۳,۲. ایجاد داوطلب

به گونه ای که پیشتر گفته شد، ایجاد داوطلب مرحله ای لازم و اساسی در FSM است. چگونگی ایجاد قاعده مند زیرگراف های داوطلب بدون زوائد (یعنی هر زیر گراف فقط یک بار 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 آن جایی 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 طور یکسانی برای FTM نیز قابل اجرا هستند و برعکس.

۳,۳,۲,۱. اتصال سطح ـ محور

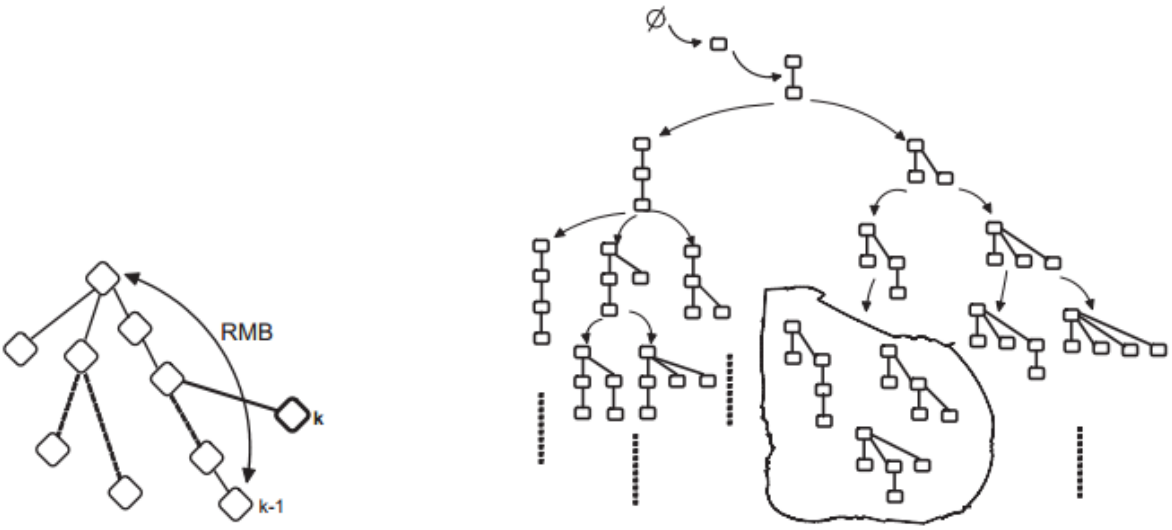
در استراتژی اتصال سطح ـ محور، اساساً داوطلب زیر گراف (K+۱) از طریق ترکیب دو زیر گراف پرتکرار K که زیر گراف (K-۱) کسانی را به اشتراک گذاشته اند ایجاد می شود. این زیر گراف (K-۱) متعارف به عنوان مرکزی برای این دو زیر گراف پرتکرار K تلقی می شود. موضوع اصلی در مورد این استراتژی این است که یک زیر گراف K می تواند حداکثر K زیر گراف (K-۱) مختلف داشته باشد و عملیات اتصال ممکن است تعداد زیادی داوطلب زائد ایجاد کند. در [۱۲۵]، 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، عبارتند از: DPMine [۱۲۶]، و HSIGRAM [۶۳]، که در ادامه مورد بررسی قرار خواهند گرفت.

۳,۳,۲,۲. 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. در شکل ۱۱ (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 ـ ۱ 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. شکل ۱۱ (b) قسمتی از DAG شمارشگر که با استفاده از تعمیم دست راستی ترین مسیر گسترش یافته است را نشان می دهد. هر مربع در شکل ۱۱ (b) نشان دهنده 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 و به دنبال آن به شیوه های مختلفی در اکثر الگوریتم های استخراج گراف های درختی فرعی به کار گرفته شده اند.



(a) The rightmost path (b) A partial enumeration DAG for unlabelled trees

شکل ۱۱: مثالی از تعمیم دست راستی ترین مسیر

۳،۳،۲،۳. امتداد و اتصال

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

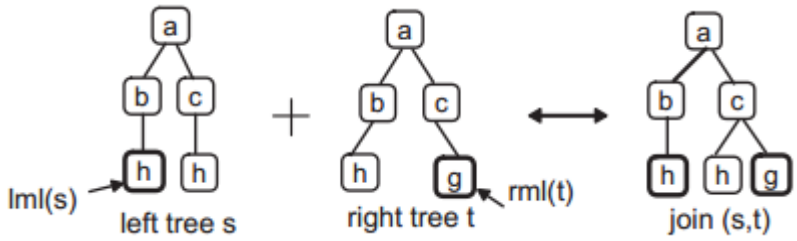
امتداد طبقه ـ محور هم ارز [۱۱۹, ۱۲۰] بر اساس بازنمایی DFS-LS برای گراف های درختی طراحی شد. اساساً، گراف درختی فرعی ـ k+۱ با اتصال دو گراف درختی فرعی ـ k پرتکرار ایجاد می شود. این دو گراف درختی فرعی ـ k باید در طبقه هم ارزی مشابه (C^3) باشند. تمام طبقه های هم ارز شامل کد گذاری پیشوند طبقه و فهرستی از اعضا هستند. هر یک از اعضاء طبقه را می

۳

توان به صورت جفت (L,p) بیان کرده که L برچسب رأس k ام است و P موقعیت عمقی والدین رأس k ام است. 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 [۱۱۹].

۳,۳,۲,۵. اتصال گراف درختی راست ـ و ـ چپ

استراتژی اتصال گراف درختی راست و چپ در [۱۲۷] معرفی شد. این استراتژی اساساً از دست راستی ترین برگ 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 آن ها به این صورت تعریف می شود: Join (s.t) = s □ rmL (t) = LmL (s) □t . شکلی که این عملیات اتصال را توضیح می دهد در شکل ۱۲ داده شده است.



شکل ۱۲: 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.

۳,۴. الگوریتم های استخراج درخت های 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 [۱۲۳]، استخراج کاربرد شبکه [۱۱۹]، نسخه کامپیوتر، استخراج XML [۳۱]، بیوانفوماتیک 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. 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.

جدول ۳

	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			⋆		⋆	

۳,۴,۱. 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.

گراف درختی فرعی نامنظم FTM قصد دارد برای بازنمایی گراف های درختی 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 [۱۲۰]. 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. دو مکانیسم تعمیم برای ایجاد داوطلب در نظر گرفته شد: (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 مشاهده شده، میان استفاده از دو مکانیسم تعادل و موازنه وجود دارد. آزمایش های انجام شده 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. 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.

الگوریتم Rooted Tree Miner 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 [۱۲۸]. 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 در [۱۲۹] یک الگوریتم FTM گراف درختی نامنظم، Cousin Pair، برای کاربرد در تکامل نژادی ارائه کردند. آن ها 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 این گونه 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.

۳,۴,۲. 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، از تعمیم دست راستی ترین مسیر با در نظر گرفتن الگوریتم FREQT بهره می گیرند. به علاوه، فقط پیشامدهای دست راستی ترین برگ الگوها ذخیره می شوند تا محاسبه پشتیبانی کارایی بیشتری پیدا کند. داده های نیمه ـ ساختاری (به طور مثال صفحات وب) با استفاده از گراف درختی منظم برچسب دار برای ارزیابی FREQT طراحی شدند.

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. 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 [۱۲۷]. به دنبال آن، الگوریتم AMIOT برای استفاده از یک طرح شمارش جدید به منظور کاهش تعداد داوطلب های تصادفی در عین حفظ مزیت های استراتژی تعمیم دست راستی معرفی گردید. 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.

یک الگوریتم FTM به نام TreeMiner معرفی شد 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.

الگوریتمی به نام Chopper برای استخراج گراف های درختی فرعی تثبیت شده پرتکرار از مجموعه داده های درختی پیشنهاد گردید [۱۲۲]، که از بازنمایی DLS استفاده شد. الگوریتم Chopper ابتدا از Poefix Span اصلاح شده 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 [۹] نیز استخراج گراف های درختی فرعی تثبیت شده پرتکرار (از پایگاه داده های گراف درختی نامنظم) را هدف گرفته است، اما از پارامتری استفاده می کند 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.

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 معرفی گردیدند [۱۳۰]. 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 بهتر 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.

۳,۴,۳. 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 [۱۳۱] 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. 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 یک بازنمایی مجاز برای گراف های درختی فرعی بر چسب دار تعریف می کرد که در یک 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 گردآوری 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.

با تمرکز عمده بر کاهش هزینه ایجاد داوطلب، الگوریتم استخراج گراف درختی آزاد به نام F3TM معرفی شد [۱۲۴]. 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.

۳,۴,۴. 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، آن ها 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.

Hybrid TreeMiner [۶] 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 FreeTreeMiner 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 [۱۳۲]. 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 HybridTreeMiner عمل می کند.

۳,۴,۵. 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، HybridTreeMiner

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 HybridTreeMiner
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 : 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

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

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.

۳,۵. الگوریتم های استخراج گراف های فرعی پرتکرار

الگوریتم های FGM دارای کاربردهای 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 آن جایی 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 اخیرتر در زمینه استخراج الگوهای پرتکرار شامل: مجموعه آیتم ها، توالی ها، و زیر گراف ها در [۱۳۳] موجود است.

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). در نتیجه، دانستن ماهیت این الگوهای ویژه می تواند کاهش فضای جستجو را امکان پذیر کند.

۳,۵,۱. استخراج زیر گراف های پرتکرار "هدف عمومی "

در این زیر بخش، تعدادی از الگوریتم های FGM هدف عمومی مورد بررسی قرار می گیرند. 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 (BFS یا DFS)

۳,۵,۱,۱. 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. 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 [۱۳۴]. 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 را 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 [۱۰۴] و RAM [۱۳۵]. 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 می دهد 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.

۳,۵,۱,۲. 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 کند. 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. 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 حالی که 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.

الگوریتم های FGM بر مبنای BFS نیز مانند الگوریتم های استخراج قوانین وابسته از قبیل آپریوری، 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 آن را 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 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. 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.
- FSG 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، 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 [۱۲۶] 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. 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 [۱۸] 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 [۱۷] 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 [۱۳۶] 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 [۳] 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 [۱۹]. این الگوریتم با منشعب کردن فرآیند استخراج زیر گراف های پرتکرار به استخراج مسیر، سپس استخراج گراف درختی فرعی و سرانجام استخراج زیر گراف توانست راه حلی برای این مسئله ارائه دهد. 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 و ضعف الگوریتم های مختلف دشوار است. با این وجود در [۱۳۷] مقایسه ای مشروح از چهار استخراج کننده DFS ـ محور: MoFa، gSpan، FFam و GASTON با توجه به عملکرد آن ها در مجموعه داده های شیمیایی گوناگون ارائه شده است. در آزمایش ها، به این نکته پی برده شد که استفاده از لیست های تثبیت، در ازای اشغال حافظه بیشتر، دستاورد چشمگیری ارائه نمی دهند. همچنین تأیید شده است 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"، عملکرد gSpan در زمینه استخراج پایگاه داده های مولکولی بهبود بخشیده شد.

این زیر بخش را با بررسی الگوریتم های FGM دقیق بر مبنای گراف مجزا که در آن ها 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 [۶۳]. در [۱۱۸]، تعریف رسمی به همراه برهان هایی برای شرایط کافی و ضروری برای مقیاس های پشتیبان پیشامد ـ محور در جهت تداوم DCP ارائه شد. این کار 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 [۱۳۸].

دو الگوریتم HSIGRAM و VSIGRAM را برای یافتن زیر گراف های پرتکرار در گراف های غیر متراکم بزرگ ارائه شد. 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 [۱۱۸]. 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 SUBDUE، 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. این کار 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.

۳،۵،۲. استخراج زیر گراف های پرتکرار 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 آن ها 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.

۳,۵,۲,۱. استخراج الگوی ارتباطی

گراف های ارتباطی برای طراحی شبکه های گسترده ای مانند شبکه های زیستی و اجتماعی مناسب هستند. 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 مینیمم یک الگو). بنابراین، هدف استخراج گراف ارتباطی تعیین تمام الگوهای پرتکرار نمایشگر محدودیت اتصال مشخص شده است [۳۱].

CLOSECUT و SPLAT، 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.

۳,۵,۲,۲. استخراج الگوهای بیشینه و مسدود

تعداد زیرگراف های پرتکرار موجود به طور بالقوه با اندازه گراف افزایش می یابد، یعنی برای K گراف پرتکرار تعداد زیرگراف های پرتکرار آن می تواند به بزرگی ۲^k باشد. در حدود ۱/۰۰۰/۰۰۰ الگوی گراف پرتکرار از ۴۲۲ ترکیب شیمیایی تولید شده است (با استفاده از آستانه پشتیبان ٪۵)؛ که تعداد زیادی از این ها 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: FS ⊆ CFS ⊆MFS.

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، آن ها به عنوان جالب ترین الگوها پنداشته می شوند [۱۳۹]. 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 [۶۲] 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 [۴] 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 می یابد. 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. CLOSECUT و SPLAT دو نمونه از الگوریتم های FGM مسدود هستند. نمونه دیگر نیز CloseGraph 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 زود هنگام شکست می خورد و اجرای آن عملی نمی شود، ارزیابی ناکامی حذف زود هنگام به اجرا در می آید. نتایج تجربی نشان داد که CloseGraph بهتر از gSpan و FSG عمل می کند.

۳,۵,۲,۳. استخراج دسته ها

یک دسته (یا به ظاهر دسته) زیر مجموعه ای از یک زیر گراف با مکان ثابت است. تعداد زیادی الگوریتم طراحی شدند که انواعی از مسائل جستجو و بررسی دسته ها را نشانه گرفته بودند. 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، استخراج بیان ژن، و شناسایی سهام بسیار متناظر از گراف های بازار اوراق بهادار [۱۴۰]. از الگوریتم های FGM هدف عمومی می توان برای شناسایی این گونه "الگوهای خاص" بهره گرفت، هر چند محاسبات کارآمد تر خواهند بود اگر ویژگی های خاص دسته ها نیز در نظر گرفته شوند. دو نمونه از الگوریتم های استخراج دسته، CLAN و Cocain، در پاراگراف های بعد مورد بررسی قرار خواهند گرفت.

CLAN [۱۴۰] 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. این الگوریتم 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.

با تعمیم و گسترش کران CLAN یک شکل کلی و عمومی از الگوریتم استخراج دسته، به نام Cocain معرفی شد [۱۴۱]، تا γ دسته یا 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.

۳,۵,۲,۴. 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. چارچوبی به نام gPrune ارائه شد تا محدودیت های مختلف را با فرآیند استخراج زیر گراف های پرتکرار درهم آمیزد [۱۴۲]. در 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 بر این، 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.

۳,۵,۳. خلاصه

جدول ۵، خلاصه ای از رویکردهای بازنمایی مجاز، ایجاد داوطلب و محاسبه پشتیبان که در الگوریتم FGM به کار گرفته می شوند 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 جالب 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 حالی که gspan، با چند تغییر اندک، می تواند با گراف های مستقیم سازگار شود.

جدول ۵: خلاصه ای از الگوریتم های *FGM* متداول و مکانیسم های ایجاد داوطلب و محاسبه پشتیبان آن ها

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 فعالیت، آمیخته
DPmine	n/a	اتصال سطح ـ محور	n/a
MoFa	n/a	تعمیم	لیست جاسازی ها
gSpan	m-DFSC	تعمیم دست راستی ترین	لیست فعالیت
ADI-mine	M-DFSC	تعمیم دست راستی ترین	لیست فعالیت
FFSM	CAM	اتصال + تعمیم	لیست جاسازی ها
GASTON	n/a	مسیر گراف درختی و شمارش گراف	لیست جاسازی ها
HSIGRAM	CAM	اتصال سطح ـ محور	مقیاس های MIS مختلف
VSIGRAM	CAM	تعمیم	مقیاس های MIS مختلف
FPF	CAM	تعمیم	مقیاس های MIS
DPMine	n/a	اتصال سطح ـ محور	n/a
CLOSECUT	M-DFSC	تعمیم دست راستی ترین مسیر	لیست فعالیت
SPLAT	n/a	n/a	n/a
SPIN	n/a	اتصال	مجموعه جاسازی ها
MARGIN	n/a	Expandcat	n/a
CLoseGraph	m-DFSC	تعمیم دست راستی ترین مسیر	لیست فعالیت
CLAN	توالی برچسب رأس	تعمیم DFS ـ محور	n/a
COCain	توالی برچسب رأس	تعمیم DFS ـ محور	n/a
Gprune	M-DFSC	تعمیم دست راستی ترین مسیر	لیست فعالیت

یکی از ویژگی های مشترک اکثر الگوریتم های موجود در جدول ۵ 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 مشبک 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 مادر ـ

زاده برقرار است). بنابراین، 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 (۱ـK) به دست آورد. اطلاعات این محدوده بالایی را می توان 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 ۱ـK استوار است.

یک الگوریتم FGM کارآمد معمولاً دارای سه ویژگی 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 کار گرفته می شود و تعمیم دست راستی ترین مسیر که توسط *MoFa* مورد استفاده قرار می گیرد.
- **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 زیرگراف 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 گراف هستند.

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 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.

۳,۶. 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؟

فصل چهارم: جمع بندی و پیشنهادات

۴,۱. مقدمه

فصل یک

نتیجه گیری

این بخش مقدمه ای مختصر در مورد مسئله استخراج گراف ارائه داده است که به گراف های حلقه ای که تمرکز اصلی کتاب بر آنهاست، محدود نمی شود. برخی تعاریف رسمی با مثال های تصویری ارائه شده تا به خواننده امکان درک پیچیدگی و دشواری مسئله و بعضی از ویژگی های نامطلوب که عامل این دشواری هستند، داده شود. این دشواری و پیچیدگی دلیل اصلی طراحی تعداد زیادی رویکردهای مختلف برای حل مسئله استخراج گراف را شرح می دهد (که به این نکته در بخش ۴ اشاره شده است). می توانیم مشاهده کنیم که تجربه های تازه اغلب زمانی به وجود می آیند که مسئله تا حدی تخفیف پیدا می کند. به طور مثال در مورد الگوریتم GASTON (نیحسن و کوک ۲۰۰۴) که با شناسایی الگوهای ساده تر در ابتدای فرایند و افزایش پیچیدگی با حرکت از توالی ها، درخت ها و در نهایت گراف های حلقه ای به یک شروع سریع و چشم گیر دست می یابد. با این وجود، لازم به ذکر است علیرغم اینکه مسئله استخراج گراف از نظر تئوری علمی نیست ولی در شرایط علمی الگوریتم های موجود می توانند این عملیات را در چارچوب زمانی معقول برای داده های دارای ساختار گراف به اتمام برسانند. کاربرد های استخراج گراف گوناگون و متنوع است و به طور کلی در امر تحلیل داده ها در هر حوزه ای مفید و سودمند هستند با فرض اینکه داده ها در یک ساختار گراف سازماندهی شده باشند. وقتی الگوریتم ها را مورد بحث و بررسی قرار می دادیم به برخی از حوزه های کاربردی اشاره کردیم و می توانیم به طور کلی ادعا کنیم که برنامه های کاربردی تا حدودی به بسیاری از نرم افزارهای گراف های درختی که در فصل ۹ بررسی شد، شباهت دارند با این تفاوت عمده که در استخراج گراف، داده هایی کاربردی شامل حلقه هایی در میان موضوعات داده هاست. از دیگر نمونه های برنامه های کاربردی می توان به تحلیل ترکیب های شیمیایی، تحلیل شبکه های اجتماعی، استخراج ساختار وب، استخراج مباحث مربوط به هستی شناسی، و به طور کلی استخراج تکنیک های سودمند و موثر برای بسیاری از نرم افزار های وب و رسانه های اجتماعی اشاره کرد.

فهرست مراجع

1. Chakrabarti, D. and C. Faloutsos, *Graph mining: Laws, generators, and algorithms.* ACM Comput. Surv., 2006. **38**(1): p. 2.

2. Washio, T. and H. Motoda, *State of the art of graph-based data mining.* SIGKDD Explor. Newsl., 2003. **5**(1): p. 59-68.

3. Huan, J., W. Wang, and J. Prins, *Efficient Mining of Frequent Subgraphs in the Presence of Isomorphism*, in *Proceedings of the Third IEEE International Conference on Data Mining*. 2003, IEEE Computer Society. p. 549.

4. Thomas, L.T., S.R. Valluri, and K. Karlapalem, *MARGIN: Maximal frequent subgraph mining.* ACM Trans. Knowl. Discov. Data, 2010. **4**(3): p. 1-42.

5. Chi, Y., et al., *Frequent Subtree Mining - An Overview.* Fundam. Inf., 2005. **66**(1-2): p. 161-198.

6. Yun, C., Y. Yirong, and R.R. Muntz. *HybridTreeMiner: an efficient algorithm for mining frequent rooted trees and free trees using canonical forms*. in *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*. 2004.

7. Asai, T., et al. *Efficient substructure discovery from large semi-structured data*.

8. Asai, T., et al. *Discovering Frequent Substructures In Large Unordered Trees*. in *IN PROC. OF THE 6TH INTL. CONF. ON DISCOVERY SCIENCE*. Springer-Verlag.

9. Tan, H., et al. *IMB3 Miner: Mining Induced/Embedded Subtrees by Constraining the Level of Embedding*. in *In Proc. of PAKDD'06*.

10. Cordella, L.P., et al. *An improved algorithm for matching large graphs*. in *In: 3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition, Cuen*.

11. CONTE, D., et al., *THIRTY YEARS OF GRAPH MATCHING IN PATTERN RECOGNITION.* International Journal of Pattern Recognition and Artificial Intelligence, 2004. **18**(03): p. 265-298.

12. Foggia, P., C. Sansone, and M. Vento. *A performance comparison of five algorithms for graph isomorphism*. in *in Proceedings of the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition*.

13. Inokuchi, A., T. Washio, and H. Motoda, *An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data*, in *Principles of Data Mining and Knowledge Discovery*, D. Zighed, J. Komorowski, and J. Żytkow, Editors. 2000, Springer Berlin Heidelberg. p. 13-23.

14. Kuramochi, M. and G. Karypis. *Frequent Subgraph Discovery*.

15. Kuramochi, M. and G. Karypis, *Discovering frequent geometric subgraphs.* Inf. Syst., 2007. **32**(8): p. 1101-1120.

16. Vanetik, N., E. Gudes, and S.E. Shimony, *Computing Frequent Graph Patterns from Semistructured Data*, in *Proceedings of the 2002 IEEE International Conference on Data Mining*. 2002, IEEE Computer Society. p. 458.

17. Yan, X. and J. Han, *gSpan: Graph-Based Substructure Pattern Mining*, in *Proceedings of the 2002 IEEE International Conference on Data Mining*. 2002, IEEE Computer Society. p. 721.

18. Borgelt, C. and M.R. Berthold, *Mining Molecular Fragments: Finding Relevant Substructures of Molecules*, in *Proceedings of the 2002 IEEE International Conference on Data Mining*. 2002, IEEE Computer Society. p. 51.

19. Nijssen, S. and J.N. Kok, *A quickstart in frequent structure mining can make a difference*, in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2004, ACM: Seattle, WA, USA. p. 647-652.

20. Raedt, L.D. and S. Kramer, *The levelwise version space algorithm and its application to molecular fragment finding*, in *Proceedings of the 17th international joint conference on Artificial intelligence - Volume 2*. 2001, Morgan Kaufmann Publishers Inc.: Seattle, WA, USA. p. 853-859.

21. Lisi, F. and D. Malerba, *Inducing Multi-Level Association Rules from Multiple Relations.* Machine Learning, 2004. **55**(2): p. 175-210.

22. Ketkar, N.S., L.B. Holder, and D.J. Cook, *Subdue: compression-based frequent pattern discovery in graph data*, in *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*. 2005, ACM: Chicago, Illinois. p. 71-76.

23. Holder, L., et al., *Structural Pattern Recognition in Graphs*, in *Pattern Recognition and String Matching*, D. Chen and X. Cheng, Editors. 2002, Springer US. p. 255-279.

24. Noble, C.C. and D.J. Cook, *Graph-based anomaly detection*, in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2003, ACM: Washington, D.C. p. 631-636.

25. Cook, D.J., et al., *Approaches to parallel graph-based knowledge discovery.* J. Parallel Distrib. Comput., 2001. **61**(3): p. 427-446.

26. Flake, G.W., R.E. Tarjan, and K. Tsioutsiouliklis, *Graph Clustering and Minimum Cut Trees.* 2003: p. 385-408.

27. Shijie, Z., Y. Jiong, and V. Cheedella. *Monkey: Approximate Graph Mining Based on Spanning Trees*. in *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*. 2007.

28. Wang, W., et al., *GraphMiner: a structural pattern-mining system for large disk-based graph databases and its applications*, in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. 2005, ACM: Baltimore, Maryland. p. 879-881.

29. Saigo, H. and K. Tsuda. *Iterative Subgraph Mining for Principal Component Analysis*. in *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*. 2008.

30. Yan, X. and J. Han, *CloseGraph: mining closed frequent graph patterns*, in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2003, ACM: Washington, D.C. p. 286-295.

31. Yan, X., X.J. Zhou, and J. Han, *Mining closed relational graphs with connectivity constraints*, in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 2005, ACM: Chicago, Illinois, USA. p. 324-333.

32. Aggarwal, C.C., et al., *Xproj: a framework for projected structural clustering of xml documents*, in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2007, ACM: San Jose, California, USA. p. 46-55.

33. Jagadish, H.V., et al., *TIMBER: A native XML database.* The VLDB Journal, 2002. **11**(4): p. 274-291.

34. Jagadish, H.V., et al., *TAX: A Tree Algebra for XML*, in *Database Programming Languages*, G. Ghelli and G. Grahne, Editors. 2002, Springer Berlin Heidelberg. p. 149-164.

35. Kaushik, R., et al., *Covering indexes for branching path queries*, in *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. 2002, ACM: Madison, Wisconsin. p. 133-144.

36. Min, J.-K., C.-W. Chung, and K. Shim, *An adaptive path index for XML data using the query workload.* Inf. Syst., 2005. **30**(6): p. 467-487.

37. Cooper, B., et al., *A Fast Index for Semistructured Data*, in *Proceedings of the 27th International Conference on Very Large Data Bases*. 2001, Morgan Kaufmann Publishers Inc. p. 341-350.

38. Li, Q. and B. Moon, *Indexing and Querying XML Data for Regular Path Expressions*, in *Proceedings of the 27th International Conference on Very Large Data Bases*. 2001, Morgan Kaufmann Publishers Inc. p. 361-370.

39. Haixun, W. and M. Xiaofeng. *On the sequencing of tree structures for XML indexing*. in *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*. 2005.

40. Rao, P. and B. Moon. *PRIX: indexing and querying XML using prufer sequences*. in *Data Engineering, 2004. Proceedings. 20th International Conference on*. 2004.

41. He, H. and A. Singh, *Query Language and Access Methods for Graph Databases*, in *Managing and Mining Graph Data*, C.C. Aggarwal and H. Wang, Editors. 2010, Springer US. p. 125-160.

42. Wang, H. and C. Aggarwal, *A Survey of Algorithms for Keyword Search on Graph Data*, in *Managing and Mining Graph Data*, C.C. Aggarwal and H. Wang, Editors. 2010, Springer US. p. 249-273.

43. Yan, X., P.S. Yu, and J. Han, *Graph indexing: a frequent structure-based approach*, in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. 2004, ACM: Paris, France. p. 335-346.

44. Yan, X., P.S. Yu, and J. Han, *Substructure similarity search in graph databases*, in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. 2005, ACM: Baltimore, Maryland. p. 766-777.

45. Chen, L., A. Gupta, and M.E. Kurul, *Stack-based algorithms for pattern matching on DAGs*, in *Proceedings of the 31st international conference on Very large data bases*. 2005, VLDB Endowment: Trondheim, Norway. p. 493-504.

46. Tri, S., #223, and U. Leser, *Fast and practical indexing and querying of very large graphs*, in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 2007, ACM: Beijing, China. p. 845-856.

47. Haixun, W., et al. *Dual Labeling: Answering Graph Reachability Queries in Constant Time*. in *Data Engineering, 2006. ICDE '06. Proceedings of the 22nd International Conference on*. 2006.

48. Cheng, J., et al. *Fast computation of reachability labeling for large graphs*. in *In Proc. of EDBT'06*.

49. Cheng, J., et al., *Fast computing reachability labelings for large graphs with high compression rate*, in *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*. 2008, ACM: Nantes, France. p. 193-204.

50. Jin, R., et al., *Computing label-constraint reachability in graph databases*, in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. 2010, ACM: Indianapolis, Indiana, USA. p. 123-134.

51. Cordella, L.P., et al., *A (sub)graph isomorphism algorithm for matching large graphs.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2004. **26**(10): p. 1367-1372.

52. Neuhaus, M. and H. Bunke, *Self-organizing maps for learning the edit costs in graph matching.* Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 2005. **35**(3): p. 503-514.

53. Neuhaus, M. and H. Bunke, *Automatic learning of cost functions for graph edit distance*. Information Sciences, 2007. **177**(1): p. 239-247.

54. Hristidis, V. and Y. Papakonstantinou, *Discover: keyword search in relational databases*, in *Proceedings of the 28th international conference on Very Large Data Bases*. 2002, VLDB Endowment: Hong Kong, China. p. 670-681.

55. Bhalotia, G., et al. *Keyword searching and browsing in databases using BANKS*. in *Data Engineering, 2002. Proceedings. 18th International Conference on*. 2002.

56. Kacholia, V., et al., *Bidirectional expansion for keyword search on graph databases*, in *Proceedings of the 31st international conference on Very large data bases*. 2005, VLDB Endowment: Trondheim, Norway. p. 505-516.

57. He, H., et al., *Blinks: Ranked keyword searches on graphs*. 2007.

58. Sarma, A.D., S. Gollapudi, and R. Panigrahy, *Estimating PageRank on graph streams*, in *Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 2008, ACM: Vancouver, Canada. p. 69-78.

59. Kuramochi, M. and G. Karypis, *Frequent Subgraph Discovery*, in *Proceedings of the 2001 IEEE International Conference on Data Mining*. 2001, IEEE Computer Society. p. 313-320.

60. Bringmann, B. and S. Nijssen, *What Is Frequent in a Single Graph?*, in *Advances in Knowledge Discovery and Data Mining*, T. Washio, et al., Editors. 2008, Springer Berlin Heidelberg. p. 858-863.

61. Fiedler, M. and C. Borgelt. *Support Computation for Mining Frequent Subgraphs in a Single Graph*. in *The 5th International Workshop on Mining and Learning with Graphs*. 2007.

62. Huan, J., et al., *SPIN: mining maximal frequent subgraphs from graph databases*, in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2004, ACM: Seattle, WA, USA. p. 581-586.

63. Kuramochi, M. and G. Karypis, *Finding Frequent Patterns in a Large Sparse Graph*.* Data Min. Knowl. Discov., 2005. **11**(3): p. 243-271.

64. and Ambuj K. Singh, S.R. *GraphSig: A Scalable Approach to Mining Significant Subgraphs in<p> Large Graph Databases*. in *Proceedings of the 25th International Conference on Data Engineering,<p> ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*.

65. Fan, W., et al., *Direct mining of discriminative and essential frequent patterns via model-based search tree*, in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2008, ACM: Las Vegas, Nevada, USA. p. 230-238.

66. Chen, C., et al., *Mining graph patterns efficiently via randomized summaries.* Proc. VLDB Endow., 2009. **2**(1): p. 742-753.

67. Zaki, M.J. and C.C. Aggarwal. *XRules: An Effective Structural Classifier for XML Data*.

68. Kudo, T., E. Maeda, and Y. Matsumoto, *An Application of Boosting to Graph Classification*. 2004.

69. Peng, J.-y., et al. *An Efficient Algorithm for Detecting Closed Frequent Subgraphs in Biological Networks*. in *BioMedical Engineering and Informatics, 2008. BMEI 2008. International Conference on*. 2008.

70. Deshpande, M., et al., *Frequent substructure-based approaches for classifying chemical compounds.* Knowledge and Data Engineering, IEEE Transactions on, 2005. **17**(8): p. 1036-1050.

71. Huan, J., et al. *Mining spatial motifs from protein structure graphs*. in *In Proc. of the 8th Annual Int. Conf. on Research in Computational Molecular Biology (RECOMB'04*.

72. Lee, M.L., et al., *XClust: clustering XML schemas for effective integration*, in *Proceedings of the eleventh international conference on Information and knowledge management*. 2002, ACM: McLean, Virginia, USA. p. 292-299.

73. Gibson, D., R. Kumar, and A. Tomkins, *Discovering large dense subgraphs in massive graphs*, in *Proceedings of the 31st international conference on Very large data bases*. 2005, VLDB Endowment: Trondheim, Norway. p. 721-732.

74. Pei, J., D. Jiang, and A. Zhang, *On mining cross-graph quasi-cliques*, in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 2005, ACM: Chicago, Illinois, USA. p. 228-238.

75. Lian, W., et al., *An Efficient and Scalable Algorithm for Clustering XML Documents by Structure.* IEEE Trans. on Knowl. and Data Eng., 2004. **16**(1): p. 82-96.

76. Dalamagas, T., et al., *Clustering XML Documents Using Structural Summaries*, in *Current Trends in Database Technology - EDBT 2004 Workshops*, W. Lindner, et al., Editors. 2005, Springer Berlin Heidelberg. p. 547-556.

77. Zhou, D., et al., *Learning from labeled and unlabeled data on a directed graph*, in *Proceedings of the 22nd international conference on Machine learning*. 2005, ACM: Bonn, Germany. p. 1036-1043.

78. Saigo, H., et al., *gBoost: a mathematical programming approach to graph classification and regression.* Machine Learning, 2009. **75**(1): p. 69-89.

79. Getoor, L., *Link-based Classification*, in *Advanced Methods for Knowledge Discovery from Complex Data*. 2005, Springer London. p. 189-207.

80. Zhang, T., A. Popescul, and B. Dom, *Linear prediction models with graph regularization for web-page categorization*, in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006, ACM: Philadelphia, PA, USA. p. 821-826.

81. Leskovec, J., J. Kleinberg, and C. Faloutsos, *Graphs over time: densification laws, shrinking diameters and possible explanations*, in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 2005, ACM: Chicago, Illinois, USA. p. 177-187.

82. Sun, J., et al., *GraphScope: parameter-free mining of large time-evolving graphs*, in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2007, ACM: San Jose, California, USA. p. 687-696.

83. Asur, S., S. Parthasarathy, and D. Ucar, *An event-based framework for characterizing the evolutionary behavior of interaction graphs.* ACM Trans. Knowl. Discov. Data, 2009. **3**(4): p. 1-36.

84. Leskovec, J., et al. *Cascading behavior in large blog graphs*. in *In SDM*.

85. Fogaras, D., et al., *Towards Scaling Fully Personalized PageRank: Algorithms, Lower Bounds, and Experiments.* 2005: p. 333-358.

86. Chakrabarti, S., *Dynamic personalized pagerank in entity-relation graphs*, in *Proceedings of the 16th international conference on World Wide Web*. 2007, ACM: Banff, Alberta, Canada. p. 571-580.

87. Sarkar, P., A.W. Moore, and A. Prakash, *Fast incremental proximity search in large graphs*, in *Proceedings of the 25th international conference on Machine learning*. 2008, ACM: Helsinki, Finland. p. 896-903.

88. Sarkar, P. and A.W. Moore, *Fast dynamic reranking in large graphs*, in *Proceedings of the 18th international conference on World wide web*. 2009, ACM: Madrid, Spain. p. 31-40.

89. Chakraborty, A. *Top-K aggregation over a large graph using shared-nothing systems*. in *Big Data, 2013 IEEE International Conference on*. 2013.

90. Rattigan, M.J., M. Maier, and D. Jensen, *Graph clustering with network structure indices*, in *Proceedings of the 24th international conference on Machine learning*. 2007, ACM: Corvalis, Oregon. p. 783-790.

91. Satuluri, V. and S. Parthasarathy, *Scalable graph clustering using stochastic flows: applications to community discovery*, in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, ACM: Paris, France. p. 737-746.

92. Bordino, I., et al. *Mining Large Networks with Subgraph Counting*. in *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*. 2008.

93. Aggarwal, C., Y. Xie, and P.S. Yu, *GConnect: a connectivity index for massive disk-resident graphs.* Proc. VLDB Endow., 2009. **2**(1): p. 862-873.

94. Liu, C., et al., *Mining Behavior Graphs for "Backtrace " of Noncrashing Bugs*. 2005.

95. Eichinger, F., et al., *Mining Edge-Weighted Call Graphs to Localise Software Bugs*, in *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*. 2008, Springer-Verlag: Antwerp, Belgium. p. 333-348.

96. Eichinger, F., K. Bohm, and M. Huber, *Improved Software Fault Detection with Graph Mining*, in *Mining and Learning with Graphs*.

97. Fatta, G.D., S. Leue, and E. Stegantova, *Discriminative pattern mining in software fault detection*, in *Proceedings of the 3rd international workshop on Software quality assurance*. 2006, ACM: Portland, Oregon. p. 62-69.

98. Ray-Yaung, C., A. Podgurski, and Y. Jiong, *Discovering Neglected Conditions in Software by Mining Dependence Graphs.* Software Engineering, IEEE Transactions on, 2008. **34**(5): p. 579-596.

99. Ke, Y., J. Cheng, and W. Ng, *Correlation search in graph databases*, in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2007, ACM: San Jose, California, USA. p. 390-399.

100. Yiping, K., J. Cheng, and J.X. Yu. *Efficient Discovery of Frequent Correlated Subgraph Pairs*. in *Data Mining, 2009. ICDM '09. Ninth IEEE International Conference on*. 2009.

101. Ozaki, T. and T. Ohkawa, *Mining Correlated Subgraphs in Graph Databases*, in *Advances in Knowledge Discovery and Data Mining*, T. Washio, et al., Editors. 2008, Springer Berlin Heidelberg. p. 272-283.

102. Yan, X., et al., *Mining significant graph patterns by leap search*, in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 2008, ACM: Vancouver, Canada. p. 433-444.

103. Sharan, R., et al., *Conserved patterns of protein interaction in multiple species.* Proceedings of the National Academy of Sciences of the United States of America, 2005. **102**(6): p. 1974-1979.

104. Chen, C., et al. *gApprox: Mining Frequent Approximate Patterns from a Massive Network*. in *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*. 2007.

105. Xin, D., et al., *Extracting redundancy-aware top-k patterns*, in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006, ACM: Philadelphia, PA, USA. p. 444-453.

106. Chen, C., et al., *On effective presentation of graph patterns: a structural representative approach*, in *Proceedings of the 17th ACM conference on Information and knowledge management*. 2008, ACM: Napa Valley, California, USA. p. 299-308.

107. Kudo, T., E. Maeda, and Y. Matsumoto, *An Application of Boosting to Graph Classification*, in *Neural Information Processing Systems*. 2004.

108. Huang, X. and W. Lai, *Clustering graphs for visualization via node similarities.* J. Vis. Lang. Comput., 2006. **17**(3): p. 225-253.

109. Newman, M.E.J., *Detecting community structure in networks.* The European Physical Journal B - Condensed Matter and Complex Systems, 2004. **38**(2): p. 321-330.

110. Yan, X., et al. *Searching Substructures with Superimposed Distance*. in *Data Engineering, 2006. ICDE '06. Proceedings of the 22nd International Conference on*. 2006.

111. Chen, C., et al., *Towards graph containment search and indexing*, in *Proceedings of the 33rd international conference on Very large data bases*. 2007, VLDB Endowment: Vienna, Austria. p. 926-937.

112. Gärtner, T., P. Flach, and S. Wrobel, *On Graph Kernels: Hardness Results and Efficient Alternatives*, in *Learning Theory and Kernel Machines*, B. Schölkopf and M. Warmuth, Editors. 2003, Springer Berlin Heidelberg. p. 129-143.

113. Borgwardt, K.M. and H.P. Kriegel. *Shortest-path kernels on graphs*. in *Data Mining, Fifth IEEE International Conference on*. 2005.

114. Kashima, H., K. Tsuda, and A. Inokuchi. *Marginalized kernels between labeled graphs*. in *Proceedings of the Twentieth International Conference on Machine Learning*. AAAI Press.

115. Getoor, L. and C.P. Diehl, *Link mining: a survey.* SIGKDD Explor. Newsl., 2005. **7**(2): p. 3-12.

116. Greco, G., et al., *Mining Constrained Graphs: The Case of Workflow Systems*, in *Constraint-Based Mining and Inductive Databases*, J.-F. Boulicaut, L. De Raedt, and H. Mannila, Editors. 2006, Springer Berlin Heidelberg. p. 155-171.

117. Hu, H., et al., *Mining coherent dense subgraphs across massive biological networks for functional discovery.* Bioinformatics, 2005. **21**(1): p. 213-221.

118. Vanetik, N., S.E. Shimony, and E. Gudes, *Support measures for graph data*.* Data Mining and Knowledge Discovery, 2006. **13**(2): p. 243-260.

119. Zaki, M.J., *Efficiently mining frequent trees in a forest*, in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2002, ACM: Edmonton, Alberta, Canada. p. 71-80.

120. Zaki, M.J., *Efficiently Mining Frequent Embedded Unordered Trees.* Fundam. Inf., 2004. **66**(1-2): p. 33-52.

121. Yan, X., et al., *Searching Substructures with Superimposed Distance*.

122. Wang, C., et al., *Efficient Pattern-Growth Methods for Frequent Tree Pattern Mining*, in *Advances in Knowledge Discovery and Data Mining*, H. Dai, R. Srikant, and C. Zhang, Editors. 2004, Springer Berlin Heidelberg. p. 441-451.

123. Chi, Y., Y. Yang, and R.R. Muntz, *Canonical forms for labelled trees and their applications in frequent subtree mining.* Knowl. Inf. Syst., 2005. **8**(2): p. 203-234.

124. Zhao, P. and J. Yu, *Fast Frequent Free Tree Mining in Graph Databases.* World Wide Web, 2008. **11**(1): p. 71-92.

125. Kuramochi, M. and G. Karypis, *An Efficient Algorithm for Discovering Frequent Subgraphs.* IEEE Trans. on Knowl. and Data Eng., 2004. **16**(9): p. 1038-1051.

126. Gudes, E., S.E. Shimony, and N. Vanetik, *Discovering Frequent Graph Patterns Using Disjoint Paths.* Knowledge and Data Engineering, IEEE Transactions on, 2006. **18**(11): p. 1441-1456.

127. Hido, S. and H. Kawano. *AMIOT: induced ordered tree mining in tree-structured databases*. in *Data Mining, Fifth IEEE International Conference on*. 2005.

128. Termier, A., M.-C. Rousset, and M. Sebag. *TreeFinder: a first step towards XML data mining*. in *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*. 2002.

129. Shasha, D., J.T.L. Wang, and Z. Sen. *Unordered tree mining with applications to phylogeny*. in *Data Engineering, 2004. Proceedings. 20th International Conference on*. 2004.

130. Tatikonda, S., S. Parthasarathy, and T. Kurc, *TRIPS and TIDES: new algorithms for tree mining*, in *Proceedings of the 15th ACM international conference on Information and knowledge management*. 2006, ACM: Arlington, Virginia, USA. p. 455-464.

131. Yun, C., Y. Yirong, and R.R. Muntz. *Indexing and mining free trees*. in *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. 2003.

132. Chi, Y., et al., *CMTreeMiner: Mining Both Closed and Maximal Frequent Subtrees*, in *Advances in Knowledge Discovery and Data Mining*, H. Dai, R. Srikant, and C. Zhang, Editors. 2004, Springer Berlin Heidelberg. p. 63-73.

133. Han, J., et al., *Frequent pattern mining: current status and future directions.* Data Min. Knowl. Discov., 2007. **15**(1): p. 55-86.

134. Kuramochi, M. and G. Karypis, *GREW-A Scalable Frequent Subgraph Discovery Algorithm*, in *Proceedings of the Fourth IEEE International Conference on Data Mining*. 2004, IEEE Computer Society. p. 439-442.

135. Zhang, S. and J. Yang, *RAM: Randomized Approximate Graph Mining*, in *Proceedings of the 20th international conference on Scientific and Statistical Database Management*. 2008, Springer-Verlag: Hong Kong, China. p. 187-203.

136. Wang, C., et al., *Scalable mining of large disk-based graph databases*, in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2004, ACM: Seattle, WA, USA. p. 316-325.

137. Wörlein, M., et al., *A Quantitative Comparison of the Subgraph Miners MoFa, gSpan, FFSM, and Gaston*, in *Knowledge Discovery in Databases: PKDD 2005*, A. Jorge, et al., Editors. 2005, Springer Berlin Heidelberg. p. 392-403.

138. Calders, T., J. Ramon, and D. Van Dyck. *Anti-monotonic Overlap-Graph Support Measures*. in *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*. 2008.

139. Koyutürk, M., A. Grama, and W. Szpankowski, *An efficient algorithm for detecting frequent subgraphs in biological networks.* Bioinformatics, 2004. **20**(suppl 1): p. i200-i207.

140. Jianyong, W., Z. Zhiping, and Z. Lizhu. *CLAN: An Algorithm for Mining Closed Cliques from Large Dense Graph Databases*. in *Data Engineering, 2006. ICDE '06. Proceedings of the 22nd International Conference on*. 2006.

141. Zhang, S. and J.T.L. Wang, *Mining Frequent Agreement Subtrees in Phylogenetic Databases*, in *Proceedings of the 2006 SIAM International Conference on Data Mining*. p. 222-233.

142. Zhu, F., et al., *gPrune: A Constraint Pushing Framework for Graph Pattern Mining*, in *Advances in Knowledge Discovery and Data Mining*, Z.-H. Zhou, H. Li, and Q. Yang, Editors. 2007, Springer Berlin Heidelberg. p. 388-400.

واژه نامه فارسی به انگلیسی

Abstract

ISLAMIC AZAD UNIVERSITY

QAZVIN Branch

Survey of Graph Mining

By

Iman Rezaeipour

Supervisor

Ph.D. Mohammad Saniee

Winter 2015