مساله افراز بندی گراف

آمنه قصاب مهرجرد

۱۷ تیر ۱۳۹۱

فهرست مطالب

لیست تصاویر

لیست جداول

۱.۰ چکیده

در شاخه بهینه‌سازی ترکیبیاتی با مسایل بسیار مهم و کاربردی آشنا می‌شویم، که هر یک با توجه به درجه سختی، در رده خاصی از مسایل قرار می‌گیرند. از جمله این مسایل، مساله افرازبندی در گراف است که در رده مسایل NP-Hard قرار داده می‌شود. در حالت کلی یک مساله افرازبندی در گراف را به صورت زیر در نظر می‌گیریم:

فرض کنید گراف G=(V,E) داده شده باشد. هدف در این مساله، تقسیم کردن راس‌های G به K زیر مجموعه جدا از هم، با اندازه‌های مساوی است به طوری‌که تعداد یال‌های عبوری بین هر مجموعه جدا از هم، کمترین باشد. در این‌جا برآنیم تا برخی از الگوریتم‌هایی نظیر جستجوی محلی، فراابتکاری، تقریبی و ... را برای مساله افرازبندی گراف ارایه کنیم.

فصل ۱

مقدمه

۱.۱ تاریخچه

همان‌طور که می‌دانیم مساله رنگامیزی در گراف‌ها یکی از زیر مساله‌های افرازبندی در گراف‌ها است. از این‌رو اگر بخواهیم به طور دقیق تاریخچه این مساله را مورد بررسی قرار دهیم، نقطه شروع این مساله به زمانی برمی گردد که مساله رنگامیزی نقشه اینگلند مورد بحث قرار گرفت. در این زمان فرنسیز گادری [۱] بر روی این مساله که چگونه با چهاررنگ می‌توان این نقشه را رنگ‌آمیزی کرد به طوری‌که هیچ یک از دو شهر مجاور همرنگ نباشند، متمرکز شد. وی یکی از دانشجویان دی مورگان [۲] در یکی از دانشگاه های لندن بود. در طول تلاش برای اثبات این مساله، برادر وی فردریک گادری [۳] وارد دانشگاه شد و به عنوان یکی از دانشجویان دی مورگان مشغول به تحصیل شد. فرنسیز تلاش‌های خود را برای این اثبات با برادرش در میان گذاشت و از او خواست تا این مساله را برای استاد خود، مورگان مطرح کند. پس از این‌که مورگان قادر به پاسخ‌گویی به این سوال نشد، در سال ۱۸۵۲ طی نامه‌ای به هامیلتون [۴] در دابلین، این مساله را با او در میان گذاشت. وی در جواب این نامه بیان کرد که نمی‌تواند زود به این سوال پاسخ دهد. پس از کندوکاوهای بسیار و تلاش برای یافتن اثباتی برای این مساله، آلفرد بری کمپ [۵] در سال ۱۸۷۹ ادعای اثبات این مساله را بیان کرد. اما هیوود [۶] این ادعا را رد و خود اثباتی برای پنج رنگ‌پذیری نقشه ارایه کرد.

---

[۱] Francis Gatheri
[۲] De Morgan
[۳] Fredrick Gatheri
[۴] Hamilton
[۵] Alferd Bray kempe
[۶] Hivood

۱.۱.۱ کاربردها

پس از معرفی تئوری و بیان کلیت مفروضات و اهداف در مساله افرازبندی، به سراغ کاربردهای این مساله می رویم. مساله افرازکردن گراف مانند سایر مسایل بهینه سازی در مسایل علمی نوین و با اهداف مختلف کاربرد دارد. طی سال ها، محققان بسیاری در زمینه های مختلف این مساله را مورد بررسی قرار داده و کاربردهایی از آن را معرفی کردند. استفاده موثر از حافظه توزیع شده در کامپیوترهای موازی، باعث متوازی شدن فشار عملیاتی و کم شدن هزینه ارتباطات بین پردازشگرها می شود. بسیاری از کاربردها در محاسبات علمی، مربوط به مساله تفاوت محاسبات بین پرازشگرها می شود که می تواند به راحتی در زبان گراف ها توصیف شود. به عنوان نمونه، برای اجرای موثر محاسبات بر روی سیستم های موازی، گراف باید به قطعاتی با تعداد رئوس برابر، شکسته شود به طوری که کمترین تعداد یال از بین قطعات بگذرد. این مدل گراف برای تعداد متناهی عنصر به خصوص در شبیه سازی پویای مولکولی مورد استفاده است. از کاربردهای دیگر این مساله می توان به تشخیص آسیب پذیری توان سیستم های بزرگ، مدیریت پایگاه های داده، طراحی مدارهای الکتریکی از جمله $VLSI$، مدل سازی سیستم های پویا و شبیه سازی،فیزیک مولکولی، ترتیب گذاری در DNA و طبقه بندی اسیدهای آمینه و غیره اشاره کرد. اکنون کاربرد این مساله در طراحی مدارهای الکتریکی و چگونگی ارتباط بین آن ها را بررسی می کنیم. قبل از آن لازم است بدانیم EDA [۷] دسته ای از ابزارهای نرم افزاری هستند که در جهت طراحی مدارهای الکتریکی مانند مدارهای مجتمع و یا صفحات مدارهای چاپی شکل گرفته اند. هدف از این ابزارها قراردادن قطعات در یک مدارالکتریکی است به طوری که بیشترین کارایی و کمترین هزینه را برروی سیستم داشته باشد . به عنوان مثال، برای طراحی مدار PCB، اگر صفحه مدار را به قسمت های مختلف تقسیم کنیم و قطعات متفاوت آن را در این بخش ها قرار دهیم به گونه ای که قطعاتی که بیشترین ارتباط را با یکدیگر دارند در یک بخش و قطعاتی که ارتباط کمتری با اجزای آن بخش دارند در قسمت های دیگر قرار دهیم، در واقع یک مساله افرازبندی در گراف را داریم که در جهت کمینه کردن هزینه، مدل می شود.

[۷] Electronic Design Automation

فصل ۲

الگوریتم های دقیق

می دانیم الگوریتم های دقیقی برای آن دسته از مسایل بهینه سازی که در رده ی مسایل Np-hard قرار می گیرند، وجود دارند که به برخی از آن ها اشاره می کنیم. از جمله الگوریتم های دقیقی برای حل مساله افراز بندی در گراف وجود دارد که با استفاده از روش شاخه و کران [۱] جواب بهینه را به دست می آورند. برای جزئیات این الگوریتم می توان به [۱] مراجعه کرد. همچنین مدل های برنامه ریزی عدد صحیح [۲] و برنامه ریزی خطی [۳] نیز برای این مساله ارایه شده است که مدل برنامه ریزی خطی آن در [۲] قابل دسترسی است و در این جا به توضیح مدل برنامه ریزی عدد صحیح می پردازیم.

۱.۲ مدل برنامه ریزی عدد صحیح

در مدل بندی عدد صحیح این مساله با فرض داشتن گراف همبند $G = (V, E)$ و دو پارامتر داده شده مانند K, L، هدف افراز کردن مجموعه رئوس به L زیر مجموعه مانند $v_1, ..., v_L$ به طوری که تعداد رئوس در هر افراز حداکثر K باشد و تعداد یال های عبوری بین هر افراز کمترین باشد. در این صورت اگر مجموعه ی یال های عبوری را با $I_G(V_i)$ نمایش دهیم و آن را به صورت زیر تعریف کنیم:

$$I_G(U) = \{u \in U | (v, u) \in A, v \in V/U\}$$

دراین صورت چون هدف تعیین مدل برنامه ریزی عدد صحیح برای این مساله است، متغیرهای تصمیم را به صورت زیر در نظر می گیریم:

$$\begin{cases} x_{vl} = 1, & if \quad v \in I_{G(V_l)} \\ x_{vl} = 0, & O.W \end{cases}$$

[۱] Branch and Bound
[۲] ILP
[۳] LP

و

$$\begin{cases} y_{vl} = ۱, & if \quad v \in V_l \\ y_{vl} = ۰, & O.W \end{cases}$$

در این صورت تابع هدف مساله با توجه به تعریف متغیر تصمیم x_{vl} به صورت زیر فرموله می کنیم:

$$\min \sum_{i=۱}^{n} x_{iv}$$

از طرفی هر راس دقیقا در یک افراز قرار می گیرد و حداکثر تعداد راس ها در هر افراز بنا به فرض مساله K است.بنابراین چنین محدودیت هایی را به صورت زیر در نظر می گیریم:

$$\begin{cases} \sum_{i=۱}^{L} y_{vi} = ۱, & v \in V \\ \sum_{v \in V} y_{vl} = K, & i \in ۱, ..., L \end{cases}$$

همچنین با توجه به تعریف متغیرهای تصمیم، برای هر راس دو حالت به وجود می آید. اول اینکه راس u یا در افرازi-ام قرار دارد و یا به راسی مانند v متصل است که در افراز مورد نظر قرار گرفته است. اصطلاحا راس u یک راس ارتباطی برای افرازi-ام است. پس دو محدودیت زیر همواره برقرار هستند:

$$\begin{cases} x_{vi} = y_{vi}, & v \in V, \quad i \in ۱, ..., L \\ y_{vi} = y_{ui} + x_{iv}, & \\ (u, v) \in E, & i \in ۱, ..., L \end{cases}$$

کنون می توان مدل برنامه ریزی عدد صحیح زیر را برای این مساله به صورت زیر در نظر گرفت.

$$\begin{cases} \min \sum_{i=۱}^{n} v \in V x_{iv} \\ \sum_{i=۱}^{L} y_{vi} = ۱, & v \in V \\ \sum_{v \in V} y_{vl} = K, & i \in \{۱, ..., L\} \\ x_{vi} = y_{vi}, & v \in V, \quad i \in \{۱, ..., L\} \\ y_{vi} = y_{ui} + x_{iv}, & (u, v) \in E, \quad i \in \{۱, ..., L\} \end{cases}$$

مراجعه کنید.

فصل ۳

الگوریتم های ابتکاری

۱.۳ الگوریتم تقریبی

در ابن بخش به معرفی الگوریتم تقریبی برای شماری از مسایل ماکزیمم افرازبندی در گراف می پردازیم. این الگوریتم که تعمیمی از کارهای هالپرینگ و زوئیک [۱] در سال۲۰۰۲ است، بر روی زیر مسایل مختلفی از مساله افرازبندی گراف مطرح می شود. کلید این بهبود در تکنیک زیر است. قبلا توسط هان و ژان [۲] در سال ۲۰۰۲ مشاهده شد که یک پارامتر انتخابی از ابرصفحه های تصادفی، منجر به فاکتورهای تقریبی بهتری نسبت به الگوریتم تقریبی ای که توسط جئومان و ویلیامسون [۳] در سال ۱۹۹۵ ارایه شد، می شود. آنچه کران تقریب را در این الگوریتم ها بهتر می کند، استفاده از ایده مناسب برای انتخاب پارامترها است. در این جا ابرصفحه های وابسته به چندین پارامتر را بررسی می کنیم که با استفاده از جواب برنامه ریزی خطی به دست آمده در این مقاله، انتخاب بهینه ای برای این پارامترها را فراهم می کنند.

۲.۳ مقدمه

در علوم کامپیوتر الگوریتم های تقریبی، الگوریتمهایی برای پیداکردن راه حل های تقریبی برای مسایل بهینه سازی هستند.این الگوریتم ها اغلب برای حل تقریبی مسایل NP-hard بکار می‌روند، زیرا بسیاری از مسایل بهینه سازی NP-hard هستند (در واقع بررسی کردن درستی جواب این گونه مسایل با حل کلی آنها معادل است) طبق تئوری پیچیدگی محاسباتی [۴] تا زمانیکه$P = NP$، الگوریتم‌های کارآمد با زمان چند جمله ای برای چنین مسایلی پیدا

[۱] Halpering and Zweek
[۲] Hane and Zhang
[۳] Geoman and Wiliyamson
[۴] Computational complexity theory

نخواهد شد . برخلاف روش‌های ابتکاری [۵] که راه حلهایی بهینه(اغلب بدون اثبات و بدون کران) برای جواب خود هستند، الگوریتم‌های تقریبی راه حلهایی شبه بهینه همراه با ضریبی برای میزان تقریب جواب واقعی ارائه می‌دهند. همچنین وجود جواب خود را در بازهٔ خطای اعلام شده تضمین می‌کنند .(مثلا جواب آنها ۲ برابر جواب بهینه است) منتها جواب خود را در زمان چندجمله‌ای تولید می‌کنند. الگوریتم‌های تقریبی برای مسایل Pنیز استفاده می‌شوند ولی به ازای ورودی‌های بزرگ خوب عمل نمی‌کنند.

۱.۲.۳ معرفی الگوریتم

اکنون با توجه به توضیحات بالا به معرفی یک الگوریتم تقریبی برای مساله ماکزیمم افراز بندی گراف می پردازیم.

فرض می کنیم گراف همبند G(V,E) با n راس، و وزن های مفروض w_ij برای هر یال در این گراف داده شده باشد. اما با توجه به این که الگوریتم تقریبی ارایه شده ، بر اساس زیر مسایل افرازبندی گراف طراحی شده است و با توجه به هر زیر مساله، پارامترها و کران تقریب متفاوتی به دست می آورد، در ابتدا به هر یک از این زیر مسایل را معرفی می کنیم.

۲.۲.۳ مساله max-k-cut

در این مساله هدف پیدا کردن زیرمجموعه ای مانند S با kراس از مجموعه ی رئوس است، به طوریکه وزن نهایی یال های گذرنده از S به V/S ویا وزن نهایی یال های گذرنده از V/S به S بیشترین باشد.

۳.۲.۳ مساله max-k-uncut

در این مساله هدف پیدا کردن زیرمجموعه ای مانند S با kراس از مجموعه ی رئوس است، به طوریکه وزن نهایی یال های گذرنده از زیرگراف القا شده توسط S و زیرگراف القا شده توسط V/S بیشترین باشد.

۴.۲.۳ مساله max-k-Directedcut

در این مساله هدف پیدا کردن زیرمجموعه ای مانند S با kراس از مجموعه ی رئوس است، به طوریکه وزن نهایی یال های گذرنده از زیرگراف القا شده توسط S به زیرگراف القا شده توسط V/S و یا وزن نهایی یال های گذرنده از زیرگراف القا شده توسط V/S به زیرگراف القا شده توسط S بیشترین باشد.

[۵] Heuristics

۵.۲.۳ مساله max-k-DenseSubGraph

در این مساله هدف پیدا کردن زیرمجموعه ای مانند S با kراس از مجموعه ی رئوس است، به طوریکه وزن نهایی یال های زیرگراف القا شده توسط S بیشترین باشد.

۶.۲.۳ مساله max-k-VertexCover

در این مساله هدف پیدا کردن زیرمجموعه ای مانند S با kراس از مجموعه ی رئوس است، به طوریکه وزن نهایی یال های touching S بیشترین باشد.

۷.۲.۳ مساله max-k-Directeduncut

در این مساله هدف پیدا کردن زیرمجموعه ای مانند S با kراس از مجموعه ی رئوس است، به طوریکه وزن نهایی یال های گذرنده از S به V/S بیشترین باشد.

تمام زیرمسایل بالا در رده ی مسایل Np-hard قرار می گیرند. جئومان و ویلیامسون در مقاله خود [۷]، با استفاده از مدل برنامه ریزی نیمه معین (SDP) الگوریتم تقریبی با کران ۰.۸۷۸ برای مساله Max-k-cut ارایه کردند. فیدل و لندبرگ [۶] نیز در [۸] ، از همین مدل برنامه ریزی نیمه معین برای یافتن کران تقریب استفاده کردند. هان، ژان و یه نیز در [۹] ، و همچنین هالپرینگ و زوئیک در [۱۰]، برای حل این دسته از مسایل الگوریتم های تقریبی ارایه کردند. این الگوریتم برای حل تقریبی ۶ زیر مساله بالا به کار می رود که تعمیمی از کارهای هالپرینگ و زوئیک است. در لم ۲ از این مقاله نشان داده می شود، فاکتورهای تقریبی به انتخاب یک مجموعه از پارامترها وابسته اند، که کلید اصلی این مشاهدات، انتخاب بهینه از پارامترها است که به وسیله یک برنامه ریزی خطی متناهی، تعیین می شوند.

برای انتخاب مجموعه $S \subset V$، مجموعه ی یال ها را به صورت زیر تقسیم می کنیم:

$$\begin{cases} E = S_1 \cup S_2 \cup S_3 \cup S_4 \\ S_1 = (i,j) | i,j \in S \\ S_2 = (i,j) | i \in S, j \in V/S \\ S_3 = (i,j) | j \in S, i \in V/S \\ S_4 = (i,j) | i,j \in V/S \end{cases}$$

بیان شد که این الگوریتم تقریبی بر روی ۶ زیر مساله طراحی شده است، لذا بر حسب این که کدام زیر مساله مورد نظر است یال های مجموعه را به بخش های جدا از هم تفکیک کردیم. اما این سئوال مطرح می شود، که چگونه زیرمساله مورد نظر تعیین شود؟ در این جا پارامتری برای این مجموعه های تفکیک شده ارائه می شود که بر حسب این که

[۶] Fidel and Landberg

الگوریتم برای کدام مساله و ماکزیمم سازی وزن بر روی چه یال هایی است، مقدار می گیرد. در این صورت تعریف می کنیم:

$$\begin{cases} a_i = ۱ & S_i \\ a_i = ۰ & else \quad i \in ۱, ..., ۴ \end{cases}$$

با این تعریف پارامترهای a_i برای مساله Max-k-cut به صورت زیر خواهد بود:

مساله با این تعریف، از نماد زیر برای نمایش جمع جبری وزن ها در مجموعهS استفاده می کنیم:

$$\begin{cases} W(F) = \sum\limits_{(i,j)\in F} w_i j \\ F \subset E \end{cases}$$

و خواهیم داشت:

$$w_{(a_۱,a_۲,a_۳,a_۴)}(S) = \sum_{i=۱}^{۴} a_i w(S_i)$$

که با تعریف بالا، تابع هدف مساله را به صورت زیر می نویسیم:

$$\begin{cases} \max w_{(a_۱,a_۲,a_۳,a_۴)}(S) \\ s \subset V, |S| = k \end{cases}$$

اکنون فرض کنید OPT جواب بهینه مساله ماکزیمم سازی باشد، و w^* جواب بدست آمده برای مساله نیمه معین باشد. با استفاده از الگوریتمی نشان می دهیم در یک زمان چند جمله ای کران تقریبی با تقریب Q به دست آورده می شود که در آن Q عددی بین ۰ تا ۱ است. در این صورت اگر گراف همبند و وزن دار $G = (V, E)$ با n راس به عنوان ورودی برای این الگوریتم داده شده باشد و همچنین پارامترهای ($a_۴$،$a_۳$،$a_۲$،$a_۱$) برای مساله ماکزیمم سازی تعیین شده باشد، خروجی الگوریتم یک مجموعه با k راس و بیشترین وزن روی یال های مورد نظر در آن مجموعه، می باشد. اکنون مساله برنامه ریزی خطی نیمه معین را به صورت زیر در نظر می گیریم. لازم به توضیح است که این مساله توسط هالپرینگ و زوئیک در [۱۰] ارایه شده است.

$$\begin{cases} \max \sum\limits_{i,j=۱}^{n} \frac{۱}{۴} w_i j[(a_۱ + a_۲ + a_۳ + a_۴)+ \\ (a_۱ + a_۲ - a_۳ - a_۴)X_{i۰} + (a_۱ - a_۲ + a_۳ - a_۴)X_{j۰} + (a_۱ - a_۲ - a_۳ + a_۴)X_{ij} \\ \text{s.t} \\ \sum\limits_{i,j=۱}^{n} X_{i۰} = (۲k - n) \\ \sum\limits_{i,j=۱}^{n} X_{ij} = (۲k - n)^۲ \\ X_{ij} + X_{jl} + X_{il} \geq -۱ \\ i, j, l \in ۰, ..., n \\ X_{ij} - X_{jl} - X_{il} \geq -۱ \quad i, j, l \in ۰, ..., n \\ \sum\limits_{i,j=۱}^{n} X_{ij} = ۱/۲((۲k - n)^{۲-n}) \end{cases}$$

بیان شد که OPT جواب بهینه برای مساله اصلی و $w*$ جواب به دست آمده از حل این برنامه ریزی نیمه معین باشد. در یک الگوریتم تقریبی پس از این مرحله به دنبال یافتن کرانی برای این جواب هستیم تا تقریبی از الگوریتم حاصل شود. حال با در نظر گرفتن جواب حاصل از مساله نیمه معین، با استفاده از الگوریتمی آن را به یک جواب شدنی برای مساله اصلی تبدیل می کنیم. این الگوریتم که روند کردن نامیده می شود، در یک زمان چندجمله ای کران تقریبی برای مساله اصلی به دست می آورد. در این الگوریتم از الگوریتمی برای تنظیم پارامترهای مساله استفاده می شود، که گام های این الگوریتم را به صورت زیر بیان می کنیم:

فرض کنید Q_0 بهترین کران تقریبی به دست آمده برای زیر مساله مورد نظر باشد. در ابتدا کران Q را با Q_0 مقداردهی می کنیم. در گام بعد، پارامترهای مورد نظر را در بازه های خود مقدار می دهیم. در اثبات یک قضیه های اساسی در این مقاله، با استفاده از پارامترها و جواب های به دست آمده، مدل برنامه ریزی خطی را ارایه می کنند که منجر به پیدا کردن کرانی برای این الگوریتم می شود. این مدل برنامه ریزی خطی را در [۱۱] می توان با جزئیات بیشتری مورد بررسی قرار داد. این مدل برنامه ریزی خطی به صورت زیر تعریف شده است:

$$\begin{cases} \min z \\ \text{s.t} \\ a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 \geq h(Q, O, t, y, k, x_1, x_2) \\ y_1 + y_2 + y_3 + y_4 = 1 \\ y_1, y_2, y_3, y_4 \geq 0 \\ z \geq f_{a_1,a_2,a_3,a_4}(Q, O, t, y, k, x_1, x_2) \end{cases}$$

حال اگر مساله برنامه ریزی خطی بالا با پارامترهای معلوم حل شود، به اندازه ۰.۰۰۰۱ به مقدار Q افزوده می شود و در غیر این صورت به گام تعیین پارامترها برگشته و پارامترها را دوباره مقداردهی می کنیم. با انجام محاسبات لازم برای الگوریتم بالا کران تقریبی برای مساله اصلی به دست می آید که Q دربازه ۰ تا ۱ قرار می گیرد. این الگوریتم برای نمونه ها و زیر مسایل مختلفی اجرا شده است که در هر زیر مساله کران تقریب به دست آمده، متفاوت است. در این جا جواب های تقریبی به دست آمده با بهترین جواب های به دست آمده برای این دسته از مسایل به صورتی که در جدول زیر مشاهده می کنید، مقایسه شده است. نتایج حاصل از این مقایسه نشان داد، که اگر از مدل برنامه ریزی نیمه معین برای مرحله آزادسازی مساله استفاده شود، کران تقریبی بهتری نسبت به سایر الگوریتم های تقریبی که از مدل های دیگر برای آزادسازی مساله اصلی استفاده می کنند، به دست خواهد آمد.

۳.۳ الگوریتم حریصانه

در بسیاری از موارد مساله از ما می‌خواهد که یک متغیر را کمینه یا بیشینه کنیم و یا اینکه صورت مساله مستقیما از ما نمی‌خواهد که چیزی را کمینه یا بیشینه کنیم، اما می‌توان این مسئله را تبدیل به یک چنین مساله‌ای کرد. در اکثر مسایل از این دست ما باید تعدادی انتخاب انجام دهیم و مساله در تعدادی مرحله حل شود. بازهٔ وسیعی از این مسایل توسط الگوریتم‌های حریصانه قابل حل‌اند. الگوریتم‌های حریصانه در هر مرحله بر اساس یک ایده حریصانه در نظر می‌گیرند که کدام انتخاب در حال حاضر بهترین وضعیت را برای ما رقم می‌زند و آن را انجام می‌دهند. بدون در نظر گرفتن این که انجام این حرکت در آینده ممکن است به ضرر ما تمام شود. بنابراین الگوریتم‌های حریصانه همیشه جواب درست را به ما نمی‌دهند اما خیلی وقت‌ها هم اینطور نیست این دسته از الگوریتم‌ها در علوم رایانه کاربرد وسیعی دارند. اکنون

با فرض داشتن گراف G=(V,E) به طراحی الگوریتم حریصانه ای که در زیر بخش های آن از الگوریتم های حریصانه دیگر استفاده می شود، می پردازیم. الگوریتم های حریصانه ی دیگری در دوافراز بندی گراف ها استفاده شده است، که از آن جمله می توان به الگوریتم حریصانه که توسط کرنیقان و لین [۷] که در [۳] ارایه شده است، اشاره کرد . اما ایده الگوریتم استاندارد بر اساس قرار دادن دو راس به طور تصادفی در دو مجموعه ی افراز است. با استفاده از این ایده راس ها در هر مرحله به صورت یک روند دوری به دو مجموعه اضافه می شود.بعد از آن باتیتی و برتسی [۸] دو الگوریتم حریصانه برای دو افرازبندی در گراف ها، به نام Min-Max Greedy و Diff Greedy ارایه کردند که در [۴، ۵] قابل دسترس است. ایده حریصانه در الگوریتم Min-Max Greedy با اضافه کردن قاعده انفصال گره به ایده حریصانه در الگوریتم استاندارد به دست می آید. به عبارت دیگر، برای راس هایی که افزایش اندازه برش یکسان دارند، راسی انتخاب شود که بیشترین همسایگی را در افراز مورد نظر داشته باشد.این ایده باعث می شود در گام های بعدی، تعداد یال های عبوری بین افراز ها کمترین باشد. ایده حریصانه در الگوریتم Diff Greedy ، استفاده از محک انتخاب راس است، که در آن راس ها به ترتیب ارجحیت شان ، انتخاب می شوند. به عبارت دیگر، راسی که کمترین اختلاف را بین یال های عبوری جدید با یال های داخلی افراز داشته باشد، برای قرار دادن در آن افراز انتخاب می شود. نتایجی که از این الگوریتم ها گرفته شد، برای طراحی الگوریتم حریصانه ای برای K افراز بندی در گراف ها مورد استفاده قرار گرفت. ایده این الگوریتم تعمیمی از ایده الگوریتم Diff Greedy است. این الگوریتم را با نام k-Greedy معرفی می کنیم، که ایده آن بر اساس قرار دادن k راس به صورت تصادفی در k افراز مختلف است. در

[۷] Kernighan and Lin
[۸] Battiti and Bertossi

هر تکرار از این الگوریتم، راسی که کمترین افزایش در اندازه برش را داشته باشد، به افراز اضافه می شود.اجرای این الگوریتم به داده ساختار صف نیاز دارد و برای هر افراز یک آرایه به کار می برد. در یک اجرا از این الگوریتم، اگر راسی به افراز مورد نظر اضافه شود، در ساخت حریصانه الگوریتم برای راس های بعد، آن افراز برداشته می شود. این همان کاری است که در الگوریتم MinP Greedy انجام می شود. بهبود دیگری بر این الگوریتم، استفاده از الگوریتم MaxN Greedy است. ایده این الگوریتم بر اساس انتخاب راسهایی است که بیشترین همسایگی در افراز مورد نظر را دارا باشند. به طور شهودی آن است که راسی که بیشترین یال های داخلی در یک افراز را داشته باشد، در گام های بعدی از الگوریتم، یال های عبوری کمتری دارد. از ترکیب الگوریتم های بیان شده ، الگوریتم حریصانه ی MinP-MaxN به دست می آید. در این الگوریتم، مجموعه ای از رئوس به صورت حریصانه انتخاب شده و از قاعده انفصال گره برای انتخاب راس ها در مراحل بعد استفاده می شود. طریق دیگری برای الگوریتم k-Greedy این است که به جای انتخاب k راس تصادفی ، فقط یک راس به صورت تصادفی انتخاب شود و آن را با افراز دیگر جایگزین کنیم.به عنوان مثال حالت k=۳ را در نظر بگیرید. در ۳ افراز کردن گراف، در ابتدا مجموعه راس های V را به دونیم بخش که هر افراز دارای n/۲ راس است، تقسیم می کنیم. اکنون اگر یکی از این افرازها را به دو افراز دیگر تقسیم کنیم، سه افراز با اندازه های n/۴ وn/۴ وn/۲ به دست می آید که افراز یکنواختی نیست. بنابراین از ایده حریصانه Diff برای ارایه نتیجه بهتر در کیفیت جوابها استفاده می کنیم.

۴.۳ k- الگوریتم حریصانه برای افراز بندی گراف

در ابتدا به معرفی برخی نماد ها در این الگوریتم می پردازیم. در این جا از ۰ تا $k-۱$ برای اندیس های افراز ها استفاده می کنیم. فرض کنید X_i افرازی باشد که راس i متعلق به آن است، در این صورت برای راس i و افراز p نماد هایی به صورت زیر تعریف می کنیم.

از نماد N(i,p) برای نمایش تعداد یال های گذرنده از راسi که راس انتهایی آن ها در افراز p قرار دارد، استفاده می کنیم و که به صورت زیر نمایش داده می شود:

$$N(i,p) = |(i,j) : j \ \ in \ \ p|$$

و از نماد Ext(i,p) برای نمایش تعداد یال های گذرنده از راسi که راس انتهایی آن ها در سایر افرازها به جز افراز p ام قرار دارد، استفاده می کنیم و که به صورت زیر نمایش داده می شود:

$$Ext(i,p) = \sum_{j=۰}^{k-۱} N(i,j)$$

که از تفاضل این دو اندازه برش برای افراز p ام به دست می آید که به صورت زیر است:

$$Diff(i,p) = Ext(i,p) - N(i,p)$$

۵.۳ الگوریتم k-Greedy

در ابتداk راس تصادفی در k زیر مجموعه قرار می گیردو در یک روند دوری برای اضافه کردن راسی مانندi به افراز p، راسی را که کمترین Diff(i,p) را دارد، به افراز اضافه می شود.

۶.۳ الگوریتم Greedy MinP

در الگوریتم k-Greedy افرازها از ۰ تا k-۱ پر می شوند. برای هر افراز مانند j نماد minval(j) را که به ازای تمام رئوس مینیمم مقدار Diff(i,p) را محاسبه می کند، به کار می بریم. در این صورت افرازی مانندp که این مینیمم مقدار در آن اتفاق افتاده است به عنوان افرازی انتخاب می شود که راس جدیدبه آن اضافه می شود، و به صورت زیر تعریف می کنیم:

$$\begin{cases} \min \quad val(p) = \min \quad val(j) \\ j \in ۰, ..., k-۱ \end{cases}$$

که درآن

$$\min val(j) = \min Diff(i,j) \qquad i \in ۱, ..., n$$

باید دقت کرد که در هر مرحله افراز به صورت حریصانه انتخاب می شود، در k تکرار،k زیرمجموعه انتخاب می شود. این الگوریتم MinP نامیده شده، زیرا در هر مرحله افرازی با مینیمم مقدار minval انتخاب می شود.

۷.۳ الگوریتم Greedy MaxN

الگوریتم حریصانه ماکزیمم همسایگی ، از بین مجموعه ای از رئوس که مینیمم اندازه Diff(i,addset) را دارند، راسی را انتخاب می کند که بیشترین همسایگی را در این افراز داشته باشد.

۸.۳ الگوریتم MinP-MaxN

در این الگوریتم ترکیبی که در ابتدا به جای برداشتن k راس به صورت تصادفی، یک راس را بر می دارد. با کاهش تعداد راس ها از k به یک، یال های معرفی شده در برش کاهش می یابد . در حالت کلی روند انجام این الگوریتم به صورت زیر است.

---

remaning -> V
f<- ۰
v<- random vertex $rv \in remaning$
set(۰)<- v
remaning<- remaning v
while|remaning|> ۰ Do
forall j $\in ۰, ..., k - ۱$ Do
minval(j)= min Diff(i،j) $i \in remaning$
addset<- min minval(j)
$j \in ۰, ..., k - ۱$
v<-$rv \in i : Diff(i, addset) = \min val, N(i, addset) ismaximize$
set(addset)<- set(addset)+v
remaning<- remaning{v}
$f = f + Ext(v, addset)$
return f

۱.۸.۳ پیچیدگی محاسباتی

در این الگوریتم عملیات مربوط به مقداردهی به طور واضح متعلق به $O(n)$ است. حلقه اصلی در بدترین حالت $O(n)$ زمان نیاز دارد. در داخل حلقه k بار عمل حذف و حداکثر $O(k.deg(v))$ زمان برای به هنگام کردن انجام می شود.لذا پیچیدگی زمانی زمانی الگوریتم k-Greedy متعلق به $O(k|E|)$ زمان نیاز دارد.از طرفی زمان مورد نیاز برای محاسبه $N(i, addset)$ در بدترین حالت $O(n)$ است. در نهایت پیچیدگی الگوریتم مورد بررسی متعلق به $O(k|E| + |V|^۲)$ خواهد بود. برای دریافت جزئیات بیشتر در رابطه با نحوه محاسبه پیچیدگی به [۶]

۲.۸.۳ الگوریتم جستجوی محلی

۳.۸.۳ مقدمه

در علم کامپیوتر، جستجوی محلی یک روش فرا ابتکاری برای حل مسایل بهینه سازی سخت، بصورت محاسباتی می باشد.جستجوی محلی می تواند در مسایلی مورد استفاده قرار گیرد که می تواند به عنوان یافتن راه حلی برای حداکثر

رساندن یک معیار در میان تعدادی از راه حل‌های پیش رو، مطرح شود. الگوریتم‌های جستجوی محلی از یک راه حل به راه حل دیگر در فضای جستجو با استفاده از تغییرات محدود حرکت می کنند تا یک راه حل به نظر مطلوب یافت شود یا یک زمانی سپری شود. الگوریتم‌های جستجوی محلی در تعداد زیادی از مسایل سخت محاسباتی، از جمله مسایلی از علم کامپیوتر (مخصوصا هوش مصنوعی)، ریاضیات، تحقیق در عملیات، مهندسی، بیو انفورماتیک به طور گسترده به کار می رود. نمونه‌های از الگوریتم‌های جستجوی محلی، الگوریتم‌های WalkSAT و الگوریتم $opt-$ برای مساله فروشنده دوره گرد می باشد. به طور معمول الگوریتم‌های جستجوی محلی، الگوریتم هایی تقریبی یا نا کامل هستند، از این جهت که حتی اگر بهترین راه حل پیدا شده توسط الگوریتم بهینه نباشد جستجو ممکن است متوقف شود.این حتی در صورتی که خاتمه ی الگوریتم، به علت عدم امکان بهبود راه حل باشد، می تواند اتفاق بیفتد، بدین صورت راه حل بهینه می تواند دور از مجاورت و همسایگی راه حل هایی باشد که الگوریتم از آن‌ها عبور کرده است.این الگوریتم دارای سه گام اصلی نقطه شروع، ساختار همسایگی و محاسبه تغییرات است. در جستجوی محلی با داشتن ساختار همسایگی و با استفاده از یک نقطه ابتدایی (نقطه شروع) به دنبال جواب بهتر در این همسایگی هستیم. باید توجه داشت که پیچیدگی این الگوریتم به تعریف نوع ساختار بستگی دارد. ساختار کلی این الگوریتم به صورت زیر است.

Local Search Algorithm – Initializing :
Neighborhood type
Starting point S°(
Condition stop
S^{cur} –> S°
if Condition stop is satisfied
return S^{cur}
Choose $s \in N(S^{cur})$
if $C(s) < C(S^{cur})$
S^{cur} –> s and goto ۳

در این الگوریتم $C(s)$ مقدار تابع هدف به ازای جواب s است که با توجه به مساله تعیین می شود.

۹.۳ جستجوی محلی در افراز بندی گراف

اکنون به پیاده سازی این الگوریتم برای مساله افرازبندی یکنواخت بر روی گراف مفروض $G = (V, E)$ می پردازیم. اگر ماتریس D نشان دهنده فاصله راس ها درگراف داده شده باشد، هدف پیداکردن دوافراز یکنواختی مانندA و B

است به طوری که $C(A,B) = \sum_{i\in A, j\in B} d_{ij}$ کمترین باشد. همان طور که بیان شد برای جستجوی محلی نیاز به نقطه شروع است که برای مساله مورد نظر این نقطه یه راحتی انتخاب دو افراز یکنواخت به صورت تصادفی است.

توجه کنید که پیداکردن نقطه شروع برای هر مساله ای ساده نیست!

گام بعد تعیین همسایگی برای الگوریتم است. راه های مختلفی برای تعیین همسایگی وجود دارد که با توجه به تعریف آن ها پیچیدگی محاسباتی مساله متفاوت می شود. در این جا همسایگی را به صورت افرازهای یکنواختی مانند (A', B') که از تعویض دو عنصر در دو افراز(A,B) به دست می آید و به صورت زیر تعریف می شود:

$$N_۲(A,B) = \{(A',\ B')|(A',\ B')\ is\ obtained\ by\ pari-exchang\ \ from\ \ (A,B)\}$$

در گام نهایی نیز محاسبه تغییرات برای تابع هدف مساله را به صورت زیر داریم:

برای راس دلخواه a از افرازA، مجموع هزینه یال های خروجی از راس a به راس هایی در B را با $E(a)$ و مجموع هزینه یال های ورودی از راس هایی در A به راس a را با $I(a)$ نشان می دهیم و $D(a)$ را تفاضل این دو مقدار قرار می دهیم. به طور متقارن مقادیر هزینه های فوق را برای هر راس مانند b در افراز B محاسبه می کنیم بنابراین داریم:

$$\begin{cases} I(a) = \sum_{j\in B} d_{aj} \\ E(a) = \sum_{i\in I} dia \\ D(a) = E(a) - I(a) \end{cases}$$

لم ۱.۹.۳. *میزان بهبود تابع هدف پس از جابه جایی دو عنصر a, b برابر است با*

$$g(a,b) = D(a) + D(b) - ۲d_{ab}$$

برهان. فرض کنید قبل از تعویض دو عنصرa, b از دو افرازA, B هستیم. دراین صورت اگر مقدار تابع هدف را با T نمایش دهیم، مقدار آن از به صورت زیر محاسبه می شود:

$$T = z + E(a) + E(b) - d_{ab}$$

که درآن $E(a)$ هزینه یال های خروجی از راس a و $E(b)$ هزینه یال های خروجی از راس b است. در این صورت یال عبوری (a,b)دو بار در محاسبه هزینه، شمرده می شوند، بنابراین به اندازه d_{ab} از این هزینه کم می کنیم و درآخر هزینه روی سایر یال ها را (z) را در نظر می گیریم. اکنون اگر بر طبق الگوریتم دو عنصر a, b برای تولید جواب بعدی در نظر گرفته شوند، هزینه پس از این جابه جایی را T' محاسبه می کنیم:

$$T' = z + I(a) + I(b) + d_{ab}$$

که درآن $I(a)$ مجموع هزینه یال های ورودی از راس هایی در A به راس a است که به طور متقارن برای راس bدر افرازB نیز محاسبه می شود $(I(b))$. ازطرفی چون در این محاسبات هزینه بین دو راس a, b شمرده نمی شود، این مقدار (d_{ab}) را به همراه هزینه روی سایر یال ها (z) در محاسبات اضافه می کنیم. در این صورت میزان تغییر تابع هدف از تفاضل این دو مقدار و به صورت زیر به دست می آید.

$$g(a,b) = T - T' = E(a) - I(a) + E(a) - I(a) - ۲d_{ab} = D(a) + D(b) - ۲d_{ab}$$

□

اکنون می توان روند کلی جستجوی محلی را به صورت زیر برای مساله پیاده کرد:

Suppos initial solution (A،B)
while g(a،b)!=۰ DO
g(a'، b'(=Max g(a،b) $a \in A$، $b \in B$
A' = A – $a' \cup b'$
B' =B – $b' \cup a'$
A -> A' B -> B'

توجه کنید در این الگوریتم شرط توقف زمانی است که هیچ بهبودی برای تابع هدف پیدا نشود یعنی زمانی که $g(a,b)$ برابر صفر باشد ، بهبودی برای تابع نیست و الگوریتم خاتمه می یابد.

۱۰.۳ پیچیدگی زمانی

در هر تکرار از الگوریتم به ازای تمام رئوس در دو افراز میزان تغییر تابع هدف محاسبه می شود و الگوریتم تا زمانی که این جابه جایی دو عضو در دو افراز تابع هدف را بهتر کند ادامه می یابد. پس در بدترین حالت همه n عنصر در افراز A با n عنصر در افراز B جا به جا می شوند که دراین حالت پیچیدگی الگوریتم متعلق به مرتبه زمانی$(O(n^۲)$است.

۱۱.۳ نتایج و تعمیم الگوریتم

طبق نتایج محاسباتی این جستجوی محلی در ۱۰ درصد از مواقع موفق به پیداکردن جواب بهینه سراسری برای nهای کمتر از ۳۶ است. در این الگوریتم تغییراتی بر روی روند جستجو انجام شده و الگوریتم دیگری به نام Variable Deapth Search توسط کرنیقان و لین به وجود آمد که درآن به جای تعویض دو عنصر از دو افراز ، یک زیر مجموعه

مانند Xاز افراز Aرا با یک زیر مجموعه مانند Yاز افراز B جا به جا می کند. این الگوریتم نیز در بدترین حالت پیچیدگی زمانی $(O(n^۲)$ دارد.

فصل ۴

الگوریتم های فراابتکاری

۱.۴ الگوریتم جستجوی ممنوع

۲.۴ تاریخچه

الگوریتم جستجوی ممنوع [۱] یک الگوریتم بهینه‌سازی فراابتکاری است که برای اولین بار در سال ۱۹۸۶ توسط گلوورِ [۲] [۱۲، ۱۳، ۱۴] معرفی شد و اولین کتابی که کاملاً به جستجوی ممنوع اختصاص داشت در سال ۱۹۹۷ توسط گلوورِ و لاگونا منتشر شد. واژهٔ تابو به معنای شیء مقدسی است که به دلیل قداست نباید آن را لمس کرد از تُنگان زبان مردم جزایر پلینزی در اقیانوس آرام گرفته شده‌است. بر اساس واژه‌نامه‌ی وِبستر، امروزه این واژه در معنای «ممنوعیت ایجاد شده به دلیل فرهنگ اجتماعی برای ایجاد اقدام حفاظتی» یا «ممنوعیت چیزی که دارای ریسک است، به کار می‌رود. ریسکی که در الگوریتم جستجوی ممنوع از آن اجتناب می‌شود، خطر مسیرهای نامناسب است.

۳.۴ ساختار کلی جستجوی ممنوعه

برای رسیدن به جواب بهینه در یک مساله بهینه‌سازی، الگوریتم جستجوی ممنوع ابتدا از یک جواب اولیه شروع به حرکت می‌کند. سپس الگوریتم بهترین جواب همسایه را از میان همسایه‌های جواب فعلی انتخاب می‌کند. در صورتی که این جواب در لیست ممنوع [۳] قرار نداشته باشد، الگوریتم به جواب همسایه حرکت می‌کند. در غیراین‌صورت الگوریتم معیاری به نام تابع تنفس [۴] را چک خواهد کرد. بر اساس معیار تنفس اگر جواب همسایه از بهترین جواب یافت شده تا کنون بهتر باشد، الگوریتم به آن حرکت خواهد کرد، حتی اگر آن جواب در لیست ممنوع باشد. پس از

[۱] Tabu Search- (TS)
[۲] Glover
[۳] Tabu List
[۴] Aspiration Function

حرکت الگوریتم به جواب همسایه، لیست ممنوع به هنگام می‌شود. به این معنا که حرکت قبل که بوسیله‌ی آن به جواب همسایه حرکت کردیم در لیست ممنوع قرار داده می‌شود تا از بازگشت مجدد الگوریتم به آن جواب و ایجاد سیکل جلوگیری شود. در واقع لیست ممنوع ابزاری در الگوریتم جستجوی ممنوع‌است که توسط آن از قرار گرفتن الگوریتم در بهینه‌ی محلی جلوگیری می‌شود. پس از قرار دادن حرکت قبلی در لیست ممنوع ، تعدادی از حرکت‌هایی که قبلا در لیست ممنوع قرار گرفته بودند از لیست خارج می‌شوند. مدت زمانی که حرکت‌ها در فهرست ممنوعه قرار می‌گیرند توسط یک پارامتر که زمان ممنوع [۵] نام دارد تعیین می‌شود. حرکت از جواب فعلی به جواب همسایه تا جایی ادامه می‌یابد که شرط خاتمه دیده شود. شرط‌های خاتمه متفاوتی می‌توان برای الگوریتم در نظر گرفت. به طور مثال محدودیت تعداد حرکت به جواب همسایه می‌تواند یک شرط خاتمه باشد.

کرنیقان و لین در سال ۱۹۷۰ الگوریتم ابتکاری برای به دست آوردن جوابی در مساله افرازبندی گراف ارایه کردند [۱۵] که بعد ها سرعت اجرای این الگوریتم توسط فیداک و متیس [۶] با به کارگیری داده ساختاری مناسب ارایه شد [۱۶] . در انتها برای مقایسه الگوریتم ها به این الگوریتم با عنوان الگوریتم $KLFM$ بر می گردیم. البته عملکرد نتایج روی این الگوریتم نامنظم است که در این جا از نتایج این الگوریتم که در [۱۷، ۱۸] آورده شده برای مقایسه استفاده می کنیم. الگوریتم دیگری که برای مقایسه به کار گرفته می شود، الگوریتم کاهش تدریجی دما [۷] است، که در [۱۹] ارایه شده است. در اینجا به بررسی الگوریتم جستجوی ممنوع برای به دست آوردن جوابی مناسب در افراز بندی گراف می پردازیم، که جواب هایی قابل مقایسه با دو الگوریتم فوق از لحاظ کیفیت و سرعت اجرا به دست می آورد.

۴.۴ معرفی الگوریتم

فرض کنید گراف $G = (V, E)$ یک گراف ناهمبند با هزینه $c(u, v)$ برای روی هر یال داده شده باشد. در پیاده سازی این الگوریتم فقط گراف هایی را درنظر می گیریم که تعداد رئوس آن ها زوج باشد. می دانیم که یک مساله افراز بندی یکنواخت گراف با کمترین اندازه برش که با $BGPP$ نشان داده می شوند، افراز مجموعه رئوس به دو زیر زیر مجموعه با اندازه مساوی است، به طوریکه هزینه کمترین باشد. واضح است که برای به دست آوردن یک مجموعه برش با کمترین هزینه بدون توجه به اندازه ی زیر مجموعه ها در افرازبندی، می توانیم از هر الگوریتم ماکزیمم جریان چند جمله ای برای به دست آوردن جواب بهینه استفاده کنیم. اگرمساله را به اندازه هر زیر مجموعه، محدود کنیم در

[۵] tabu tenure
[۶] Fidducia and Matheyese
[۷] Simulated Annealing-(SA)

واقع مساله سخت تر می شود. رض کنید گراف $G = (V, E)$ یک گراف ناهمبند با هزینه $c(u, v)$ برای روی هر یال داده شده باشد. در پیاده سازی این الگوریتم فقط گراف هایی را درنظر می گیریم

که تعداد رئوس آن ها زوج باشد. می دانیم که یک مساله افراز بندی یکنواخت گراف با کمترین اندازه برش که با $BGPP$ نشان داده می شوند، افراز مجموعه رئوس به دو زیر زیر مجموعه با اندازه مساوی است، به طوریکه هزینه کمترین باشد. واضح است که برای به دست آوردن یک مجموعه برش با کمترین هزینه بدون توجه به اندازه ی زیر مجموعه ها در افرازبندی، می توانیم از هر الگوریتم ماکزیمم جریان چند جمله ای برای به دست آوردن جواب بهینه استفاده کنیم. اگرمساله را به اندازه هر زیر مجموعه، محدود کنیم در واقع مساله سخت تر می شود.

قضیه ۱.۴.۴. *مساله افرازبندی یکنواخت درگراف NP-hard است.*

اثبات این قضیه در [۲۰] آمده است.

الگوریتم جستجوی ممنوع برای بسیاری از مسایل NP-hard جواب های خوبی بدست آورده است که مدل کلی این الگوریتم به صورت زیر است:

۵.۴ روند اجرا

روند کلی الگوریتم جستجوی ممنوع روی مفروضات مساله،انتخاب یک جواب بدتر است که برای خارج شدن از بهینه محلی لازم است. در هر تکرار از این جستجو بهترین جواب در همسایگی انتخاب می شود. در این الگوریتم از دو ساختار لیست ممنوع و تابع تنفس استفاده می کند. این دو ساختار برای جلوگیری از به دور افتادن و نگهداری اطلاعات حرکت های قبلی است که به منظور متنوع بودن حرکت و محدود کردن تعداد جستجوها برای پیدا کردن جواب خوب طراحی شده است. لیست ممنوع با توجه به محدودیت ها ی مساله به شکل های مختلفی طراحی می شود. ساده ترین و متداول ترین شکل لیست ممنوع، لیست خطی است که حداکثر k حرکت اخیر را ذخیره می کند. هدف اصلی این لیست محدود کردن جستجوی مستقیم است و باعث می شود که حالتی که قبلا بررسی شده، بدست نیاید و این عمل در بسیاری از مواقع ما را از گیر کردن در بهینه های محلی باز می دارد. به علاوه برای لیست ممنوع یک تابع به عنوان تابع تنفس فراهم می شود که دارای قابلیت از بین بردن محدودیت های ممنوع است. این روند به عنوان مکانیزمی برای متنوع بودن جستجو و پیدا کردن نواحی جدید در جستجو است. در شکل زیر نمونه ای از اجرای این الگوریتم بر روی گرافی با ۱۰۰ راس مشاهده می شود.

Input
An instance of the problem to be solved
Definitions
X : Set of feasible solution
f : Objective function
N(x): Neighborhood of $x \in X$
T: Tabu list
A: Aspiration function
max: Maximum number of iterations between improvement
Initialization:
Seti=۰;
Generate an initial solution $x_i \in X$;
Initialize tabu list(s) T and aspiration function A;
Set best = x_i, bestcost =f(best) and besti=i;
Body
While(i−besti > max)
i=i+۱;
locate the best $x_i \in N(x_{i-۱})$ where x_i does not
satisfy tabu conditions or if aspiration function overrules tabu conditions;
if) f)x_i (> bestcost)
best = x_i; bestcost=f(best); besti=i;

update tabu list(s) T;
update aspiration function A;

Output
best and bestcost
Figure :۱ General framework of Tabu Search

۶.۴ تنظیم پارامترهای الگوریتم

ورودی الگوریتم برای به دست آوردن مینیمم اندازه برش در حالت افراز بندی یکنواخت، گراف وزن دار $G = (V, E)$ است. مجموعه جواب های ممکن X را به صورت زیر تعریف می کنیم.

$$X = \{(S_۱, S_۲)|S_۱ \cup S_۲ = V, |S_۱| = |S_۲|\}$$

و برای هر $x = (s_۱, s_۲) \in X$، تابع هدف را به صورت زیر تعریف می کنیم:

$$f(x) = \min \sum_{u \in s_۱, v \in s_۲} c(u, v)$$

در ابتدا یک جواب اولیه به طور تصادفی تولید می کنیم. نتایج نشان می دهد، اگر از الگوریتم خوشه بندی برای به دست آوردن جواب اولیه استفاده شود، کیفیت بیشتری در جواب ها مشاهده خواهد شد.در این روند همسایگی برای جستجوی جواب، با تعویض دوعنصر از دو مجموعه با هم شکل می گیرد. طبیعتا بهترین عضو در همسایگی، متناظر با جوابی است که بیشترین کاهش هزینه را داشته باشد. در ابتدا زیر مجموعه ی $s_۱$ را به ترتیب زیر مرتب می کنیم . گوییم راس u از راس v کوچکتر است هرگاه کاهش هزینه زمانی که راس u از $s_۱$ به $s_۲$ انتقال داده می شود کمتر از کاهش هزینه زمانی که راس v از $s_۱$به $s_۲$ انتقال داده می شود، باشد. زیر مجموعه ی $s_۲$ نیز به طریق مشابه مرتب می شود. با این تعریف، برای ساختن یک حرکت، زوج تغییراتی که در بین k عنصر اول، بیشترین کاهش را درهزینه داشته باشند، در نظر گرفته می شود. همان طور که می دانید زمان مورد نیاز برای این مرتب سازی متعلق به $O(nlogn)$ دارد. پیچیدگی جستجوی ممنوع نیز در همین قسمت محاسبه می شود.در ساختن لیست ممنوع هر عضو لیست یک تغییر انجام شده در مجموعه ها را ذخیره می کند.برای مثال اگر تغییر کنونی زوج (u, v) که در آن $u \in s_۱$ و $v \in s_۲$ است، باشد در این صورت زوج تغییر (v, u) به لیست ممنوع اضافه می شوند. اما اگر یک تغییر خاص جالب باشد، حتی اگر در شرایط ممنوع صدق کند، انتخاب می شود. این قابلیت توسط تابع آزادسازی (تابع تنفس) انجام می شود. این تابع که با $A(y)$ نمایش داده می شود، یکی کمتر از هزینه بهترین پیکربندی بدست آمده از پیکر بندی با هزینه y است و به صورت زیر تعریف می شود.

if($A(f(x_{i-۱})) < f(x_i)$
$A(f(x_{i-۱}) = f(x_i)$؛۱-
else if $A(f(x_i))<f(x_{i-۱})$
$A(f(x_i)= f(x_{i-۱})$؛۱-

۷.۴ نتایج

در این جا الگوریتم KLFM و الگوریتم ارایه شده در این قسمت را ۱۰۰ بار برای گراف ها اجرا کردیم، اما این مقایسه برای الگوریتم سردکردن تدریجی دما SA فقط یک بار انجام شده است. نتایج حاصل که در جدول زیر قابل مشاهده است نشان داد که الگوریتم ارایه شده و الگوریتم KLFM تقریبا مشابه با هم عمل می کنند فقط در برخی موارد الگوریتم مورد نظر تا ۳۳ درصد در کیفیت جواب ها بهبود حاصل می کند. در مقایسه با الگوریتم سرد کردن تدریجی ، از لحاظ کیفیت جواب ها این الگوریتم بهتر نبود، اما از لحاظ سرعت اجرا در به دست آوردن جواب ها نسبت به الگوریتم SA بهتر عمل می کند.

الگوریتم ژنتیک [۸] تکنیک جستجویی در علم رایانه برای یافتن راه‌حل تقریبی برای بهینه‌سازی و مسایل جستجو است. الگوریتم ژنتیک نوع خاصی از الگوریتمهای تکامل است که از تکنیکهای زیست‌شناسی فرگشتی مانند وراثت و جهش استفاده می‌کند. در واقع الگوریتم‌های ژنتیک از اصول انتخاب طبیعی داروین برای یافتن فرمول بهینه جهت پیش‌بینی یا تطبیق الگو استفاده می‌کنند.الگوریتم‌های ژنتیک اغلب گزینه خوبی برای تکنیک‌های پیش‌بینی بر مبنای رگرسیون هستند. مختصراً گفته می‌شود که الگوریتم ژنتیک یک تکنیک برنامه‌نویسی است که از تکامل ژنتیکی به عنوان یک الگوی حل مساله استفاده می‌کند.ورودی مساله‌ای است که باید حل شود و راه‌حل ها طبق یک الگو کد گذاری می‌شوند که تابع برازندگی [۹] نام دارد هر راه حل کاندید را ارزیابی می‌کند که اکثر آنها به صورت تصادفی انتخاب می‌شوند.

پس از بیان اصطلاحات الگوریتم ژنتیک، برای حل یک مساله توسط این الگوریتم، مرحله اصلی تنظیم پارامترهای مساله است. اصولا در اولین گام برای تعیین پارامترها، یک جواب شدنی به صورت تصادفی برای مساله انتخاب می شود. در این جا به سراغ پیاده سازی الگوریتم ژنتیک برای مساله افرازبندی در گراف می رویم. یکی از مسایل مهم در محاسبات موازی، افراز کردن گراف به گروه هایی با تعداد رئوس مساوی، به طوری که تعداد یال های عبوری بین

[۸] Genetic Algorithm - GA
[۹] Fitness

هر دو افراز کمترین باشد.در این جا با استفاده از عملگر تقاطع جدید الگوریتم ژنتیکی ارایه می شود که نسبت به سایر الگوریتم های ژنتیکی که از عملگر($KNUXDKNUX$) های تقاطع کلاسیک استفاده می کنند، نتایج بهتری در کیفیت و تعداد رئوس در افراز ها می دهد.

در این الگوریتم جمعیت، مجموعه ای از جواب ها که به طور یکنواختی با زمان بر حسب کاربرد عملگرهای جهش و تقاطع عمل می کند. در این الگوریتم روند انتخاب منحصر به فردی وجود دارد که تعیین می کند کدام شخصیت ها در تولید نسل بعدی باقی بمانند. الگوریتم های ژنتیک متفاوتی برای این مساله وجود دارد که می توان به آن ها مراجعه کرد [۲۱، ۲۲، ۲۳] . عملگر تقاطع به کار رفته در این الگوریتم اطلاعات جدیدی از تاریخچه جستجوی الگوریتم در اختیار ما قرار می دهد. الگوریتم از اطلاعات قبلی برای بهبود در جواب ها استفاده می کند. همچنین در این الگوریتم، قابلیت موازی سازی با استفاده از الگوریتم های ژنتیک دیگر و کیفیت دادن به افرازهای به دست آمده با استفاده از روش های مختلف، انجام شده است. نتایجی که از این الگوریتم به دست می آید، بهتر و یا قابل مقایسه با بهترین روش های شناخته شده برای افراز بندی گراف ها است.اگرچه الگوریتم ژنتیک زمان اجرای بیشتری را نسبت به سایر الگوریتم های حریصانه می خواهد، ولی در کاربردهای عملی که کیفیت جواب های به دست آمده مهم است، این الگوریتم پیشنهاد می شود.

۱.۷.۴ نمادها

فرض کنید گراف $G = (V, E)$ با $V = |n|$ و $E = |m|$ داده شده باشد. مساله افراز بندی گراف معادل با پیدا کردن، نگاشتی است که راس ها را به افرازی می نگارد، این نگاشت را به صورت $M = v- > p$نشان می دهیم. همچنین تعریف می کنیم:

$$B(q) = v \in V; M(v) = q$$

که در آ ن $B(q)$ ، مجموعه راس هایی که به افراز

q متناظر می شود، را نمایش می دهد.

مختصات فیزیکی در فضایd به صورت زیر تعریف می کنیم:

$$X = (x_{i۱}, x_{id}).$$

همچنین برای هر راس مانند $v_i \in V$، $i \in ۱, ..., n$

$$(v_{i۱},,v_{id})$$

پس هر یال یک زوج در نظر می گیریم، است. اگر w_i هزینه محاسباتی راس v_i باشد و هزینه ارتباطات بین هر دو راس را با $W_e(v_۱,v_۲)$ نشان دهیم، وزن هر افراز را به صورت

$$w(q)=\sum_{i=۱}^{n} v_{ieB(q)}$$

ومقدار نهایی هزینه ارتباطی، که تعداد یال های خروجی از هر افراز را نشان می دهد را به صورت زیر تعریف می کنیم:

$$C(q)=\sum_{i=۱}^{n} v_{ieB(q)}$$
$$v_{ie}=B(q)W_{e(v_i,v_j)}$$

هدف در این جا این است که هزینه نهایی برای هر گره را مینیمم کنیم. اگر B هزینه ارتباطی برای به کارگیری هر واحد بر روی سیستم باشد، تابع هدف را به صورت زیر می نویسیم:

$$\sum_q W(q)+b\sum_q \max C(q)$$

در این صورت روی مینیم کردن هزینه ارتباطی برای بدترین افراز متمرکز می شویم. این بدین معنی است که تابع هدف زیر را مینیم کنیم:

$$\sum_q W(q)+b\sum_q \max C(q)$$

۲.۷.۴ تنظیم پارامترها برای مساله افراز بندی در گراف ها

یک جواب تصادفی در ابتدا تولید می کنیم و آن را به عنوان جمعیت اولیه برای الگوریتم در نظر می گیریم. اگر هر راس را به عنوان یک ژن و برای هر افراز یک کروموزوم در نظر بگیریم، با استفاده از کدگذاری دودویی هر راس را به یک افراز تخصیص می دهیم. به عنوان نمونه کد ۱۱۱۰۰۰۱۱ به این معنی است که در آن راس های ۱،۲،۳،۷ و ۸ متناظر با یک و راس های ۴،۵و ۶ متناظر با صفر اند. در مرحله تقاطع جوابهایی به دست می آیند که از بین آنها بایدجواب هایی برای تکثیر انتخاب شود. تابع برازندگی معیاری است که به ما کمک می کند تا ببینیم کدام یک از فرزندان برای تکثر در جامعه انتخاب می شوند. در این الگوریتم از یک تابع برازندگی که تقریبی از تابع هدف مساله است، به صورت زیر استفاده می کنیم:

$$\sum_q(|B(q)|-|V|/n)^۲-b\sum_q C(q)$$

با این تعریف هر کروموزومی که برازندگی بیشتری دارد با احتمال بیشتری در جمعیت جدید باقی می ماند. اما فرزندان در این الگوریتم چگونه ساخته می شوند؟ در مقاله های مختلف عملگرهای مانند عملگر تقاطع یک نقطه ای و یا عملگر تقاطع K نقطه ای استفاده می شود، که می توانید در صورت تمایل به آنها مراجعه کنید. [۴] ایده ای که در اینجا از آن استفاده می کنیم ، به کارگیری عملگر تقاطع غیر یکنواخت (KNUX) در الگوریتم است که از یک بردار احتمالی(P_i) استفاده می کند و به صورت زیر تعریف می شود:

$$P = (p_۱, ..., p_n) p_i \in [۰, ۱]$$

حال تعریف می کنیم $v(i)$ مجموعه ی همسایگی های راس i در گراف مورد نظر و (i, X, I) را برابر با تعداد راس هایی در $v(i)$ در نظر می گیریم که توسط I به افراز X_i برده می شود. اکنون اگر a،b دو والد باشند ، فرزند $C = (c_۱, ..., c_n)$ به صورت زیر ساخته می شود:

اگر برای هر مولفه، a_i برابر با b_i باشد، آنگاه $c_i i$ مقدار a_i را می گیرد، و در غیر این صورت مولفه ی iام در C با احتمال p_i مقدار a_i را می گیرد که این احتمال با استفاده از رابطه زیر محاسبه می شود.

$$p_i =/۰۵ \quad if \quad (i, a, I) + (i, b, I) = ۰$$
$$p_i = \frac{(i,a,I)}{(i,b,I)} \quad O.W$$

بیان کردیم که در عمل تقاطع فرزند خصوصیات خود را از یکی از والدین دریافت میکند، اما ممکن است زمانی که ژن ها با یکدیگر ترکیب می شوند، تحت عواملی یکی از ژن ها بدون اینکه از والدی آمده باشد یک جهش پیدا کند و نتیجه در فرزند خلاف آنچه باشد که انتظار می رود. این عمل به عنوان جهش در ژنتیک مورد بحث قرار می گیرد. جهش باعث می شود اطلاعات جدیدتری از جامعه به ما داده شود. در اینجا عمل جهش در صورت نیاز با یک تغییر تصادفی صفرها به یک و بالعکس انجام می شود.

کیفیت جواب های بدست آمده توسط الگوریتم وابسته به انتخاب جواب اولیه ای است که دراین جا با I نمایش داده شد. لذا اگر این جواب اولیه بهتر انتخاب شود، جواب های بعدی از کیفیت بهتری برخوردارند. توجه کنید که جمعیت اولیه می تواند با یک جواب ابتکاری برآورد شود.

۳.۷.۴ نتایج تجربی

در این بخش با نتایج به دست آمده از الگوریتم، به مقایسه آن با ابتکارهای کلاسیک و الگوریتم هایی که از عملگر تقاطع کلاسیک استفاده می کنند، می پردازیم.دراین آزمایشات نرخ تقاطع p_c برابر ۰.۷ و نرخ جهش p_mبرابر با ۰.۰۱ در نظر گرفته شده است. در جدول ؟؟ همان گونه که مشاهده می کنید، از مقایسه این الگوریتم با الگوریتم ابتکاری

RSB که در [۲۴] قابل دسترس است ، نتیجه به دست آمده این که بدترین اندازه برش بدست آمده در الگوریتم ارایه شده بهتر از بدترین اندازه برش بدست آمده در الگوریتم RSB است.

کتاب‌نامه

[1] W. W. Hager and D. T. Phan, An ellipsoidal branch and bound algorithm for global optimization, SIAM J. Optim., pp., 20, 740–758, 2009.

[2] J. Wiley, D. Sons,Theory of linear and integer programming,1998.

[3] B. Kernighan and S. Lin, An efficient heuristic procedure for partitioning graphs, Bell systems technical J., 49, 291-307, 1970.

[4] R. Battiti and A. Bertossi, Diifferential greedy for the 0-1 equicut problem, In proceeding of the DIMACS workshop on network design: connectivity and facility location, 3-22, 1997.

[5] R. Battiti and A. Bertossi, Greedy and prohibition based heuristics for graph partitioning: UTM 512 Dip. matematica: A Mathematical, univ. di Trento. Italy. February, 1997.

[6] S. Jain, C. Swany, K. Balaji, Greedy algorithm for k-way graph partitioning, 1990.

[7] U. Feige, M. Langberg, The RPR2 rounding technique for semidefinite programs, Proceedings of the 33th Annual ACM Symposium on Theory of Computing, Crete, Greece, pp. 213–224, 2001.

[8] U. Feige, M. Langberg, Approximation Algorithms for Maximization Problems arising in Graph Partitioning, Journal of Algorithms, 41, pp. 174-211, 2001.

[9] Q. Han, Y. Ye, J. Zhang, An Improved Rounding Method and Semidefinite Programming Relaxation for Graph Partition, Mathematical Programming, 92 (3), pp. 509–535, 2002.

[10] E. Halperin, U. Zwick, A unified framework for obtaining improved approximation algorithms for maximum graph bisection problems, Random Structures and Algorithms, 20 (3), pp. 382-402, 2002.

[11] G. Jager, A. Srivastav, Improved Approximation Algorithms for Maximum Graph Partitioning Problems, 2001.

[12] F. Glover , Future paths for integer programming and links to artificial intelligence, Computer and Operations Research 13, 533-549, 1986.

[13] F. Glover and H.J. Greenberg, New approaches for heuristic search: a bilateral linkage with artificial intelligence, European Journal of Operational Research 39,pp, 119-130, 1989.

[14] F. Glover, Tabu search - part I, ORSA Journal on Computing 1, pp, 190-206,1989.

[15] B.W. Kemighan and S. Lin , An efficient heuristic procedure for partitioning graphs, Bell System Technical Journal 49,pp, 291-307, 1970.

[16] C.M. Fiduccia and R.M. Mattheyses, A linear-time heuristic for improving network partitions, Proceedings of the 19th Design Automation Conference (ACM Press), pp, 175-1982, 1982.

[17] T.K. Ng, J. Oldfield and V. Pitchumami (), Improvement of a mincut partition algorithm, Proceedings of the International Conference on Computer-Aided Design (ACM Press) pp. 470-473, 1987.

[18] B. Krishnanmurthy, An improved mincut algorithm for partitioning of VLSI networks, IEEE Transaction on Computers 33,pp. 438-446, 1984.

[19] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, Optimization by simulated annealing, Science 220, 67 1-680, 1983.

[20] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness (Freeman, San Francisco, CA), 1979.

[21] G. J. cohoon, W. N. martin, D. S. richards, A multi-population genetic algorithm for solving the k-partitioning problem on hypercubes, Proc .4th ICGA, pp. 244-248, 1991.

[22] D. R. jones, M. A. beltramo, Solving partitioning problem with genetic algorithms, Proc .4th ICGA, pp. 442-450, 1991.

[23] G. von laszeweski, Intelligent structural operators for the k-way graph partitioning problem, Proc .4th ICGA, pp. 45-52, 1991.

[24] R. collins, D. jefferson, Selection in massively parallel genetic algorithms, Proc .4th ICGA, pp. 249-256, 1991.