داده‌پراکنی جزیی و شبه تطابق

توسط

محمدتقی حاجی آقایی

رساله ارائه شده به عنوان بخشی از ملزومات برای دریافت درجه

کارشناسی نرم‌افزار کامپیوتر

زیرنظر

دکتر محمد قدسی

تیرماه ۱۳۷۹

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

تهران

قدردانی

در این‌جا لازم است از کلیه‌ی افرادی که مرا در انجام این پروژه کمک نموده‌اند، خصوصاً استاد گرامی آقای دکتر محمد قدسی و دوستانم سید وهاب میررکنی و محمد مهدیان که در تمام مراحل انجام این پروژه با مساعدت‌ها و راهنمایی‌های بی‌دریغ خود مرا یاری کردند، تشکر کنم.

این پایان‌نامه با استفاده از سیستم حروفچینی فارسی‌تک تهیه شده است که شایسته می‌دانم از زحمات گروه تولید کننده‌ی این نرم‌افزار نیز تقدیر به عمل بیاورم.

داده‌پراکنی جزیی و شبه تطابق

چکیده

در این پایان‌نامه مساله داده پراکنی یک به چند مورد بررسی قرار می‌گیرد که در آن یک نود شبکه قطعه‌ی اطلاعاتی مشابهی را به یک یا چند نود دیگر در شبکه می‌فرستد و این عمل کاربردهای بسیاری در محاسبات موازی و توزیع شده دارد. در اینجا این مساله در مدل خطی بررسی می‌شود که فرض می‌کند تماسهای طویل بین پردازنده‌های غیر مجاور صورت می‌گیرد.

این موضوع که در مدل خطی می‌توان بایک الگوریتم چندجمله‌ای در $\lceil \log_2(d) \rceil$ مرحله (d تعداد پردازنده‌های مقصد است) عمل داده‌پراکنی را انجام داد، قبلاً ثابت شده است ولی این الگوریتم از منابع شبکه بصورت متعادل استفاده نمی‌کند. در این پایان‌نامه ما ضمن ارائه الگوریتمی با کارائی بهتر، نشان خواهیم داد که اکثر مسایل بهینه‌سازی داده پراکنی جزیی نظیر بهینه کردن طول مسیر، بار رئوس و زمان تعویض، در رده NP-complete قرار می‌گیرند. هم‌چنین بعد از مدل کردن این شبکه با گراف، بسیاری از این مسائل تبدیل به مساله‌های شبه تطابق می‌شوند که یک مجموعه از مسیرهای مجزای یالی هستند که جفتهای متمایزی از رئوس را به هم متصل می‌سازند.

در این پایان‌نامه یک الگوریتم تقریبی برای نوع خاصی از مساله شبه تطابق و الگوریتم پیداکردن تطابق بیشینه درگرافهای کلی ارائه شده است. در ضمن حالتهای خاصی از مساله در ابرمکعب مورد بررسی قرار گرفته است .

واژه‌های کلیدی: داده‌پراکنی کلی، داده‌پراکنی جزیی، تطابق، شبه‌تطابق، مسیرهای مجزای یالی (راسی)، f—فاکتور، الگوریتم‌های تقریبی

فهرست

۱ مقدمه ۴

۲ تطابق و الگوریتم‌های مربوطه ۷

۱.۲ مقدمه ۷

۲.۲ شرط تطابق هال ۸

۳.۲ الگوریتم پیدا کردن تطابق دوبخشی ماکزیمم ۹

۴.۲ تطابق در حالت کلی ۱۲
۱.۴.۲ قضیه ۱—عامل تات ۱۳
۲.۴.۲ f—عامل‌ها در گراف ۱۴

۵.۲ الگوریتم شکوفه ادموند (برای یافتن تطابق در گراف کلی) ۱۴

۶.۲ ورودی و خروجی ۱۸

۳ تاریخچه‌ی شکل‌گیری مسایل شبه‌تطابق ۲۰

۱.۳ مقدمه ۲۰

۲.۳ یک پروتکل کلی برای توابع مسیردهی مینیمال ۲۵

۳.۳ قضایای مربوط به کمینه کردن بار، تعداد مسیریاب‌ها و زمان تعویض ۳۱

۴.۳ نتیجه‌گیری و مسایل باز ۳۲
۱.۴.۳ مدل تمام‌پورت ۳۳
۲.۴.۳ مساله شایعه پراکنی ۳۳
۳.۴.۳ بررسی این مدل در گراف‌های جهت‌دار ۳۳

۴ نتایج جدید در مورد شبه‌تطابق ۳۴

۱.۴ مقدمه ۳۴

۲.۴ مسئله‌ی شبه‌تطابق در حالت $U = V(G)$ ۳۵

۳.۴ الگوریتم‌هایی برای حالاتی از مسئله‌ی شبه‌تطابق ۳۶

۴.۴ NP-Complete بودن شبه‌تطابق محدود ۴۰

۵.۴ مسایل حل‌نشده ۴۵

۵ خلاصه‌ای از دیگر کارهای انجام‌شده ۴۶

۱.۵ شبه‌تطابق در گراف‌ها ۴۶
۱.۱.۵ شبه‌تطابق‌های کامل ۴۸
۲.۱.۵ شبه‌تطابق‌های کامل دوبخشی ۵۱

۲.۵ مسیریابی چندرنگی در ابرمکعب بدون‌جهت ۵۳
۱.۲.۵ ایده‌هایی در مورد اثبات قضیه ۵۶

۳.۵ NP-complete بودن بعضی مسایل مسیرهای مجزای یالی ۵۷
۱.۳.۵ مسایل اویلری ۵۸
۲.۳.۵ مسایل مسطح ۶۰

۴.۵ داده‌پراکنی جزیی در چندپردازنده‌های ابرمکعب ۶۱
۱.۴.۵ مدل درخت داده‌پراکنی جزیی بهینه ۶۲
۲.۴.۵ پیداکردن OMT در یک ابرمکعب ۶۴
۳.۴.۵ الگوریتم مکاشفه‌ای برای داده‌پراکنی جزیی ۶۵

۵.۵ بیشترین تعداد مسیرهای مجزا متصل کننده‌ی تعداد مشخصی پایانه ۶۵

۶.۵ مسایل تطابق‌مسیری ۶۶

۷.۵ زمانبندی تماس‌ها برای داده‌پراکنی جزیی در شبکه‌های درخت ۶۷

۶ الگوریتم ۱—تقریبی برای مساله‌ی شبه‌تطابق تنگنایی ۶۸

۱.۶ مقدمه ۶۸

۲.۶ بعضی نتایج مقدماتی ۶۹

۳.۶ پیاده‌سازی الگوریتم ۷۵

۴.۶ ورودی و خروجی ۷۷

۷ خلاصه و نتیجه‌گیری ۷۹

A متن برنامه‌ی اول ۸۱

B متن برنامه‌ی دوم ۹۰

C واژه‌نامه‌ی فارسی به انگلیسی ۹۹

D واژه‌نامه‌ی انگلیسی به فارسی ۱۰۱

فهرست شکل‌ها

۱.۲ مثالی برای درک بهتر الگوریتم مسیر افزایشی در گراف‌های دوبخشی. ۱۰

۲.۲ مثالی برای درک بهتر قضیه‌ی تات. ۱۳

۳.۲ مثالی برای درک بهتر مسیر افزایشی با وجود دور فرد در گراف. ۱۵

۴.۲ یک نمونه از فایل خروجی برنامه تطابق در حالت کلی. ۱۹

۱.۴ گراف ساخته‌شده ۴۲

۱.۵ زیرگراف استفاده شده ۶۰

۱.۶ یک گراف وزندار G (A) و درخت پوشای کمینه آن (B). ۷۰

۲.۶ جنگل پوشای ریشه‌دار F که از T کل باحذف یال‌های E_T حاصل شده است. . ۷۳

۳.۶ یک نمونه از فایل خروجی برنامه تقریبی ۷۸

فصل ۱

مقدمه

هدف اصلی از این پایان‌نامه بررسی داده‌پراکنی جزیی [۱] وارتباط آن با شبه‌تطابق [۲] می‌باشد. داده‌پراکنی جزیی عبارت است از رساندن پیغام یک گره در شبکه به مجموعه مشخص‌شده‌ای از گره‌های دیگر شبکه. این شبکه می‌تواند توپولوژی‌های مختلفی نظیر: درخت، ستاره، ابرمکعب، مش و ... داشته باشد. داده‌پراکنی جزیی دارای کاربردهای فراوان و یکی از اصلی‌ترین توابع اجرایی در هر زبان موازی ویا توزیع شده است. در واقع با این عمل است که یک سری پردازنده از اطلاعات یک پردازنده خاص مطلع شده ویا باآن ارتباط برقرار می‌کنند. برای پیاده‌سازی ارتباط بین دو پردازنده که مستقیم با هم ارتباط دارند تکنیک‌های بسیاری نظیر حافظه مشترک وجود دارد که در این‌جا ما قصد پرداختن به آن را نداریم، بلکه فقط ارتباط این دو راس بعنوان دو گره از شبکه و به‌صورت مجرد مورد نظر این پایان‌نامه می‌باشد. حال اگر دو پردازنده به‌صورت مستقیم با هم ارتباط نداشته باشند باید از یک پروتکل ارتباطی بین آنها استفاده کرد. برای مجردکردن بحث، شبکه را به‌صورت یک گراف بدون جهت مدل می‌کنیم و لذا ارتباط بین دو پردازنده غیرمجاور همانند وجود یک مسیر در گراف می‌شود.

حال بحث داده‌پراکنی جزیی به‌صورت کاملاً مجرد در گراف مطرح می‌شود. این بحث ابتدا ما را به سمت اثبات قضایایی در زمینه داده‌پراکنی و بهینه‌سازی ارتباط می‌کشاند. آنگاه بحث به

[۱] Multicasting

[۲] Pseudomatching

سمت الگوریتم‌های تقریبی[۳] و مکاشفه‌ای [۴] پیش خواهد رفت که می‌توانند به‌صورت درون‌خطی [۵] یا برون‌خطی [۶] محاسبه شوند و بامطرح‌شدن این الگوریتم‌هاست که بار دیگر از نظری به عمل روی آورده و الگوریتم‌هایی قابل پیاده‌سازی ارایه می‌دهیم.

ساختار این پایان‌نامه به‌صورت زیر است. در فصل ۲ به بحث در مورد تطابق و الگوریتم‌های مربوط به آن می‌پردازیم. در آن‌جا شرط تات[۷] برای وجود یک تطابق کامل در گراف بررسی شده و الگوریتم شکوفه‌ی ادموند [۸] برای پیداکردن یک تطابق کامل در گراف مورد تجزیه و تحلیل قرار می‌گیرد. در فصل ۳ به بحث در مورد تاریخچه داده‌پراکنی جزیی و تعریف مفهومی بنام شبه‌تطابق که در واقع مجردشده‌ی ارتباطات موازی بین پردازنده‌هاست می‌پردازیم. در آن‌جا هم‌چنین به پیچیدگی [۹] مسایل بهینه‌سازی در مورد داده‌پراکنی اشاره‌ای خواهیم داشت و به اثبات NP-complete بودن اکثر این مسایل خواهیم پرداخت. در فصل ۴ باز هم در زمینه‌ی مفهوم مجردشده‌ی شبه‌تطابق و مسایل نظری که البته جنبه‌ی عملی هم دارند، بحث بیشتری خواهیم کرد. در این‌جاست که به مرز مسایل باز بسیاری می‌رسیم که البته حدس‌هایی هم در مورد آن‌ها وجود دارد. در فصل ۵ که کاملاً نظری است به تعمیم مساله شبه‌تطابق و بررسی آن از جنبه‌های مختلف و هم‌چنین تاریخچه‌ی کارهایی که تاکنون روی آن انجام شده می‌پردازیم. در فصل ٦ به جنبه‌های عملی پرداخته و سعی می‌کنیم با ارایه‌ی یک الگوریتم ۱ —تقریبی برای یک حالت خاص از مساله‌ی شبه‌تطابق و هم‌چنین پیاده‌سازی آن، راه را برای استفاده جنبه‌های نظری در کارهای عملی بازکنیم. سرانجام در فصل ۷ که آخرین فصل این پایان‌نامه است به نتیجه‌گیری این بحث پرداخته و مسایلی را که می‌توان روی آن‌ها تحقیق بیشتری نمود و هم‌چنین تعمیم‌هایی که در مورد این بحث وجود دارد ارایه می‌کنیم، که خواننده می‌تواند از آن برای کارهای بعدی استفاده کند. پس از آن در پیوست‌های A و B، متن برنامه‌های نوشته شده برای این پروژه خواهد آمد. پیوست‌های C و D نیز حاوی بعضی اصطلاحات تخصصی به همراه ترجمه‌ی

[۳] Approximation algorithms
[۴] Heuristic algorithms
[۵] On-line
[٦] Off-line
[۷] Tutte
[۸] Edmonds' blossom algorithm
[۹] Complexity

آن‌ها هستند که در این پایان‌نامه به‌کار رفته است.

فصل ۲

تطابق و الگوریتم‌های مربوطه

۱.۲ مقدمه

در فصول بعدی شبه‌تطابق را به‌صورت دقیق تعریف می‌کنیم. سپس برای به‌دست آوردن الگوریتم‌های مربوط به بعضی از گونه‌های شبه‌تطابق، تبدیل آن مسایل به پیداکردن تطابق کامل در گراف مورد بحث و بررسی قرار می‌گیرد. در این فصل در مورد تطابق درگراف‌های کامل بحث کرده، الگوریتم‌هایی را در این زمینه ارایه کرده و در انتها پیاده‌سازی آن‌ها را توضیح می‌دهیم. در تمامی این فصل گراف را بدون جهت فرض می‌کنیم. با توجه به این فرض، تطابق درگراف‌های کلی در دو حالت گراف وزندار و بدون وزن قابل بررسی است که در این‌جا به حالت بدون وزن می‌پردازیم.

تعریف ۱. یک تطابق در گراف بدون جهت G، مجموعه‌ای از یال‌های مجزا می‌باشد. رئوس متعلق به یال‌های تطابق را آلوده‌شده توسط تطابق و بقیه رئوس را آلوده‌نشده توسط تطابق می‌نامند. اگر یک تطابق تمامی رئوس گراف G را آلوده کند آن‌را تطابق کامل می‌نامیم.

مساله‌ای که در این‌جا مطرح می‌شود، مساله پیداکردن یک تطابق درگراف است. ولی ازآن‌جا که لزوماً هرگراف یک تطابق کامل ندارد مساله پیداکردن یک تطابق بیشینه یا تطابقی که بیشترین تعداد یال راداشته باشد، مطرح می‌شود.

تعریف ۲ . یک تطابق M داده شده است. یک $M-$مسیر تناوبی[۱] یک مسیر در گراف است که به‌تناوب بین یال‌های M و یال‌های خارج از M می‌گذرد. یک $M-$مسیر تناوبی P که رئوس ابتدایی و انتهایی آن توسط M آلوده نشده‌اند یک $M-$مسیر افزایشی[۲] نامیده‌می‌شود. با تعویض $M \cap E(P)$ با $E(P) - M$، یک تطابق جدید M' حاصل می‌شود که یک یال بیشتر از M دارد.

تطابق بیشینه آن تطابقی است که هیچ مسیر افزایشی ندارد. ما می‌توانیم این مطلب را با بررسی زیرگرافی که از اجتماع یال‌های دو تطابق ذکر شده در تعریف ۲ و حذف یال‌های مشترک آن‌ها حاصل شده است، اثبات کنیم[٦] که در این‌جا از ذکر آن خودداری می‌کنیم.

۲.۲ شرط تطابق هال

مثالی را درنظر بگیرید که در آن شغل‌ها و افراد بیکار، تعداد شغل و تعدادی فرد بیکار که خواهان بعضی از این شغل‌ها هستند و تعدادشان می‌تواند بیشتراز شغل‌ها باشد، وجوددارند. هدف به‌دست آوردن یک تطابق ماکزیمم درگراف دو بخشی است که یک بخش آن X شامل شغل‌ها و بخش دیگر آن Y شامل افراد بی‌کار است. فرض کنید که S یک زیرمجموعه از رئوس G باشد ما علامت $N(S)$ را برای نشان دادن مجموعه‌ای ازرئوس G که یک همسایه در S دارند بکارمی‌بریم. حال قضیه زیر را داریم:

قضیه ۱ . (P. Hall[1935]) اگر G یک گراف دوبخشی با دو بخش Y و X باشد، آن‌گاه G یک تطابق از X به Y دارد که تمام رئوس X را می‌پوشاند اگر و تنها اگر شرط $|N(S)| \geq |S|$ برای تمام زیرمجموعه‌های $S \subseteq X$ درست باشد[۸].

این قضیه را به روش‌های مختلفی می‌توان اثبات کرد، ولی یکی از روش‌های جالب آن، استفاده از روش حالتی بحرانی است. ما اثبات این قضیه را صرفاً به‌منظور معرفی این روش و به‌خاطر

[۱] M-alternating path

[۲] M-augmenting path

این‌که این روش، یک روش کلی برای حل مسایل مشابه است و می‌تواند بعداً در اثبات مسایل شبه‌تطابق به‌کار رود، توضیح می‌دهیم:

اثبات. لزوم این شرط واضح است. برای اثبات کافی بودن آن از استقرا روی X استفاده می‌کنیم. حالت $|X| = ۱$ بدیهی است. لذا فرض می‌کنیم که قضیه برای تمام $|X| < h$ درست باشد و حالت $|X| = h$ را بررسی می‌کنیم. دو حالت باید در نظر گرفته شود.

۱) برای هرمجموعه k عضوی S $(k < h)$ از X ، $N(S)$ حداقل $k + ۱$ عضو دارد. در این‌صورت یکی از اعضای X مانند $x_۰$ را با یکی از همسایه‌هایش در Y مانند $y_۰$ جفت‌کرده و حال از استقرا روی بقیه رئوس X و Y استفاده می‌کنیم. چون از همسایه‌های هر عضو باقی‌مانده در X حداکثر یک عضو کم شده است و فرض کردیم که هر مجموعه kعضوی حداقل $k + ۱$ همسایه دارد، حال شرایط استقرا برای گراف باقی‌مانده برقرار است.

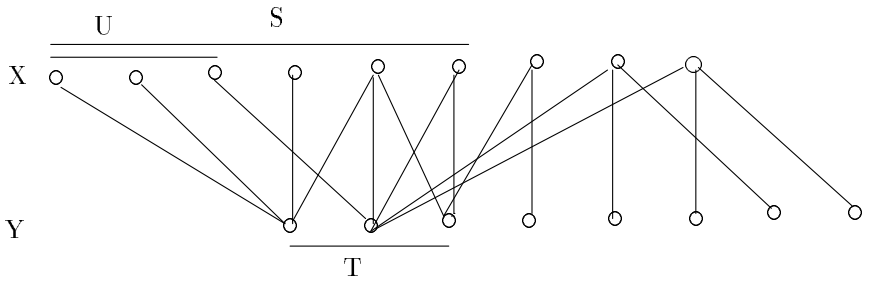
۲) برای یک $l < h$، یک مجموعه S که دارای l عضو است وجود دارد به‌طوری‌که $|N(S)| = |S|$ باشد. در این‌صورت برای مجموعه‌ی دوبخشی $(S, N(S))$ برطبق فرض استقرا یک تطابق کامل از S به $N(S)$ وجود دارد. در مورد بقیه‌ی گراف هم، به‌راحتی برای هر مجموعه S' از $X - S$ با در نظر گرفتن $S' \cup S$ و استفاده از فرض مساله می‌توان ثابت کرد که $|N(S')| \geq |S'|$ است و لذا بقیه‌ی رئوس هم تطابق خواسته شده را دارند و اثبات تکمیل می‌شود.

□

برای مطالعه‌ی بیشتر درمورد اثبات‌هایی از این‌دست، می‌توانید به [۱۰] و [۲]مراجعه نمایید.

۳.۲ الگوریتم پیدا کردن تطابق دوبخشی ماکزیمم

همان‌گونه که در بخش قبل ملاحظه شد، اثبات‌های حالتی بحرانی صرفاً اثبات‌های وجودی هستند و شرایط لازم و کافی برای یک مساله را ارایه می‌دهند، ولی درمورد چگونگی به‌دست آوردن جواب، حرفی نمی‌زنند. لذا امکان دارد که ما برای یک مساله NP-complete یک شرط لازم و کافی از این



شکل ۱.۲: مثالی برای درک بهتر الگوریتم مسیر افزایشی در گراف‌های دوبخشی.

طریق بیابیم، ولی امکان به‌دست آوردن یک الگوریتم چندجمله‌ای را نداشته باشیم. مثلاً در مثال قبل چک کردن شرط وجود یک تطابق کامل از Y به X از درجه نمایی است ولی این، نقصان ما را از حمله به مسایلی از این دست باز نمی‌دارد و می‌توان سعی نمود که از این روش در به‌دست آوردن شرایط لازم و کافی در مورد مسایل شبه‌تطابق استفاده شود.

خاصیت مسیر افزایشی یک تطابق بیشینه ما را به‌سمت یک الگوریتم برای پیداکردن تطابق بیشینه رهنمون می‌شود ما مکرراً به دنبال مسیرهای افزایشی می‌گردیم تا هر بار یکی به تعداد یال‌های تطابق موجود اضافه شود. در گراف دوبخشی اگر ما نتوانیم یک مسیر افزایشی بیابیم می‌توان ثابت کرد که تطابق بیشینه را یافته‌ایم. این روش یک الگوریتم برای گراف‌های دوبخشی به‌ما می‌دهد.

یک تطابق در گراف دوبخشی G داده شده است. ما می‌توانیم به‌دنبال یک M—مسیرافزایشی از یک راس آلوده نشده توسط M، بگردیم. فقط لازم است که رئوس آلوده نشده در یک بخش گراف دوبخشی رامورد بررسی قرار دهیم، چون هر مسیر افزایشی یک انتها در هر بخش دارد. مامی‌توانیم از تمام رئوس آلوده نشده در یک بخش به‌طور هم‌زمان با حفظ اطلاعات به‌دست آمده در حین عمل جستجو، به‌دنبال مسیر افزایشی بگردیم. در ابتدا با یک تطابق به اندازه‌ی صفر شروع کرده و پس از تعداد متناهی تکرار به تطابق بیشینه می‌رسیم.

الگوریتم ۱ (الگوریتم مسیرافزایشی)

- **ورودی:** یک گراف دوبخشی G با دوبخش X و Y، یک تطابق M در G و یک مجموعه U از تمام رئوس آلوده نشده توسط M در X داده شده است.
- **خروجی:** یک تطابق بیشینه
- **ایده:** $M-$مسیرهای افزایشی از U را کشف کن، بدین ترتیب که $S \subseteq X$ و $T \subseteq Y$ را رئوس دست‌یافته از U درنظر بگیر. رئوسی از S را که برای توسعه‌ی مسیرها مورد بررسی قرار گرفته‌اند را علامت بزن. برای هر $x \in (S \cup T) - U$ راس قبل از x روی یکی از مسیرهای افزایش از U را ثبت کن.
- **مقدار دهی اولیه:** $S = U$ و $T = \emptyset$
- **تکرار:** اگر S هیچ راس علامت نخورده‌ای ندارد، الگوریتم را خاتمه بده و M را به‌عنوان تطابق بیشینه گزارش کن. درغیر این‌صورت یک راس علامت نخورده مانند $x \in S$ را انتخاب کن. برای مکاشفه از x، هر $y \in N(x)$ را که $xy \notin M$ است، درنظربگیر. اگر y آلوده نشده است، مکاشفه را خاتمه داده و از y به عقب برگردد تایک M-مسیرافزایشی از U به y را گزارش دهی. در غیر این‌صورت، y به یک راس $w \in X$ توسط M جفت شده است؛ دراین صورت y را در T (دست‌یافته از x) و w را در S (دست یافته از y) داخل کن. بعد از مکاشفه از تمام یال‌های مجاور x، x راعلامت بزن و این عمل را تکرار کن.

در شکل ۱.۲ مجموعه‌های تعریف شده در الگوریتم بالا نشان داده شده‌اند. در این شکل یال‌های تطابق با خطوط عمودی و بقیه یال‌ها با خطوط اریب نمایش داده شده‌اند.

لازم به‌ذکر است که هنگامی که ما از x (در مرحله تکرار) مکاشفه می‌کنیم، امکان دارد که y در T و w در S قبلاً موجود باشند. توضیح بیشتر این الگوریتم را در قسمت‌های بعدی همین فصل خواهید

دید.

قضیه ۲ . الگوریتم مسیر افزایشی در یک گراف دو بخشی یک تطابق بیشنه ایجاد می کند[۴].

با محاسبه پیچیدگی می‌توان به این نتیجه رسید که این الگوریتم از $O(n^۳)$ می‌باشد که در آن n تعداد رئوس گراف دوبخشی می‌باشد. الگوریتم‌های سریع‌تری هم برای این مساله وجود دارند (مثلاً از $O(n^{۲.۵})$) که به‌دلیل پیچیدگی، از ذکرشان خودداری می‌کنیم [٦].

تطابق‌های دیگری مانند تطابق سازگار نیز تعریف شده‌اند که الگوریتم‌های چندجمله‌ای برای آن‌ها وجود دارد [٦] و [۸] ولی به‌دلیل ارتباط کم آن‌ها با موضوع این پروژه از ذکر آن‌ها نیز خودداری می‌نماییم.

۴.۲ تطابق در حالت کلی

در این بخش در مورد تطابق در حالت کلی بحث می‌کنیم. برای توضیح بیشتر در این زمینه به [٦] رجوع کنید.

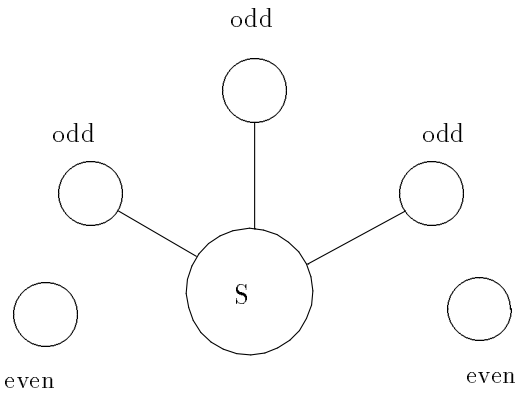
تعریف ۳ . یک عامل[۳] از G، یک زیر گراف پوشا[۴] از G می‌باشد. یک $k-$عامل یک زیر گراف $k-$منظم پوشااست(یک تطابق کامل یک ۱−عامل است). یک مولفه‌ی[۵] فرد از یک گراف، یک مولفه با تعدادی فرد راس می‌باشد. تعداد مولفه‌های فرد H را با $o(H)$ نشان می دهیم.

---

[۳] Factor

[۴] Spanning subgraph

[۵] Component



شکل ۲.۲: مثالی برای درک بهتر قضیه‌ی تات.

۱.۴.۲ قضیه ۱–عامل تات

تات[۶] یک شرط لازم و کافی برای هر گراف دل‌خواه که یک ۱–عامل (تطابق کامل) داشته‌باشد، پیدا کرد. اگر G یک ۱–عامل داشته باشد و یک زیر مجموعه $S \subseteq V(G)$ را در نظر بگیریم، آن‌گاه هر مولفه‌ی فرد $G - S$ باید یک راس داشته باشد که به بیرون آن مولفه جفت شده باشد و ناچاراً فقط می‌تواند متعلق به S باشد(شکل ۲.۲ را ببینید). چون این رئوس باید رئوس متمایز از S باشند، نتیجه می‌گیریم که باید $|o(G-S)| \leq |S|$ باشد. تات ثابت کرد که این شرط لازم واضح، شرط کافی نیز می‌باشد.

قضیه ۳. (Tutte [1947]) گراف G یک ۱–عامل دارد اگر وتنها اگر شرط $|o(G-S)| \leq |S|$ برای هر $S \subseteq V(G)$ برقرار باشد[۸].

به‌علت پیچیدگی، از اثبات این قضیه خودداری کردیم ولی در مورد اثبات این قضیه نیز از روش حالتی بحرانی تلاش‌های زیادی انجام گرفته است.

---

[۶] Tutte

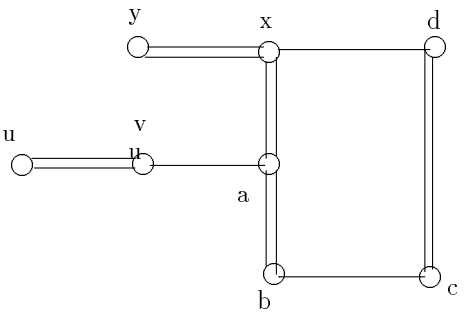
۲.۴.۲ f–عامل‌ها در گراف

یک عامل یک زیر گراف پوشای دل‌خواه از G می‌باشد. یک k–عامل، یک عامل k–منظم است. در حالت کلی‌تر ما یک تابع $f : V(G) \to N$ می‌دهیم و می‌خواهیم بدانیم که آیا G یک زیرگراف پوشای H دارد به‌طوریکه شرط $d_H(v) = f(v)$ برای هر $v \in V(G)$ برقرار باشد. این زیرگراف H یک f–عامل از G نامیده می‌شود. تات ابتدا یک شرط لازم و کافی برای وجود یک f–عامل در گراف ارایه داد. اثبات اولیه بسیار مشکل بود، ولی بعداً تات آن‌را به چک‌کردن وجود یک ۱–عامل در گراف تقلیل داد. ایده‌ی این روش به‌دین صورت است که ما به‌جای هر راس، یک گراف دوبخشی که یک بخش آن $d_G(v)$ و بخش دیگر $f(v)$ راس دارد، گذاشته و بعد با اتصال مناسب این گراف‌های دو بخشی به‌هم، وجود f–عامل را تبدیل به‌وجود ۱– عامل در گراف ساخته شده‌ی جاری می‌کنیم. برای توضیحات بیشتر به [٦] مراجعه کنید.

۵.۲ الگوریتم شکوفه ادموند (برای یافتن تطابق در گراف کلی)

در این بخش به ارایه یک الگوریتم مشهور برای یافتن تطابق در گراف کلی می‌پردازیم و نکات مربوط به پیاده‌سازی آن‌را می‌آوریم. کد برنامه پیاده‌سازی شده را می‌توانید در پیوست A بیابید. از این الگوریتم بعداً در جاهایی که مسایل شبه‌تطابق را به مسایل تطابق تبدیل می‌کنیم، استفاده می‌شود.

همانگونه که قبلاً گفته شد، یک تطابق M یک تطابق ماکزیمم است اگر و فقط اگر هیچ مسیر افزایشی نداشته باشد. لذا ما می‌توانیم یک تطابق بیشینه را با تکرار پیدا کردن مسیرهای افزایشی بیابیم. چون ما حداکثر $n/2$ بار می‌توانیم این کار را انجام دهیم، لذااگر بتوانیم جستجو برای پیداکردن مسیر افزایشی را به‌صورت بهینه انجام دهیم، می‌توانیم یک الگوریتم خوب برای یافتن تطابق کلی بیابیم. ادموند[۱۹٦۵] اولین الگوریتم را در مقاله‌ی معروف خود تحت عنوان مسیرها، درخت‌ها و گل‌ها ارایه داد. در گراف‌های دوبخشی ما می‌توانیم به‌سرعت به‌دنبال مسیرهای افزایشی بگردیم. (الگوریتم‌ارایه شده در بخش‌های قبلی همین فصل) چون می‌توانیم مکاشفه برای یافتن مسیرها را محدود کنیم، هنگامی که به‌دنبال یک مسیر افزایشی از یک راس خاص u می‌گردیم، رئوسی که در همان بخشی که u



شکل ۳.۲: مثالی برای درک بهتر مسیر افزایشی با وجود دور فرد در گراف.

در آن است، قرار دارند، فقط از طریق یال‌های M قابل دسترسی‌اند و رئوسی که در بخش مقابل u قرار دارند فقط از طریق یال‌هایی که خارج از M هستند قابل دسترسی می‌باشند—برای این موضوع، ما از یک راس حداکثر یک‌بار جستجو می‌کنیم. اما این خاصیت در مورد گراف‌های با دور فرد، که در آن‌ها یک مسیر تناوبی از یک راس آلوده نشده ممکن است به راس x توسط یک یال آلوده شده ویا یک یال آلوده نشده برسد، درست نیست.

مثال ۱. در گراف شکل ۳.۲ که در آن تطابق M با خطوط معمولی و دیگر یال‌ها با خطوط دوتایی نمایش داده شده است، جستجو برای کوتاه‌ترین مسیر افزایشی از راس u، به راس x از طریق یال آلوده نشده ax می‌رسد. اگر ما مسیر طولانی‌تری که به x از طریق یک یال آلوده شده می‌رسد را در نظر نگیریم آنگاه مسیرافزایشی u, v, a, b, c, d, x, y را از دست می‌دهیم.

ما راه‌حل ادموند را برای حل این مشکل ارایه می‌دهیم. اگر یک اکتشاف برای مسیر تناوبی از u به راس x توسط یک یال آلوده‌نشده از یک مسیر و توسط یک یال آلوده‌شده از مسیر دیگر برسد، در آن صورت x به یک دور فرد تعلق دارد. مسیرهای تناوبی از u فقط هنگامی می‌توانند از هم جدا شوند که یال بعدی آلوده نشده باشد. (به راس a در مثال ۱ دقت کنید). هنگامی که یال بعدی آلوده شده باشد فقط یک انتخاب وجود دارد. از راسی که مسیرها از هم جدامی‌شوند، می‌توان دید که یک دور فرد بوجود می‌آید.

تعریف ۴. فرض کنید M یک تطابق در گراف G باشد و u یک راس آلوده‌نشده توسط M باشد. یک گل اجتماع دو مسیر تناوبی از u است که به راس x با زوجیت طولی متضاد می‌رسند. ساقه گل بیشترین مسیر اولیه مشترک است . شکوفه گل، دور فرد ناشی از حذف ساقه است.

درمثال ۱، گل، کل گراف به جز y، ساقه، مسیر u, v, a و شکوفه یک دور فرد به طول ۵ است .

نکته جالب این‌جاست که شکوفه‌ها مانع از ادامه جستجوی ما نمی‌شوند. برای هر راس z در یک شکوفه، یک مسیر تناوبی از u به z وجود دارد که به z توسط یک یال آلوده شده می‌رسد به‌شرطی که از جهت مناسب به دور شکوفه بگردیم. لذا ما می‌توانیم جستجوی خود را در طی هر یال آلوده نشده از آن شکوفه به راسی که تاکنون به آن نرسیده‌ایم، ادامه دهیم. مثال ۱ نشان می‌دهد که چگونه این توسعه به یک راس آلوده نشده می‌رسد و یک مسیرافزایشی را کامل می‌کند.

از طرف دیگر، ما نمی‌توانیم یک شکوفه را با یک یال آلوده شده ترک کنیم. نتیجه این دو مشاهده آن است که ما می‌توانیم کل شکوفه را به‌عنوان یک ابرراس ببینیم که توسط یک یال آلوده‌شده در انتهای ساقه به آن رسیده‌ایم. ما می‌توانیم از تمام رئوس این ابرراس به‌طور هم‌زمان از طریق یال‌های آلوده‌نشده به جستجو بپردازیم.

برای پیاده‌سازی این روش ما یال‌های یک شکوفه B را وقتی آن‌را پیدا می‌کنیم فشرده می‌سازیم. نتیجه یک راس آلوده‌شده جدید b خواهد بود که مجاور آخرین یال ساقه می‌باشد. دیگر یال‌های مجاور این راس، یال‌های آلوده نشده‌ای که رئوس B را به رئوس خارج B متصل می‌کنند، خواهند بود. ما می‌توانیم از b به طریق معمولی به اکتشاف بپردازیم. هم‌چنین امکان دارد که بعداً یک شکوفه حاوی b بیابیم. اگر ما یک مسیر تناوبی در گراف فشرده‌شده، از u به یک راس آلوده نشده x بیابیم، آنگاه باید شکوفه‌ها را باز کنیم تا یک مسیر افزایشی به x در گراف اصلی بیابیم.

بجز اعمالی که در مورد شکوفه انجام می‌شود، روش همان الگوریتمی است که در مورد گراف دوبخشی در اوایل این فصل گفته شد. باز هم T زیر مجموعه‌ای از رئوس گراف جاری است که توسط یک یال آلوده نشده به آن‌ها رسیده‌ایم و S زیر مجموعه‌ای از رئوس است که توسط یال آلوده شده به

آن‌ها رسیده‌ایم. رئوسی که با فشرده‌سازی شکوفه‌ها تولید شده‌اند به S تعلق دارند.

ما مراحل الگوریتم را در زیر خلاصه می‌کنیم که نکاتی در مورد پیاده‌سازی هم به‌همراه دارد.

الگوریتم ۲ (الگوریتم شکوفه ادموند)

- **ورودی**: یک گراف G، یک تطابق M در G و یک راس آلوده‌نشده توسط M، (u) داده شده است.
- **خروجی**: یک تطابق M' که در صورت امکان یک یال بیشتر از M دارد.
- **ایده**: مسیرهای تناوبی از U را کشف کن، بدین ترتیب که S را رئوسی که توسط یال‌های M به آن‌ها دست‌یافته شده است قراربده و T را رئوسی که توسط یال‌هایی خارج از M به آن‌ها دست‌یافته شده است، قراربده. رئوسی از S را که از آن‌ها عمل اکتشاف صورت گرفته است علامت بزن وبرای هر راس $S \cup T$ راسی را که از آن دسترسی به این راس حاصل شده است نگه‌دار. هنگامی که شکوفه‌ها یافت شدند آن‌ها را فشرده‌ساز.
- **مقداردهی اولیه**: $S = u$ و $T = \emptyset$
- **تکرار**: اگر S هیچ راس علامت نخورده‌ای ندارد، کار را پایان بده و گزارش بده که هیچ مسیر افزایشی از u وجود ندارد. در غیر این‌صورت، یک راس علامت نخورده $v \in S$ را انتخاب کن. برای اکتشاف از v، هر $y \in N(x)$ به‌طوری‌که $xy \notin M$ است، را درنظر بگیر. اگر y آلوده نشده است، کار را تمام کن و از y به‌عقب برگرد (شکوفه‌ها را در صورت نیاز باز کن) تا یک مسیر افزایشی از u به y را گزارش کنی. در غیر اینصورت، y به راس w توسط یک یال تطابق جفت شده است. اگر $w \notin S$، آنگاه y را در T داخل کن (دست‌یافته از v) و w را در S داخل کن (دست یافته از y). اگر $w \in S$ (یا این‌که wیک همسایه v بود) آنگاه یک شکوفه پیدا شده است. شکوفه را فشرده کن، رئوسش را در S و T با یک راس جدید در S جایگزین کن و جستجو را در

گراف کوچک‌تر شده ادامه بده. بعد از اکتشاف از تمام این‌چنین یال‌های مجاور v، v را علامت بزن و این عمل را در مورد بقیه‌ی رئوس تکرار کن.

ما نمی‌توانیم از تمام رئوس آلوده نشده به‌طور هم‌زمان، آن‌گونه که در الگوریتم مربوط به تطابق درگراف‌های دوبخشی گفتیم جستجو کنیم، چون رفتار یک شکوفه بستگی به آن دارد که کجا ساقه به شکوفه رسیده است. با این وجود، اگر ما نتوانیم مسیر افزایشی از u بیابیم آنگاه می‌توانیم u را از گراف حذف کنیم و هنگامی که از دیگر رئوس بدنبال مسیر افزایشی می‌گردیم، ازآن صرف‌نظر کنیم.

الگوریتم اصلی ادموند در زمان $O(n^4)$ اجرا می‌شد.(پیاده‌سازی موجود در پیوست A این پایان نامه نیز از $O(n^4)$ است.) پیاده‌سازی که در سال ۱۹۹۳ توسط Orlin برای این الگوریتم داده‌شد در زمان $O(n^3)$ اجرا می‌شود، اما نیاز به دوچیز دارد:

۱) ساختمان داده مناسب برای نمایش شکوفه‌ها و عمل فشرده‌سازی.

۲) تحلیل دقیق تعداد فشرده‌سازی‌هایی که می‌توانند انجام شوند، زمانی که صرف اکتشاف از یال‌ها می‌شود و زمانی که صرف فشرده‌سازی و شکوفایی شکوفه‌ها می‌گردد.

اما اولین الگوریتمی که برای پیداکردن تطابق در حالت کلی داده شده و زمانی کم‌تر از $O(n^3)$ داشت، الگوریتمی بود که Even-Kariv در سال ۱۹۷۵ ارایه داد [۱]. بهترین الگوریتم شناخته‌شده در زمان $O(n^{1/2}m)$ برای گرافی با n راس و m یال اجرا می‌شود. این الگوریتم نسبتاً پیچیده است و در سال ۱۹۸۰ توسط Vazirani داده شد[٦].

٦.۲ ورودی و خروجی

توضیحات مربوط به ورودی و خروجی الگوریتم در ذیل آمده است.

ورودی در فایل MATCH.IN می‌باشد و شامل n (تعداد رئوس) در یک سطر و سپس شامل یال‌ها (هر یال در یک سطر) می‌باشد. ضمناً فرض شده است که تعداد رئوس از ۵۴ کم‌تر می‌باشد.

```
MATCH.IN                MATCH.OUT
8
1 2                     The number of matching edges is: 4
2 3                     1  2
3 4                     3  4
4 5                     5  6
5 6                     7  8
6 7
7 3
7 8
```

شکل ۴.۲: یک نمونه از فایل خروجی برنامه تطابق در حالت کلی.

خروجی در فایل MATCH.OUT قرار دارد که در آن ابتدا تعداد یال‌های تطابق و سپس خود یال‌ها هر کدام در یک سطر آمده‌اند.

مثال: نمونه‌ای از فایل ورودی و خروجی برای شکل ۳.۲، در شکل ۴.۲ آمده است.

فصل ۳

تاریخچه‌ی شکل‌گیری مسایل شبه‌تطابق

۱.۳ مقدمه

مسئله داده‌پراکنی کلی در هر شبکه‌ی نقطه‌به‌نقطه عبارت است از فرستادن اطلاعات از یک راس به تمام رئوس دیگر. این شمای ارتباطی در بسیاری از الگوریتم‌های موازی و الگوریتم‌های توزیع شده کاربرد فراوان دارد. به‌طور مثال، در الگوریتم‌های موازی یا توزیع شده، قبل از شروع الگوریتم از رویه‌ای به‌نام Spawn که کارش اطلاع دادن از شروع الگوریتم است، استفاده می‌شود. مسئله داده‌پراکنی جزیی تعمیمی از مسئله داده‌پراکنی کلی می‌باشد که در آن یک راس یک پیغام واحد را به زیرمجموعه‌ای از رئوس شبکه و نه لزوماً همه‌ی آن‌ها می‌فرستد. داده‌پراکنی جزیی کاربردهای فراوانی در هم‌روند کردن الگوریتم‌های موازی دارد. هم‌چنین یک ابزار اساسی برای پیاده‌سازی پایگاه‌های داده موازی می‌باشد. هم‌چنین الگوریتم‌های داده‌پراکنی جزیی برای دست‌یابی به کارایی بالا در بسیاری از الگوریتم‌های جبرخطی نظیر ضرب ماتریس‌ها، حل معادلات با روش حذف گوسی و نظایر آن کاربرد فراوان دارد.

کارایی سیستم رد و بدل پیغام، قویاً به توپولوژی شبکه ارتباطی و مکانیزم عبوردهی پیغام‌ها در شبکه، ارتباط دارد. در بسیاری از شبکه‌های مدرن موازی یا توزیع شده امروزی، مدل Cut-through جای مدل Store-and-forward راگرفته است. در مدل Store-and-forward بیشتر روی

مدل Unit-cost تکیه می‌شود که در آن هر انتقال پیغام توسط یک یال ارتباطی صورت گرفته و یک واحد زمانی به طول می‌انجامد. هم‌اکنون تحقیقات روی مدل Linear cost صورت می‌گیرد که در آن زمان پخش یک پیغام به‌نسبت طول مسیر آن پیغام می‌باشد. این مدل هم در مطالعات مسیریابی Store-and-forward و هم نوع‌های مختلفی از مدل Cut-through نظیر Circuit-switching (direct connect)، Wormhole routing مورد استفاده قرار می‌گیرد.

در مدل Circuit-switching هنگامی که یک راس x یک پیغام به راس y که همسایه‌اش نیست می‌فرستد، یک مسیر که مستقیماً این دو راس را به‌هم مرتبط می‌کند، بین این دو راس اختصاص داده می‌شود. آن‌وقت پیغام x به‌سمت y از طریق این مسیر فرستاده می‌شود. در هر راس میانی این مسیر، یک مسیریاب مسئول انتقال پیغام‌ها می‌باشد، ولی این رئوس لزوماً پیغامی را که از آن‌ها می‌گذرد، نمی‌فهمند. در حقیقت یک مسیریاب وقتی پیغام را به حافظه محلی خود می‌آورد که این راس یکی از مقاصد این پیغام باشد. مسیریابی به‌روش Wormhole با مسیریابی Ciruit-switching از این لحاظ تفاوت دارد که روش انتقال پیغام بین مبدا و مقصد اندکی تفاوت دارد. در روش Wormhole، یک پیغام به تکه‌های کوچکی شکسته می‌شود. سپس این تکه‌ها به‌صورت خط لوله به‌سمت پردازنده‌ی مبدا ارسال می‌شوند، به‌دین ترتیب که توسط اولین تکه، مسیر اختصاص داده می‌شود و با گذشت آخرین تکه از یک راس، مسیر آزاد می‌شود. بدین ترتیب در مدل Wormhole لزومی ندارد که تمام مسیرِ بین راس مبدا و راس مقصد رزرو شود بلکه همواره مقداری از مسیر بین این دو راس، بسته به طول پیغام رزرو می‌شود. با توجه به تعاریف بالا واضح است که مدل Circuit-switching بیشتر در مورد پیغام‌هایی که حجم زیادی دارند، استفاده می‌شود و مدل Wormhole برای پیغام‌های کوتاه، نظیر فرستادن یک مقدار صحیح [۱] در مسئله TRB و Consensus (این مسایل از مسایل معروف الگوریتم‌های توزیع‌شده می‌باشند). در ضمن همان‌طور که از تعریف این دو مدل برمی‌آید، این مدل‌هابا مسیرهای با طول زیاد هم کار می‌کنند، ولی کم‌کردن طول مسیرها در بهینه بودن مدل موثر است.

در این فصل ما بیشتر روی پیچیدگی الگوریتم‌ها برای داده‌پراکنی کلی و جزیی بحث می‌کنیم. برای این منظور ما مدل خطی [۲] را مدلی با سه خاصیت زیر تعریف می‌کنیم:

---

[۱] Integer

[۲] Line model

۱) یک ارتباط دقیقاً دو راس را دربرمی گیرد. این دو راس ممکن است فاصله‌ای بیشتر از یک داشته باشند.

۲) یک راس می‌تواند حداکثر در یک لحظه، در یک ارتباط شرکت کند.

۳) هر دو مسیر مجزای یالی معادل با دو ارتباط هم‌زمان می‌باشد.

مدل دیگری که به‌نام مدل تلفن شناخته می‌شود، شباهت بسیاری به مدل خطی دارد ولی تفاوت اندکی از این لحاظ دارد که طول مسیرهای ارتباطی باید یک باشد؛ به‌عبارت دیگر پیغام‌ها باید بین دو راس مجاور ردوبدل شوند. لازم به‌ذکر است که این مدل نیز کاربردهای فراوانی به‌خصوص در مسایل شایعه‌پراکنی [۳] دارد. هم‌چنین پیچیدگی محاسباتی مسایل بسیاری از این رده بررسی شده است.

در مدل خطی ماا‌جازه داریم که از مسیرهایی با طول دل‌خواه استفاده کنیم و این قابلیت از خواص مدل‌های Cut-through می‌باشد. در حقیقت ما برای برقراری ارتباطات بین رئوس دل‌خواه به‌نوعی از مدل Cut-through استفاده می‌کنیم. برای توضیح بیشتر به تعریف این مدل می‌پردازیم.

مدل Cut-through مدلی است که علاوه بر سه‌خاصیت ذکر شده در مورد مدل خطی خاصیت چهارم زیر را نیز دارا می‌باشد.

- (۴) مسیرهای ردوبدل پیغام‌ها توسط یک تابع مسیریاب ساخته می‌شوند.

برای ساده‌کردن تحلیل، این تابع مسیریاب را به‌صورت زیر تعریف می‌کنیم: تابع $R : V \times V \to E$ یک تابع مسیریاب است که در این‌جا V مجموعه رئوس و E مجموعه‌ی یال‌های گراف $G = (V, E)$ می‌باشد. این تابع به‌دین صورت عمل می‌کند که وقتی یک پیغام به مقصد y در راس x می‌باشد، این پیغام از طریق یال $R(x, y)$ فرستاده می‌شود (این یال یک یال مجاور x است). تابع R تطبیقی [۴] نامیده می‌شود اگر تابع مسیریاب چندین جواب برای مسیر دهی داشته باشد؛ یعنی $R : V \times V \to P(E)$. دراین حالت می‌تواند هر کدام از این مسیرها انتخاب شود. یک تابع مسیریاب، مینیمال نامیده می‌شود

[۳] (Gossip problems)

[۴] adaptive

اگر هر مسیر تولید شده توسط تابع مسیریاب، مینیم باشد، دراین فصل، مدل خطی به‌همراه خاصیت (۴)، مدل Cut-through می‌نامیم. (تابع لزومی ندارد که مینیمال یا تطبیقی باشد.)

بیشتر نتایج درمورد داده‌پراکنی کلی تحت این شرایط با شبکه‌های خاصی نظیر درخت‌ها، مش‌های دوبعدی، دورها یا توری‌ها [۲۰]،[۱۲]، [۱۵] بررسی شده‌اند. نتایج زیادی هم با تغییر فرض (۲) به فرض All Port که عبارت از این است که یک راس توانایی ارسال پیغام درون تمام یال‌های متصل به آن را دارد، نیز به‌دست آمده‌اند[۲۳]. هم‌چنین داده‌پراکنی کلی با تغییر شرط (۳) به مجزای راسی معرفی شده است(فصل بعد). متاسفانه بیشتر این مسایل هنگامی که به‌دنبال مینیم کردن تعداد مراحل هستیم، NP-complete می‌باشند. این مسایل در هر دو مدل Store-and-forward و Cut-through مورد بررسی قرارگرفته‌اند. برای اطلاعات بیشتر در این زمینه می‌توانید به [۲۴] رجوع کنید.

نکته قابل توجه این است که مسئله مینیم کردن مراحل در مدل خطی، چندجمله‌ای [۵] می‌باشد. دراین فصل ما مدل خطی را در مورد شبکه‌های با توپولوژی دل‌خواه بررسی می‌کنیم. لازم به‌ذکر است که با حل بسیاری از این مسایل در مدل خطی مسایل متناظر در مدل Cut-through نیز به‌دین صورت حل می‌شوند که می‌توان تابع مسیردهی را طوری طراحی کرد که این مسایل در همان تعداد مراحل مدل خطی، حل شوند.

هم‌چنین فرض می‌کنیم که در هر مرحله ارتباطات توسط یک سری مسیر مجزای راسی یا یالی برقرار می‌شود و تا اتمام ردوبدل کامل اطلاعات، در همان مرحله باقی خواهیم ماند. در مدل خطی فرض‌های (۱) و (۲) به این نکته اشاره دارند که در هر مرحله تعداد رئوس مطلع حداکثر دوبرابر می‌شوند، لذا حداقل پس از $\lceil \lg(n) \rceil$ مرحله می‌توان ادعا کرد که تمام رئوس از این پیغام آگاهند. Farley ثابت نمود که با این تعداد مرحله می‌توان این عمل را انجام داد[۱۹]. هم‌چنین در [۲۴] این قضیه بدین صورت تعمیم داده شده است: برای هر شبکه $G = (V, E)$ و برای هر راس $u \in V$ و هر مجموعه مقصد $D \subset V$ و $u \in D$، انجام داده‌پراکنی جزیی از u به‌تمام رئوس D در دقیقاً $\lceil \lg |d| \rceil$ مرحله قابل انجام است. ولی با وجود این این بحث بسته نشده است و سوالاتی از ایده زیر نشات می‌گیرند:

---

[۵] Polynomial

ایده: تا چه اندازه می‌توان کارایی این الگوریتم را بالا برد؟

برای درک بهتر این ایده که سر منشا بسیاری از مسایل این فصل و فصل بعدی است، به توضیح این ایده می‌پردازیم. درست است که مقدار بالا که کاملاً حداقل است برای مسئله داده‌پراکنی جزیی و کلی به‌دست آمده است، ولی دراین الگوریتم‌ها، بسیاری از پارامترها در نظر گرفته نمی‌شوند. به‌طور مثال امکان دارد که ترافیک انتقال پیغام‌ها برروی یک راس خاص خیلی زیاد باشد یا این‌که تعداد رئوس درگیر در ارتباطات زیاد باشد. هدف اصلی ما در این فصل این است که ببینیم چگونه می‌توان با حد پایین $\lceil \lg(n) \rceil$ مرحله، عمل داده‌پراکنی را انجام داد و در عین حال بهترین استفاده را از منابع انجام داد.

دربین پارامترهای بسیاری که کارایی داده‌پراکنی کلی و جزیی را بالا می‌برند ما در این فصل پارامترهایی را که اکثراً در [۲۴] تعریف شده‌اند، مورد بررسی قرار می‌دهیم.

قبل از هر چیز باید به این نکته توجه می‌کنیم که اکثر پارامترهای زیر را می‌توان هم دریک مرحله و هم در کل الگوریتم مورد بررسی قرارداد. بیشتر این پارامترها هنگامی که ما آن‌ها را در کل الگوریتم در نظر می‌گیریم مسئله را جزو رده NP-complete قرارمی‌دهند [۲۴]. در ابتدا، ما یک الگوریتم واحد را برای داده‌پراکنی کلی و جزیی ارایه می‌دهیم که این الگوریتم روش ما را در هر مرحله مشخص می‌کند. ولی در هر مرحله بسته به پارامترهای خواسته شده، الگوریتم فرق کرده و یا ثابت می‌کنیم الگوریتم از رده NP-complete می‌باشد. اما، پارامترهایی که در این فصل مورد توجه قرار می‌دهیم عبارتند از: طول مسیرها و تعداد مسیریاب‌ها. لازم به‌ذکر است که یک مسیریاب یک راس داخلی در مسیرها می‌باشد. هم‌چنین مسایل را به‌جای بررسی در حالت مجزای یالی به حالت مجزای راسی تعمیم داده و کلیه این مسایل را وقتی که گراف زمینه ما جهت‌دار است مورد بررسی قرار خواهیم داد. دراین بین ثابت خواهیم کرد که چندین مسئله NP-complete، P-complete، یا NC است و برای بعضی مسایل نیز الگوریتم ارایه خواهیم کرد. سرآخر در انتهای فصل چند مورد که می‌توانند مقدمه‌ای برای بحث‌های بعدی باشند ارایه می‌دهیم.

۲.۳ یک پروتکل کلی برای توابع مسیردهی مینیمال

در این بخش، ما ابتدا روی مجموع تعداد یال‌هایی که توسط یک پروتکل کلی ویا جزیی در هر مرحله و یا در کل پروتکل استفاده می‌شوند متمرکز می‌شویم. منظور از مجموع تعداد یال‌های استفاده شده در کل پروتکل، مجموع تعداد یال‌های استفاده شده در هر مرحله از پروتکل است(لذا یک یال می‌تواند بیش از یک بار در این مجموع شرکت کند) در هر مرحله از پروتکل، گره‌های آگاه از پیام با گره‌های دیگری که تاکنون پیام را نگرفته‌اند، جفت می‌شوند. برای رسمی‌تر گفتن این موضوع تعریف زیر را داریم:

تعریف ۵. شبه‌تطابق در گراف $G(V,E)$ برای رئوس U مجموعه‌ی P از $\lfloor \frac{|U|}{۲} \rfloor$ مسیر مجزای یالی P در G است که حداقل $|U| - ۱$ راس از رئوس U انتهاهای این مسیرها باشند.

نتیجه‌ی آتی نشان می‌دهد که یک شبه‌تطابق برای هر انتخاب $U \subseteq V$ وجود دارد. ما $d(x,y)$ را به‌عنوان فاصله بین دو راس y, x در گراف G در نظر می‌گیریم.

لم ۱. فرض کنید U یک زیرمجموعه از رئوس گراف $G = (V,E)$ باشد. می‌توان رئوس U را به زوج‌های مرتب $(x_k, y_k) \cdots (x_۱, y_۱)$ دسته‌بندی کرد که در این‌جا $k = \lfloor \frac{|v|}{۲} \rfloor$ و برای هر $i, j \in \{۱, \cdots, k\}$ که $x_i \neq x_j$ و $y_i \neq y_j$ است، هر کوتاه‌ترین مسیر بین y_i, x_i با هر کوتاه‌ترین مسیر بین y_j, x_j مجزای یالی بوده و سرآخر $\sum_{i=۱}^{k} d(x_i, y_i)$ کم‌ترین مقدار در بین تمام انتخاب‌های جفت‌های (x_i, y_i) $i = ۱ \cdots k$ باشد.

اثبات. فرض کنید $m = |U|$ و گراف کامل K_m که m راس دارد وهر راس متناظر با یک راس U است، در نظر بگیرید. یال $\{x, y\}$ دارای وزن $d(x,y)$ می‌باشد. یک تطابق کامل باوزن کمینه را در K_m در نظر بگیرید. این تطابق یک مجموعه P از $\lfloor \frac{|v|}{۲} \rfloor$ کوتاه‌ترین مسیرها در G است به‌طوری که هر راس U یکی از انتهاهای یکی از مسیرهاست.(در حالتی که بین دو راس چندین کوتاه‌ترین مسیر وجود دارد یکی از آن‌ها را به دل‌خواه انتخاب می‌کنیم.) ما ثابت می‌کنیم که مسیرهای P دوبدو مجزای یالی هستند. درحقیقت، فرض کنید که دو مسیر $P_{x,y}$ و $P_{x',y'}$ از P مجزای یالی نباشند. این بدان معناست

که دو راس z, z' وجود دارند به‌طوری‌که:

$$P_{x,y} = \{x, ..., z\} \cup \{z, ..., z'\} \cup \{z', ..., y\}$$

و

$$P_{x',y'} = \{x', ..., z\} \cup \{z, ..., z'\} \cup \{z', ..., y'\}$$

یا

$$P_{x',y'} = \{x', ..., y'\} \cup \{z', ..., z\} \cup \{z, ..., y'\}$$

بدون کاسته شدن از کلیت فرض می‌کنیم:

$$P_{x',y'} = \{x', ..., z\} \cup \{z, ..., z'\} \cup \{z', ..., y'\}.$$

این بدان معناست که دو جفت $\{x, y\}$ و $\{x', y'\}$ می‌توانند با دو جفت $\{x, x'\}$ و $\{y, y'\}$ جایگزین شوند. به‌راحتی می‌توان چک کرد که چون $d(z, z') \neq 0$، این تطابق جدید وزنی کم‌تر از تطابق قبلی دارد و لذا مخالف فرض انتخاب تطابق اول به‌عنوان تطابقی با کم‌ترین وزن می‌باشد. □

فرض کنید وزن یک شبه‌تطابق P در U را به‌عنوان مجموع طول تمام مسیرها در P در نظر بگیریم. ما به‌دنبال کمینه‌کردن وزن‌های چندین شبه‌تطابق در هر مرحله از پروتکل داده‌پراکنی کلی یا جزیی هستیم. یک شبه‌تطابق با وزن کم، استفاده کم‌تری از پهنای‌باند[۶] برای تبادل پیغام‌ها در دو سر مسیرها می‌کند. قضیه ۴ نشان می‌دهد که این کمینگی در زمان چند جمله‌ای ممکن است و قضیه Farley را بهبود می‌بخشد.

قضیه ۴. برای هر گراف $G = (V, E)$ و برای هر گره u از G می‌توان در زمان چند جمله‌ای یک پروتکل داده‌پراکنی جزیی از u به تمام مجموعه D از G داشت که $u \in D$ باشد و این اعمال $\lceil \log_2 |D| \rceil$ مرحله، در مدل خطی طول می‌کشد. هم‌چنین:

۱) تمام تماس‌ها از بین کوتاه‌ترین مسیرها صورت می‌گیرد.

۲) در هر کدام از $\lceil \log_2 |D| \rceil$ مرحله، وزن شبه‌تطابق متناظر، مینیمم است.

---

[۶] Band width

اثبات. $\lceil \log_2 |D| \rceil$ شبه‌تطابق به‌صورت عقب‌گرد ساخته می‌شوند. با شروع از $U_1 = D$ و با استفاده از لم ۱ یک شبه‌تطابق P_1 در U_1 با کم‌ترین وزن و شامل فقط کوتاه‌ترین مسیرها ساخته‌می‌شود. سپس یکی از دو سر هر مسیر را در P_1 انتخاب می‌کنیم. این انتخاب می‌تواند، تصادفی یا دل‌خواه باشد. فقط یک حالت خاص برای مسیری که راس شروع u را دارد، باید در نظر گرفته شود؛ چون راس شروع u باید انتخاب شود. هم‌چنین تک راس باقی‌مانده در U_1 را اگر $|U_1|$ فرد باشد باید انتخاب کنیم. تمام این گره‌ها مجموعه‌ی U_2 را تشکیل می‌دهند. سپس یک شبه‌تطابق P_2 در U_2 با کم‌ترین وزن و فقط شامل کوتاه‌ترین مسیرها را انتخاب می‌کنیم. دوباره، این‌کار برطبق لم ۱ امکان‌پذیر است. سپس مجموعه‌ی U_3 را از U_2 همان‌گونه که U_2، از U_1 استخراج شده بود، بیرون می‌کشیم و این عمل را آن‌قدر انجام می‌دهیم تا این‌که مجموعه U_i به یک راس u خاتمه یابد. بداهتاً $i = \lceil \log_2 |D| \rceil$ خواهد بود. این پروتکل می‌تواند در زمان چندجمله‌ای محاسبه شود، چون می‌توان تطابق کامل با مینیمم وزن را در یک گراف کامل در زمان چندجمله‌ای محاسبه نمود. □

البته این نتیجه در مدل Cut-through هم درست است چون طریقی که کوتاه‌ترین مسیرها انتخاب می‌شوند در اثبات لم ۱ اهمیتی ندارد، خصوصاً که آن‌ها می‌توانند با استفاده از هر تابع مسیردهی مینیمال ساخته شوند. لذا ما قضیه زیر داریم:

قضیه ۵. برای هر گراف $G = (V, E)$ و برای هر زیر مجموعه $D \subseteq V$ و برای هر گره $u \in D$ و برای هر تابع مسیردهی مینیمال (که می‌تواند تطبیقی هم باشد) می‌توان در زمان چندجمله‌ای یک پروتکل داده‌پراکنی جزیی از u به D را محاسبه نمود؛ به‌طوری‌که در $i = \lceil \log_2 |D| \rceil$ مرحله در مدل Cut-through انجام شده و در ضمن ، در هر کدام از $i = \lceil \log_2 |D| \rceil$ مرحله، وزن شبه‌تطابق آن مرحله، کمینه باشد.

قضیه ۵ می‌گوید که شبکه و تابع مسیردهی هرچه باشد، اگر این تابع مسیردهی پیام‌ها را در طی کوتاه‌ترین مسیرها بفرستد، می‌توان داده‌پراکنی جزیی و کلی را در کم‌ترین تعداد مراحل انجام داد. بیشتر توابع مسیردهی در توپولوژی‌های معمولی (نظیر مش، توری چند بعدی و...) در کوتاه‌ترین مسیر، مسیریابی می‌کند. اگر مسیردهی Wormhole استفاده شود جالب است که هیچ تناقضی بین جستجو برای تابع مسیردهی بدون وقفه و جستجو برای یک تابع مسیردهی که داده‌پراکنی کلی وجزیی را

تضمین کند وجود ندارد، در حقیقت بیشتر توابع مسیردهی بدون وقفه فقط کوتاه‌ترین مسیرها را تولید می‌کنند.

قضایای ۴ و ۵ هر دو می‌گویند که می‌توان به‌راحتی مجموع طول مسیرها در هر مرحله را کمینه کرد، اما در مورد مجموع طول تمامی مسیرها که در طی تمامی مراحل استفاده می‌شوند، چیزی نمی‌گویند. به‌عنوان یک مثال نقض مسیر $P_۴ = (x_۰, x_۱, x_۲, x_۳)$ و راس شروع $x_۰$ را در نظر بگیرید. از پروتکل اثبات قضیه ۴ $P_۱ = \{(x_۰, x_۱), (x_۲, x_۳)\}$ حاصل می‌شود. لذا اگر $x_۰$ و $x_۳$ برای مرحله‌ی بعد انتخاب شوند(با توجه به این‌که بجز راس شروع، انتخاب بقیه اختیاری بود) آن‌گاه $P_۲ = \{(x_۰, x_۱, x_۲, x_۳)\}$ خواهد بود. در هر مرحله تعداد کانال‌های ارتباطی کمینه است، ولی مجموع تمام طول مسیرها برابر $۵ = ۳ + ۱ + ۱$ است، در حالی‌که مینیمم ۴ است. نتیجه‌ی بعدی نشان می‌دهد که کمینه‌کردن کلی مجموع طول مسیرها NP-complete است.

قضیه ٦. مساله زیر NP-complete است.

کم‌ترین مجموع طول مسیر (MTPL)

ورودی: گراف $G = (V, E)$، یک راس u از G، یک زیر مجموعه $D \subseteq V$، $u \in D$ و یک عدد صحیح K داده شده است.

سوال: آیا یک پروتکل داده‌پراکنی جزیی از u به D در G وجود دارد به‌طوری‌که در $i = \lceil \log_۲ |D| \rceil$ مرحله در مدل خطی انجام گیرد و مجموع طول تمام مسیرها در تمامی مراحل این پروتکل حداکثر k باشد.

اثبات. اثبات قضیه ٦ براساس لم ۲ صورت می‌گیرد(به‌یاد بیاورید که مدل تلفن فقط ارتباطات همسایه به همسایه را اجازه می‌دهد).

لم ۲. این مساله NP-complete است:

داده‌پراکنی کلی لگاریتمی (LOGB)

ورودی: یک گراف $G = (V, E)$ با n راس و یک راس خاص u داده شده است.

سوال: آیا یک پروتکل داده‌پراکنی کلی از u در G وجود دارد که حداکثر در $\lceil \log_2 n \rceil$ مرحله انجام گیرد.

اثبات. قضیه مشهور زیر که توسط Johnson نشان داده‌شده است که NP-complete است را در نظر بگیرید.

داده‌پراکنی کلی

ورودی: یک گراف $G = (V, E)$، یک راس u از G و یک عدد صحیح k داده‌شده است.

سوال: آیا یک پروتکل داده‌پراکنی کلی از u در G وجود دارد که در حداکثر k مرحله در مدل تلفن انجام گیرد؟

لم ۲ تقریبا به‌همان روشی که در [۱۶] استفاده شده است، از قضیه بالا قابل حصول است.

□ **اثبات قضیه‌ی ٦** : واضح است که MTPL، NP می‌باشد. اثبات NP-complete بودن آن‌را با تبدیل از LOGB به این مساله می‌آوریم. فرض کنید G و u یک نمونه از مساله‌ی $LOGB$ باشند. ما این نمونه را به نمونه u، k، G و D از مساله MTPL با قرار دادن $D = V$ و $k = n - ۱$ انجام می‌دهیم. به‌وضوح اگر یک پروتکل داده‌پراکنی کلی از u در G باشد که در حداکثر $\lceil \log_2 n \rceil$ مرحله در مدل تلفن انجام شود، آن‌گاه یک پروتکل داده‌پراکنی کلی از u در G در $\lceil \log_2 n \rceil$ مرحله در مدل خطی وجود دارد به‌طوری‌که مجموع طول تمام مسیرهای ارتباطاتی حداکثر $n - ۱$ باشد. برعکس، اگر یک پروتکل داده پراکنی کلی از u در G باشد که در $\lceil \log_2 n \rceil$ مرحله در مدل خطی انجام گیرد به‌طوری‌که مجموع طول تمام مسیرهای ارتباطاتی حداکثر $n - ۱$ باشد، آنگاه تمام تماس‌ها باید بین رئوس همسایه صورت گیرد و بنابراین، یک پروتکل داده‌پراکنی کلی از u در G که در $\lceil \log_2 n \rceil$ مرحله در مدل تلفن صورت می‌گیرد، وجود دارد. □

قضیه ٦ نشان می‌دهد که مینیمم کردن بار یال‌ها در طی داده‌پراکنی کلی یا جزیی، کار دشواری است، بااین وجود می‌توان این مقدار را با یک عامل لگاریتمی تقریب زد.

تعریف ٦. برای هر گراف $G = (V, E)$، هر گره $u \in V$ و هر مجموعه $D \subseteq V$، $u \in D$، فرض کنید که $S(G, D, u)$ نشان‌دهنده کم‌ترین مقدار مجموع طول تمام مسیرها از میان تمام پروتکل‌های

داده‌پراکنی جزیی از u به D در G و در $\lceil \log_2 |D| \rceil$ مرحله، است.

قضیه ٦ نشان می‌دهد که مینیمم کردن $S(G, D, u)$ یک مساله NP-complete است. با این وجود داریم:

قضیه ۷. فرض کنید $G = (V, E)$ و $D \subseteq V$ و $u \in D$ باشد. مجموع کلی طول تمام مسیرها که توسط پروتکل داده‌پراکنی جزیی قضیه ۴ تولید شده‌اند حداکثر $\lceil \log_2 |D| \rceil S(G, D, u)$ است.

اثبات. چون هر راس در D باید پیام را دریافت کند، $S(G, D, u)$ حداقل برابر تعداد یال‌های n_s درخت اشتاینری است که D را می‌پوشاند. برای هر i، $1 \leq i \leq \lceil \log_2 |D| \rceil$ تعداد یال‌ها شبه‌تطابق U_i که در الگوریتم اثبات قضیه ۴ داده شده، حداکثر شامل تعداد یال‌های T می‌باشد. در حقیقت، می‌توان فقط یک شبه‌تطابق U_i را با استفاده از یال‌های T پیدا کرد و لذا شبه‌تطابق U_i در G از یال‌های کم‌تری استفاده می‌کند. بنابراین، برای هر i، $1 \leq i \leq \lceil \log_2 |D| \rceil$ تعداد یال‌های شبه‌تطابق U_i حداکثر برابر تعداد یال‌های n_s درخت اشتاینرِ پوشاننده D است. لذا، پروتکل قضیه ۴، مسیرهایی به طول حداکثر $\lceil \log_2 |D| \rceil n_s \leq \lceil \log_2 |D| \rceil S(G, D, u)$ تولید می‌کند. □

ما این بخش را با یک خاصیت شبه‌تطابق پایان می‌بریم که در واقع شهود خوبی از یک شبه‌تطابق و تعداد یال‌های آن به‌ما می‌دهد.

گزاره ۱. هر شبه‌تطابق با وزن کمینه یک جنگل (یک مجموعه از درخت‌های جدا از هم) می‌باشد.

اثبات این گزاره را می‌توانید در[۲۴]ببینید.

لازم به‌ذکر است که می‌توان تقریب ارایه شده در قضیه ۷ را برای درخت‌ها بهبود بخشید[۲۴].

در بخش بعدی تعدادی از تعاریف و قضایای مطرح شده در مرجع اصلی این پایان‌نامه[۲۴] لیست شده است.

۳.۳ قضایای مربوط به کمینه کردن بار، تعداد مسیریاب‌ها و زمان تعویض

تعریف ۷. یک راس مسیریاب در یک شبه‌تطابق راسی است که به‌عنوان راس درونی یک مسیر به‌کار رفته است.

قضیه ۸. مسئله‌ی یافتن یک شبه‌تطابق در گراف G که حداکثر k راس انتقال‌دهنده دارد، NP-complete است. این مسئله برای درخت‌ها قابل حل است.

اثبات این قضیه با تبدیل از مساله 3SAT انجام می‌گیرد.

قضیه ۹. مسئله‌ی یافتن یک داده‌پراکنی در تعداد مراحل کمینه و با تعداد کلی رئوس انتقال‌دهنده NP-complete است.

قضیه ۱۰. مسئله‌ی یافتن یک شبه‌تطابق در گراف G که حداکثر درجه‌ی یک راس در آن k باشد، NP-complete است.

اثبات این قضیه با تبدیل از مساله پوشش راسی[۷] انجام می‌گیرد.

قضیه ۱۱. مسئله‌ی یافتن یک شبه‌تطابق در گراف G که حداکثر طول یک مسیر در آن k باشد، NP-complete است.

اثبات این قضیه با تبدیل از مساله پوشش راسی[۸] انجام می‌گیرد.

---

[۷] Vertex cover

[۸] Vertex cover

۴.۳ نتیجه‌گیری و مسایل باز

با توجه به قضایای گفته‌شده شاید وضعیت کنونی در نگاه اول ناامید کننده به‌نظر برسد(چون بیشتر مسایل NP-complete می‌باشند) خواننده باید در نظر داشته باشد که پارامتر اصلی برای بهینه‌کردن در شبکه‌های Cut-through، استفاده بهینه از پهنای باند[۹] می‌باشد که ما آن‌را برآورده ساختیم در حقیقت، اولین خاصیتی که یک پروتکل ارتباطاتی در مدل خطی یا Cut-through باید ارضا کند این است که باید از درگیری‌های اتصال[۱۰]، بدور باشد. از این نقطه نظر، ما نتایج مهمی را به‌دست آوردیم:

۱) یک الگوریتم چندجمله‌ای یافتیم که برای هر شبکه و هر تابع مسیردهی تطبیقی مینیمال، یک پروتکل داده‌پراکنی کلی یا جزیی را برمی‌گرداند که در مینیمال تعداد مراحل انجام می‌گیرد.

۲) بعلاوه، این پروتکل‌های داده‌پراکنی جزیی یا کلی در هر مرحله مجموع طول مسیرها را کمینه نگه می‌دارند به‌عبارت دیگر پهنای باند کلی خواسته شده در هر مرحله، کمینه است.

۳) اگرچه کمینه‌سازی استفاده از پهنای باند در کل پرتکل NP-complete است، الگوریتم ما با حد پایینی، فقط در یک ضریب لگاریتمی تفاوت دارد.

کمینه‌سازی پارامترهای دسته‌دوم از قبیل تعداد انتقال‌دهنده‌ها، بار انتقال‌دهنده‌ها، بیشترین طول مسیرها، مجموع بیشترین طول مسیرهای استفاده شده در هر مرحله و غیره NP-complete می‌باشند. این مسایل اغلب برای مسیریابی Cut-through خیلی حیاتی نمی‌باشند، لذا ما سعی در تولید راه‌حل‌های تقریبی نکردیم؛ هرچند اگر، کمینه سازی این پارامترها یک موضوع مهم بخاطر بعضی دلایل تکنیکی باشد، این الگوریتم‌ها باید پیدا شوند. لذا این، یک جهت‌دهی مهم برای تحقیقات آتی بر روی این موضوع می‌باشد.

مسایل دیگری هم به‌نظر خیلی مهم می‌رسند. ما در این‌جا دو جنبه دیگر را که هم‌اکنون تحقیقات برروی آن‌ها ادامه دارد، ذکر می‌کنیم.

---

[۹] Band-width

[۱۰] Link contention

۱.۴.۳ مدل تمام‌پورت

همانطور که ما در مقدمه گفتیم، می‌توان مدلی را در نظر گرفت که گره u بادرجه Δ_u بتواند Δ_u پیام که می‌توانند هم‌زمان توسط مسیریاب‌ها به مقاصد متفاوتی گسیل شوند، تولید کند. اگر شبکه Δ—منظم باشد داریم: $b(G) \geq \lceil \log_{\Delta+1} n \rceil$ (n تعداد رئوس گراف و $b(G)$ کم‌ترین تعداد مراحل است). هم‌چنین نشان داده شده است که دانستن این‌که آیا $b(G) \leq k$ است برای هر شبکه دل‌خواه داده شده‌ی G و هر عدد ثابت k، درست است یا خیر NP-complete است. لذا این مساله الگوریتم‌های تقریبی چند جمله‌ای نیاز دارد.

۲.۴.۳ مساله شایعه پراکنی

این مساله داده‌پراکنی همه‌به‌همه نیز نامیده می‌شود و داده‌پراکنی هم‌زمان از تمام گره‌های شبکه را شامل می‌شود. یک الگوریتم اولیه این است که ابتدا تمام پیام‌ها در یک راس جمع‌آوری شده و سپس داده‌پراکنی کلی این پیام‌ها از این راس صورت گیرد. با وجود آن‌که این پروتکل کاملاً از لحاظ تعداد مراحل کارا است، ولی ترافیک خطی را به‌صورت متعادل ایجاد نمی‌کند؛ لذا پروتکل‌های کاراتری را می‌طلبد.

۳.۴.۳ بررسی این مدل در گراف‌های جهت‌دار

این تعمیم نیز در این موضوع در نظر گرفته شده است و نتایج جالبی به‌دست آمده است که در فصل بعد به آن‌ها اشاره می‌کنیم.

آنچه در این فصل مطرح شد بیشتر نتایجی بود که در منبع اصلی این پایان‌نامه [۲۴] ذکر شده است در فصل بعد به نتایجی که خود به‌دست آورده‌ایم و نیز مسایل باز در آن زمینه اشاره خواهیم کرد.

فصل ۴

نتایج جدید در مورد شبه‌تطابق

۱.۴ مقدمه

در این فصل به نتایج جدید به‌دست آمده اشاره خواهیم کرد. در بخش ۲ مسئله‌ی شبه‌تطابق در حالت $U = V(G)$ را بررسی می‌کنیم.

در بقیه‌ی بخش‌ها به بررسی مسایل شبه‌تطابق در حالت‌های مجزای رأسی و مجزای یالی و در گراف‌های جهت‌دار و بی‌جهت می‌پردازیم. در هر حالت محدودیت دیگری هم روی این مسئله قرار می‌دهیم که یکی از انواع این محدودیت‌ها، محدودیت روی طول مسیرها است. پس هر مسئله‌ی شبه‌تطابق به‌صورت یک سه‌تایی:

([مجزای رأسی/مجزای یالی]، [جهت‌دار/بی‌جهت]،محدودیت(مثلاً $k \leq$ طول مسیرها))

مشخص می‌شود. ما مسئله‌ی شبه‌تطابق با محدودیت طول مسیرها را در هر حالت برای یک k ی ثابت بررسی می‌کنیم و در هر حالت بررسی می‌کنیم که مسئله در P یا NPC قرار دارد. در بخش ۳ برای تعدادی از حالت‌هایی که k مقدار ثابت کوچکی دارد، اثبات می‌کنیم مسئله در P قرار دارد و الگوریتم چندجمله‌ای حل آن را بیان می‌کنیم. در بخش ۴ با استفاده از چند مسئله‌ی NP-Complete و تبدیل آن‌ها به مسئله‌ی خودمان ثابت می‌کنیم در حالت‌هایی که $k \geq ۹$ مسئله‌ی شبه‌تطابق NP-Complete است. در بخش آخر مسایل باز و حل‌نشده را بیان خواهیم کرد.

۲.۴ مسئله‌ی شبه‌تطابق در حالت $U = V(G)$

در مسئله‌ی داده‌پراکنی روی شبکه در مرحله‌ی آخر باید روی یک شبه‌تطابق که همه‌ی رئوس گراف را می‌پوشاند، پیغام‌ها را از نیمی از رأس‌ها به بقیه‌ی آن‌ها منتقل کنیم. چیزی که تابه‌حال ثابت شده بود، این بود که برای هر زیرمجموعه‌ی U از رئوس روی شبکه‌ی همبند شبه‌تطابق مجزای یالی‌یی روی U وجود دارد. قضیه‌ی بعدی نشان می‌دهد، در حالت $U = V(G)$ شبه‌تطابق مجزای یالی‌یی روی U وجود دارد که طول هر مسیر آن حداکثر ۲ است.

قضیه ۱۲ . اگر مجموعه‌ی U، برابر V باشد، و گراف G همبند باشد، گراف G حتماً دارای شبه‌تطابقی مجزای یالی روی U است که طول هر مسیر آن حداکثر ۲ است. ضمناً الگوریتمی چندجمله‌ای برای یافتن شبه‌تطابق با حداکثر طول مسیر ۲ وجود دارد.

اثبات. نخست ثابت می‌کنیم هر درخت T حتماً شبه‌تطابقی با شرایط مسئله دارد و الگوریتمی چندجمله‌ای برای آن ارایه می‌دهیم. الگوریتم به این منوال است که در هر مرحله اگر در T دو برگ وجود داشته باشند که همسایه‌ی مشترکی داشته باشند، در این صورت مسیر دوتایی بین این دو برگ را به‌عنوان مسیری از مسیرهای شبه‌تطابق در نظر می‌گیریم. این کار را ادامه می‌دهیم تا جایی که هیچ دو برگی با این خاصیت در گراف وجود نداشته باشند، حال به‌ازای هر برگ v، اگر u مجاور v باشد، درجه‌ی u برابر ۲ است. حال برای دو رأس u و v مسیر ۱تایی uv را در نظر می‌گیریم و این دو رأس را از گراف حذف می‌کنیم. به این ترتیب اگر تعداد رئوس گراف فرد نباشد، در آخر همه‌ی رئوس گراف ابتدا یا انتهای یک مسیر ۱ یا ۲ یالی می‌شوند و در صورت فرد بودن تعداد رئوس فقط یک رأس باقی می‌ماند. پس حکم مسئله به‌همراه الگوریتم آن ارایه شد. □

توضیح ۱ .

در قضیه‌ی بالا، الگوریتم ارایه شده الگوریتمی NC بود. زیرا برای هر گراف G ی همبند یک زیردرخت فراگیر آن را می‌توان توسط یک الگوریتم NC به‌دست آورد (در شبکه‌ی MOT می‌توان این

کار را در زمان $O(\lg n)^{۲}$ انجام داد). سپس با این الگوریتم شبه‌تطابق را در درخت می‌یابیم. و الگوریتم به‌دست آمده NC خواهد بود.

توجه دارید که مطلب بالا در مرحله‌ی آخر broadcast کردن در الگوریتم کارآیی دارد که به‌جای تطابق از این الگوریتم می‌توان استفاده کرد.

۳.۴ الگوریتم‌هایی برای حالاتی از مسئله‌ی شبه‌تطابق

در این قسمت به بررسی چندین مسئله‌ی شبه‌تطابق با محدودیت طول مسیرها برای ثابت‌های کوچک می‌پردازیم.

قضیه ۱۳. الگوریتمی با زمان اجرای چندجمله‌ای برای پیدا کردن شبه‌تطابق با مسیرهای مجزای رأسی به طول حداکثر ۲ در گراف‌های بدون جهت وجود دارد، یا به‌عبارت دیگر مسئله‌ی شبه‌تطابق (مجزای رأسی، بی‌جهت، $۲ \leq$ طول مسیرها) در P قرار دارد.

اثبات. فرض کنید گراف G و زیرمجموعه‌ی U از رئوس آن داده شده‌اند. می‌خواهیم به تعداد $\frac{|U|}{۲}$ (فرض می‌کنیم $|U|$ زوج باشد) مسیر مجزای یالی پیدا کنیم که طول هر یک از آن‌ها حداکثر ۲ باشد، مسیرها مجزای رأسی باشند (یعنی رأس مشترک نداشته باشند)، و هر یک از رئوس U انتهای دقیقاً یکی از مسیرها باشد.

اولاً بدون کاسته شدن از کلیت می‌توان فرض کرد که U یک مجموعه‌ی مستقل است و طول هر یک از مسیرهایی که باید پیدا کنیم باید دقیقاً برابر با ۲ باشد (چون در صورتی که دو رأس مثل x و y از U به هم متصل باشند، کافیست یال بین آن‌ها را حذف کرده و به‌جای آن یک مسیر به طول ۲ بین آن‌ها قرار دهیم). علاوه بر این می‌توانیم فرض کنیم که $V-U$ نیز یک مجموعه‌ی مستقل از رئوس است (و یا به‌عبارت دیگر گراف G دوبخشی است)، چون اگر یالی بین دو رأس مجموعه‌ی $V-U$ وجود داشته باشد، این یال نمی‌تواند در هیچ‌یک از مسیرها به‌کار رود، و بنابراین می‌توانیم آن را حذف کنیم.

حالا در حقیقت باید مسئله‌ی زیر را حل کنیم: گراف دوبخشی G با دو بخش X و Y داده شده است. می‌خواهیم زیرگرافی از G بیابیم که در آن درجه‌ی هر رأس X دقیقاً برابر با ۱ و درجه‌ی هر رأس Y یا برابر با ۲ و یا برابر با ۰ باشد. برای حل این مسئله گراف G' را از روی گراف G بدین‌صورت می‌سازیم: گراف G' مانند گراف G است، با این تفاوت که به‌جای هر رأس y در Y، دو رأس y_1 و y_2 (که به‌هم متصل نیستند) در آن قرار می‌دهیم، و هر دوی این رئوس را به همه‌ی رأس‌هایی که y در G به آن‌ها وصل بود، وصل می‌کنیم. دو رأس y_1 و y_2 را دو رأس دوقلو می‌نامیم. اکنون مسئله به این صورت در می‌آید: می‌خواهیم در گراف G' یک تطابق پیدا کنیم به‌طوری که تطابق همه‌ی رئوس X را بپوشاند و برای هر دو رأس دوقلو مثل y_1, y_2، یا y_1 و y_2 هر دو توسط تطابق پوشیده شده باشند، و یا هیچ‌کدام پوشیده نشده باشند. چنین تطابقی را یک تطابق قابل قبول می‌نامیم. برای حل این مسئله، متناظر با هر دو رأس دوقلو مثل y_1, y_2، به گراف G' دو رأس v_1, v_2 می‌افزاییم و y_1 را به v_1، y_2 را به v_2، و v_1 را به v_2 وصل می‌کنیم. گراف حاصل را G'' می‌نامیم.

گزاره ۲. *G' یک تطابق قابل قبول دارد اگر و تنها اگر G'' یک تطابق کامل داشته باشد.*

اثبات. اگر G' یک تطابق قابل قبول M داشته باشد، برای هر دو رأس دوقلوی y_1, y_2، اگر هر دوی آن‌ها توسط M پوشیده شده باشند، یال v_1v_2، و در غیر این صورت یال‌های v_1y_1 و v_2y_2 را به تطابق M اضافه می‌کنیم. به سادگی می‌توان تحقیق کرد که مجموعه‌ی حاصل یک تطابق کامل در G'' است.

برعکس، اگر G'' یک تطابق کامل M داشته باشد، مجموعه‌ی یال‌هایی از این تطابق که بین رئوس گراف G' هستند ($M' := M \cap E(G')$)، یک تطابق قابل قبول است، چون در غیر این صورت باید دو رأس دوقلوی y_1, y_2 وجود داشته باشند که فقط یکی از آن‌ها، مثلاً y_1، توسط تطابق پوشیده شده باشد. در این‌صورت y_2 باید در M با یال v_2y_2 پوشیده شده باشد. در نتیجه v_1 نمی‌تواند با هیچ یالی در M پوشیده شود که این با کامل بودن تطابق M در تناقض است. □

بنا بر گزاره‌ی فوق، برای حل کردن مسئله، کافیست گراف G'' را بسازیم، و سپس از الگوریتم پیدا کردن تطابق در گراف‌های کلی استفاده کنیم. □

قضیه ۱۴. الگوریتمی با زمان اجرای چندجمله‌ای برای پیدا کردن شبه‌تطابق با مسیرهای مجزای رأسی به طول حداکثر ۳ در گراف‌های بدون جهت وجود دارد، یا به عبارت دیگر مسئله‌ی شبه‌تطابق (مجزای رأسی، بی‌جهت، ۳ ≤ طول مسیرها) در P قرار دارد.

اثبات. فرض کنید گراف G و زیرمجموعه‌ی U از رئوس آن داده شده است. مانند اثبات قضیه‌ی ۱۳ سعی می‌کنیم این مسئله را به مسئله‌ی پیدا کردن یک تطابق قابل قبول (این بار در یک گراف غیر دوبخشی) تبدیل کنیم. برای این منظور گراف G' را به این صورت می‌سازیم: گراف G' مانند گراف G است، با این تفاوت که به جای هر رأس y از $V-U$، دو رأس دوقلوی y_1 و y_2 در G' می‌گذاریم و این رأس‌ها را به این صورت به‌هم وصل می‌کنیم:

- اگر در G دو رأس x و x' از U به‌هم متصل باشند، x و x' را در G' نیز به‌هم متصل می‌کنیم.
- اگر در G یک رأس x از U به y از $V-U$ وصل باشد، x را در G' به هر دوی y_1 و y_2 وصل می‌کنیم.
- اگر در G دو رأس y و y' از $V-U$ به‌هم وصل باشند، در G'، y_1 را به y'_1 وصل می‌کنیم.

به‌سادگی می‌توان تحقیق کرد که هر تطابق قابل‌قبول در G' متناظر با یک جواب مسئله است. (کافیست دو رأس دوقلوی متناظر با یک رأس را تبدیل به یک رأس کنیم.) بنابراین الگوریتمی که برای یافتن تطابق قابل‌قبول در اثبات قضیه‌ی ۱۳ ذکر شد، اثبات این قضیه را کامل می‌کند. □

قضیه ۱۵. الگوریتمی با زمان اجرای چندجمله‌ای برای پیدا کردن شبه‌تطابق با مسیرهای مجزای رأسی به طول حداکثر ۲ در گراف‌های جهت‌دار وجود دارد، به‌عبارت دیگر مسئله‌ی شبه‌تطابق (مجزای رأسی، جهت‌دار، ۲ ≤ طول مسیرها) در P قرار دارد.

اثبات. اثبات مانند اثبات قضیه‌ی قبل است، با این تفاوت که متناظر با هر رأس y از Y، دو رأس دوقلوی y_1 و y_2 در G' می‌گذاریم، و همه‌ی رأس‌هایی از G که به y یال داشته‌اند را به y_1، و همه‌ی

رأس‌هایی از G که y به آن‌ها یال داشته است را به y_2 وصل می‌کنیم. حالا به‌سادگی می‌توان تحقیق کرد که هر جواب این مسئله، متناظر با یک تطابق قابل‌قبول در G' است. □

قضیه ۱۶. الگوریتمی با زمان اجرای چندجمله‌ای برای پیدا کردن شبه‌تطابق با مسیرهای مجزای رأسی بدون محدودیت در طول مسیرها در گراف‌های بدون جهت و جهت‌دار وجود دارد.

اثبات. مشابه مسایل بالا گراف G' را از روی گراف G با اضافه کردن y_iها به رئوس و اضافه کردن یال‌های مناسب می‌سازیم. حال وجود شبه‌تطابق با مسیرهای مجزای رأسی بدون محدودیت در G متناظر وجود تطابقی قابل قبول در G' است. □

قضیه ۱۷. اگر الگوریتمی چندجمله‌ای برای مسئله‌ی شبه‌تطابق (مجزای رأسی، بی‌جهت، $k \leq$ طول مسیرها) برای یک k ی ثابت وجود داشته باشد، الگوریتمی چندجمله‌ای برای مسئله‌ی شبه‌تطابق (مجزای یالی، بی‌جهت، $k-1 \leq$ طول مسیرها) وجود دارد.

اثبات. فرض کنیم الگوریتم چندجمله‌ای ALG برای مسئله‌ی شبه‌تطابق (مجزای رأسی، بی‌جهت، $k \leq$ طول مسیرها) برای یک k ی وجود داشته باشد. برای حل مسئله‌ی شبه‌تطابق (مجزای یالی، بی‌جهت، $k-1 \leq$ طول مسیرها) ورودی مسئله را گراف G و مجموعه‌ی U از رئوس را در نظر بگیرید. از روی گراف G گراف G' را به این ترتیب می‌سازیم که متناظر هر یال G یک رأس در G' قرار می‌دهیم و متناظر هر رأس از U هم یک رأس در G' قرار می‌دهیم. حال بین هر دو رأس متناظر دو یال مجاور در G، یک یال در G' قرار می‌دهیم و هر رأس متناظر یک رأس U در G' را به همه‌ی رئوس متناظر یال‌های مجاور آن در G وصل می‌کنیم. ادعا می‌کنیم که G' روی مجموعه‌ی رئوس U شبه‌تطابق مجزای رأسی با حداکثر طول مسیر k دارد اگر و فقط اگر G روی U شبه‌تطابق مجزای یالی با حداکثر طول مسیر $k-1$ داشته باشد. دلیل این موضوع آن است که یک مسیر مجزای رأسی به طول k در G' که دو رأس انتهاییش در U باشند، متناظر یک مسیر مجزای یالی به طول $k-1$ در G است. □

نتیجه ۱. با توجه به این که مسئله‌ی شبه‌تطابق (مجزای رأسی، بی‌جهت، $3 \leq$ طول مسیرها) در P قرار دارد بنا بر قضیه‌ی بالا مسئله‌ی (مجزای یالی، بی‌جهت، $2 \leq$ طول مسیرها) در P قرار دارد.

قضیه ۱۸. مسئله‌ی شبه‌تطابق (مجزای یالی، جهت‌دار، $k \geq ۲$ طول مسیرها) این شرط را داشته باشیم که مسیرها نمی‌توانند از رئوس U بگذرند، الگوریتمی چندجمله‌ای برای مسئله وجود دارد.

اثبات. از گراف G، نخست گراف $G_۱$ را به این ترتیب می‌سازیم که به ازای هر دو رأس u و v در U که به‌هم وصلند، یال $u \to v$ را حذف می‌کنیم و به‌جای آن رأس w و یال‌های $u \to w$ و $w \to v$ را به گراف می‌افزاییم (به‌عبارت دیگر یک رأس روی یال قرار می‌دهیم)، ضمناً در گراف $G_۱$ یال‌های بین رئوس $V(G) - U$ را حذف می‌کنیم. حال گراف بی‌جهت G' را از روی گراف جهت‌دار $G_۱$ به این ترتیب می‌سازیم که به‌ازای هر رأس در U، یک رأس در G' قرار می‌دهیم و در آن رأس u را به v وصل می‌کنیم، اگر و فقط اگر در $G_۱$ رأس w وجود داشته باشد که یال‌های $u \to w$ و $w \to v$ موجود باشند، حال گراف G، شبه‌تطابق با شرایط مسئله دارد اگر و فقط اگر گراف G' تطابق کامل داشته باشد. باتوجه به چگونگی ساختن G' و مجزای یالی بودن مسیرهای شبه‌تطابق علت این امر روشن است. بدین ترتیب با یافتن تطابق کامل در G' مسئله‌ی مطلوب برای گراف G حل می‌شود. □

توضیح ۲. می‌دانیم که مسئله‌ی تطابق روی یک مجموعه‌ی U در گراف را می‌توان با گذاشتن یک رأس روی هر یال به مسئله‌ی شبه‌تطابق با مسیرهای به طول دو تبدیل کرد و به این ترتیب چون مسئله‌ی تطابق P-Complete است، پس مسئله‌ی شبه‌تطابق هم P-Complete است.

۴.۴ NP-Complete بودن شبه‌تطابق محدود

در این قسمت می‌خواهیم مسئله‌ی شبه‌تطابق (مجزای یالی، بی‌جهت، $k \leq$ طول مسیرها) را برای $k \geq ۹$ ثابت، بررسی کنیم. برای این کار از مسئله‌ی زیر استفاده می‌کنیم:

مسئله A: گراف $G = (V, E)$ با رئوس ابتدا و انتهای مشخص s و t و با اعداد صحیح $J, K \leq |V|$ داده شده‌اند. مسئله این است که آیا G، شامل J مسیر مجزای رأسی از s به t است که

هیچ‌کدام از این مسیرها بیش از K یال نداشته باشند.

قضیه ۱۹. [۳] مسئله‌ی A، NP-Complete است.

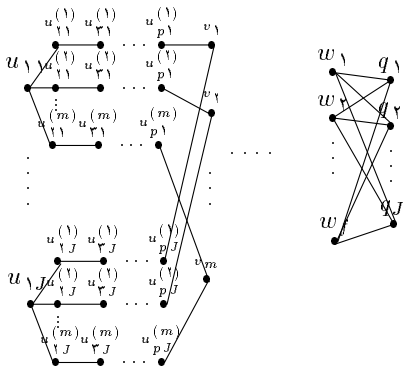
توضیح ۳. مسئله‌ی A در حالت ورودی k ی ثابت که $k \geq ۵$، NP-Complete باقی می‌ماند. این مسئله در حالت $k \leq ۴$ در زمان چندجمله‌ای دارد.

مسئله‌ی مشابه A که آن در مسئله B می‌نامیم و به‌جای مسیرهای مجزای رأسی در مورد مسیرهای مجزای یالی است، هم NP-Complete می‌باشد[۳]. مسئله‌ی B در حالت k ی ثابت $k \geq ۵$ هم NP-Complete است و در حالت $k \leq ۳$ در کلاس P قرار دارد و در حالت $k = ۴$ مسئله باز است[۳].مسئله‌ی مشابه بالا در حالت گراف جهت‌دار با مسیرهای جهت‌دار هم NP-Complete است و نتایج مشابهی در این رابطه وجود دارد[۳]. ما مسئله بالا را در حالت جهت‌دار مجزای رأسی مسئله‌ی C و در حالت جهت‌دار مجزای یالی مسئله‌ی D می‌نامیم.

در زیر اثباتی از NP-Complete بودن مسئله‌ی شبه‌تطابق با طول مسیر محدود داده شده است که هم برای هر مقدار ثابت $k \geq ۹$ درست کار می‌کند، این اثبات در حالت جهت‌دار هم درست کار می‌کند. توجه داریم که اثبات قبلی این قضیه برای $k = ۷$ و فقط در حالت بی‌جهت‌دار و مسیرهای مجزای یالی درست کار می‌کرد[۲۴].

قضیه ۲۰. مسئله‌ی یافتن شبه‌تطابق با این محدودیت که طول مسیرها حداکثر می‌تواند k باشد، برای هر k ی ثابت که $k \geq ۹$، NP-Complete است.

اثبات. این اثبات راه حل دیگری از مسئله‌ی شبه‌تطابق با محدودیت طول است که برای هر k ی ثابت ($k \geq ۹$) هم درست کار می‌کند. NP بودن مسئله بدیهی است. یک مثال از مسئله‌ی B را به



شکل ۱.۴: گراف ساخته‌شده

.

مثالی از مسئله‌ی خودمان تبدیل می‌کنیم. گراف G و دو رأس s و t به‌همراه اعداد k و J داده شده‌اند و مسئله‌ی B آن است که J مسیر مجزای یالی بین s t بیابیم که طول آن‌ها حداکثر k باشد. گراف G' را از گراف G به این ترتیب می‌سازیم، ابتدا رئوس s و t را از G حذف می‌کنیم؛ همسایه‌های s را v_1، v_2، $\cdots$، v_m در نظر می‌گیریم. به‌ازای هر v_j و هر i ($1 \leq i \leq J$ و $1 \leq j \leq m$)، رئوس $u_{1i}^{(j)}$، $u_{2i}^{(j)}$، $\cdots$، $u_{pi}^{(j)}$، را به G' می‌افزاییم ($p = k$) و یال‌های $u_{1i}^{(j)}u_{2i}^{(j)}$، $u_{2i}^{(j)}u_{3i}^{(j)}$، $\cdots$، $u_{(p-1)i}^{(j)}u_{pi}^{(j)}$ و یال $u_{pi}^{(j)}v_j$ را به G' به ازای هر i و j می‌افزاییم، سپس تمام رئوس $u_{1i}^{(j)}$ ها را به‌ازای هر i ثابت و تمام j ها در هم ادغام می‌کنیم تا از ادغام آن‌ها رئوس u_{11}، u_{12}، u_{13}، $\cdots$، u_{1J} به‌دست آید که به این ترتیب رأس u_{1i} به رئوس $u_{2i}^{(j)}$ ($1 \leq j \leq m$) متصل است. همسایه‌های t را w_1، w_2، $\cdots$، w_f می‌نامیم، رئوس q_1، q_2، $\cdots$، q_J را به G' اضافه می‌کنیم و q_i ها و w_j ها را دوبه‌دو به‌هم وصل می‌کنیم. شکل ۱.۴ چگونگی ساختن این گراف را نشان می‌دهد.

مجموعه‌ی U در گراف G' برابر رئوس سر مسیرهای p تایی اضافه شده به G' به‌همراه این رئوس جدید تازه‌اضافه شده (q_i ها) است ($U = \{u_{1i} | 1 \leq i \leq J\} \bigcup \{q_i | 1 \leq i \leq J\}$). ثابت می‌کنیم یافتن شبه‌تطابق مجزای یالی در گراف G' روی مجموعه‌ی رئوس U با این محدودیت که طول مسیرها حداکثر برابر $2k - 1$ باشد معادل حل مسئله‌ی B است (برای گراف ورودی G و پارامترهای ورودی k و J). زیرا در صورت وجود یک شبه‌تطابق با مسیرهای حداکثر به طول $2k - 1$، هر کدام از این مسیرها

حتماً از یک u_{1j} به یک w_q است، زیرا دو رأس u_{1j} و $u_{1j'}$ نمی‌توانند مسیر حداکثر $2k-1$ یالی به‌هم داشته باشند، چون طول هر مسیر بین دو تا از آن‌ها حداقل $k+k=2k$ است. حال یک مسیر $2k-1$ تایی از یک u_{1i} به یک w_j متناظر یک مسیر به طول $k-2$ از یک v_q به یک w_l است و چنین مسیری متناظر یک مسیر به طول k از s به t در G می‌باشد. پس مسیرهای شبه‌تطابق متناظر مسیرهای حداکثر به طول k و مجزای یالی از s به t می‌باشند، برعکس مسیرهای مجزای یالی به طول حداکثر به طول k از s به t، متناظر مسیرهای حداکثر $k-2$ تایی از همسایه‌های s به همسایه‌های t است و متناظر مسیرهای حداکثر به طول $2k-1=k+k-2+1$ از u_{1i}ها به w_qها در G' می‌باشد. توجه داریم که مسئله‌ی B در حالت k ی ثابت $(k\geq 5)$، NP-Complete است. دیدیم که مسئله در حالت $k=5$ به $k'=2k-1=9$ در شبه‌تطابق تبدیل می‌شود که نشان می‌دهد، مسئله شبه‌تطابق با طول محدود مسیرها برای kهای فرد که $k\geq 9$، NP-Complete است. حال اگر در استدلال بالا به‌جای $p=k$ قرار دهیم $p=k+1$، به‌طور مشابه خواهیم داشت: در گراف جدید شبه‌تطابق مجزای یالی با مسیرهای به طول حداکثر $2k$ وجود دارد اگر و فقط اگر مسئله‌ی B جواب داشته باشد. به این ترتیب مسئله‌ی شبه‌تطابق (مجزای یالی، بی‌جهت، $k\leq$ طول مسیرها) برای $k\geq 10$ ی ثابت زوج NP-Complete است. پس در کل این مسئله در حالت $k\geq 9$ ی ثابت NP-Complete است. □

نتیجه ۲. باتوجه به مسئله‌ی بالا و قضیه‌ای در [۲۴]، مسئله‌ی شبه‌تطابق (مجزای یالی، بی‌جهت، $k\leq$ طول مسیرها) برای $k\geq 7$ ی ثابت و $k\neq 8$، NP-Complete است.

قضیه ۲۱. مسئله‌ی شبه‌تطابق (مجزای یالی، جهت‌دار، $k\leq$ طول مسیرها) برای $k\geq 5$ ی ثابت، NP-Complete است.

اثبات. مشابه تبدیل بالا را برای مسئله‌ی D انجام می‌دهیم با این تفاوت که در این‌جا قرار می‌دهیم $p=1$ یا به عبارت دیگر بعد از حذف s و t از گراف رئوس u_1، u_2، $\cdots$، u_J را به گراف می‌افزاییم و از هر کدام از این رئوس یال جهت‌داری به هر یک از v_iها قرار می‌دهیم و رئوس q_1، q_2، $\cdots$، q_J را به

گراف می‌افزاییم و از هر کدام از w_i ها به هر یک از این رئوس یال جهت‌داری قرار می‌دهیم. حال وجود شبه‌تطابق جهت‌داری با مسیرهای به طول حداکثر k معادل وجود J مسیر مجزای یالی جهت‌دار از s به t است. □

قضیه ۲۲. مسئله‌ی یافتن شبه‌تطابق مجزای رأسی با این محدودیت که طول مسیرها حداکثر می‌تواند k باشد، برای هر k ی ثابت که $k \geq ۹$، NP-Complete است.

اثبات. نخست دقت می‌کنیم که بنا بر قضیه‌ی ۱۷ و قضیه‌ی ۲۰ ثابت می‌شود که مسئله برای k ی ثابت که $k \geq ۱۰$، NP-Complete است. اما کاملاً مشابه قضیه‌ی ۲۰ و با استفاده از مسئله‌ی A به‌جای مسئله‌ی B می‌توان ثابت کرد که مسئله‌ی شبه‌تطابق (مجزای رأسی، بی‌جهت، $k \leq$ طول مسیرها) برای $k \geq ۹$ ی ثابت NP-Complete است. □

نتیجه ۳. مشابه قضیه‌ی ۲۱ می‌توان ثابت کرد که مسئله‌ی شبه‌تطابق (مجزای رأسی، جهت‌دار، $k \leq$ طول مسیرها) برای $k \geq ۵$ ی ثابت، NP-Complete است.

$l < k$	$۱ \leq k \leq ۳$	$k = ۴$	$k = ۵$	$۶ \leq k \leq ۷$	$k = ۸$	$k = ۹$	$k > ۹$
[D, E, $l < k$]	P	?	?	NPC	NPC	NPC	NPC
[D, V, $l < k$]	P	?	?	NPC	NPC	NPC	NPC
[U, E, $l < k$]	P	?	?	?	NPC	?	NPC
[U, V, $l < k$]	P	P	?	?	?	NPC	NPC

جدول ۱.۴: وضعیت حل مسایل.

۵.۴ مسایل حل‌نشده

مسئله ۱. مسئله‌ی شبه‌تطابق (مجزای یالی، بی‌جهت، $k \leq$ طول مسیرها) برای $k = ۳, ۴, ۵, ۶, ۸$ در P است یا در NPC قرار دارد؟

مسئله ۲. مسئله‌ی شبه‌تطابق (مجزای رأسی، بی‌جهت، $k \leq$ طول مسیرها) برای $k = ۴, ۵, ۶, ۷, ۸$ در P است یا در NPC قرار دارد؟

مسئله ۳. مسئله‌ی شبه‌تطابق (مجزای یالی، جهت‌دار، $k \leq$ طول مسیرها) برای $k = ۳, ۴$ در P است یا در NPC قرار دارد؟

مسئله ۴. مسئله‌ی شبه‌تطابق (مجزای رأسی، جهت‌دار، $k \leq$ طول مسیرها) برای $k = ۳, ۴$ در P است یا در NPC قرار دارد؟

جدول ۱.۴ وضعیت حل مسایل را نشان می‌دهد. با توجه به مطالبی که در [۳] در مورد مسایل A، B، C، و D آمده است، به‌نظر می‌رسد از ایده و روش حل این مسایل می‌توان در مسئله‌ی شبه‌تطابق هم استفاده کرد ولی ما هنوز نتوانسته‌ایم این مرجع را پیدا کنیم. این مسایل، مسایل باز و کاربردی‌یی هستند که علاقه‌مندان می‌توانند روی آن کار کنند.

فصل ۵

خلاصه‌ای از دیگر کارهای انجام‌شده

هدف از این فصل ارایه‌ی دیگر کارهای انجام‌شده در زمینه‌ی شبه‌تطابق و تعمیم‌های آن می‌باشد. این فصل خلاصه‌ی چند مقاله بشرح زیر است که به‌دلیل کثرت منابع، از ذکر منابع این مقالات خودداری کرده و هرجا که منبعی لازم است، باید به اصل مقالات رجوع شود:

برای اطلاعات بیشتر در مورد بخش اول به [۲۵]، بخش دوم به [۲۱]، بخش سوم به [۲۲]، بخش چهارم به [۲۰]، بخش پنجم به [۱۸]، بخش ششم به [۲۷]، بخش هفتم به [۲۸] و بخش هفتم به [۲۸] مراجعه نمایید. هم‌چنین دیگر مقالات مرتبط که در این بخش از آن‌ها استفاده شده است، عبارتند از: [۱۷]، [۲۶]، [۱۱] و [۱۳].

۱.۵ شبه‌تطابق در گراف‌ها

در این مقاله مفهومی بنام تطابق‌مسیری [1] معرفی شده است که دقیقاً همان مفهوم شبه‌تطابق می‌باشد که پیشتر در مورد آن مفصل سخن گفتیم. لازم به‌ذکر است که این دو نام شبه‌تطابق و تطابق‌مسیری در جاهای مختلف به‌جای هم به‌کار رفته‌اند. هم‌چنین گراف‌های مورد بحث در این بخش همگی ساده می‌باشند.

[1] Path-matching

تعریف ۸. یک k–شبه‌تطابق، شبه‌تطابقی است که در آن k جفت راس با مسیرهای مجزای یالی به‌هم مرتبط شده باشند و هم‌چنین شبه‌تطابق کامل، شبه‌تطابقی است که همه‌ی راس‌ها(بجز حداکثر یک راس) در شبه‌تطابق شرکت داشته باشند.

در این بخش علامت $\alpha'(G)$ را برای نشان دادن اندازه‌ی بیشترین تطابق در گراف G به‌کار می‌بریم.هم‌چنین گراف G را زوج می‌نامیم اگر تمام مولفه‌های آن تعداد زوجی راس داشته باشند. قضایای زیر را که در فصول قبلی نیز ذکر کرده‌ایم، برای ارجاعات بعدی مجدداً می‌آوریم:

قضیه ۲۳. هر گراف همبند با تعداد رئوس زوج یک شبه‌تطابق دارد که تمام مسیرهای آن به طول حداکثر ۲ می‌باشند.

نتیجه ۴. یک گراف G یک شبه‌تطابق کامل دارد اگر وفقط اگر زوج باشد.

چون هر گرافی یک شبه‌تطابق کامل دارد ولی لزوماً یک تطابق کامل ندارد، سوال طبیعی که مطرح می‌شود، این است که حداقل تعداد یال‌ها در یک شبه‌تطابق کامل چقدر است که ما را به تعریف زیر رهنمون می‌شود.

تعریف ۹. در هر گراف زوج G، کمترین تعداد یال‌ها در یک شبه‌تطابق کامل G را با $\pi(G)$ نشان می‌دهیم که عدد شبه‌تطابقی G نامیده‌می‌شود. یک شبه‌تطابق کامل، مینیمال نامیده می‌شود اگر $\pi(G)$ یال داشته باشد.

واضح است که $\frac{n}{2} \leq \pi(G) \leq n-1$. هم‌چنین اگر گراف عروس $B_{n,m}$ که یک درخت با یک ستاره m یالی و یک مسیر $n-m$ راسی خارج شده از مرکز ستاره است، را در نظر بگیرید، خواهید دید که $\pi(B_{n,m}) = \lfloor \frac{n+m}{2} \rfloor$ که با انتخاب مقدار مناسب m می‌تواند هر مقدار دل‌خواه در بازه یاد شده را بگیرد.

قضیه ۲۴. برای هر گراف زوج G_n با تعداد رئوس n و c مولفه‌ی همبندی داریم:
$$n - \alpha'(G_n) \leq \pi(G_n) \leq n - c$$

اثبات. اثبات طرف دوم با توجه به‌نحوه اثبات قضیه ۲۳ واضح است. طرف اول نیز با توجه به این نکته که حداکثر $\alpha'(G)$ مسیر می‌توانند در شبه‌تطابق کامل طول ۱ داشته باشد، واضح است. □

در زیربخش بعدی، نتایجی را در مورد عدد شبه‌تطابق گراف‌ها می‌آوریم. ما ابتدا نشان خواهیم داد که مجموعه‌ی یال‌ها در هر شبه‌تطابق مینیمال G، بدون دور است و این نکته نتیجه می‌دهد که هر گراف زوج یک شبه‌تطابق کامل مینیمال با مسیرهایی با طول حداکثر دو دارد. آن‌گاه قضیه‌ی بعدی یک حد بالای مرزی براساس تعداد یال‌ها و راس‌های G برای $\pi(G)$ می‌دهد. مساله طبیعی بعدی بیشینه کردن تعداد مسیرهای با طول یک در شبه‌تطابق کامل است. ما ثابت می‌کنیم که حداقل $\lceil \frac{\alpha'(G)}{2} \rceil$ مسیر با طول یک می‌تواند در یک شبه‌تطابق کامل انتخاب شود و این یک حد بسیار خوب براساس $\alpha'(G)$ می‌باشد.

در زیر بخش آخر مفهوم شبه‌تطابق دوبخشی را معرفی می‌کنیم و نتایجی را بیشتر برای درخت‌ها ارایه می‌کنیم. هر چند که این مفهوم می‌تواند در قالب شبکه‌ی جریان بررسی شود، هدف ما روشن کردن شرایط ساده‌تری برای وجود شبه‌تطابق دوبخشی کامل در درخت‌ها می‌باشد.

۱.۱.۵ شبه‌تطابق‌های کامل

یک خاصیت اصلی شبه‌تطابق‌های مینیمال عبارت است از:

قضیه ۲۵. مسیرها در شبه‌تطابق کامل مینیمال هر گراف زوج، یک زیر گراف بدون دور (جنگل) به‌وجود می آورند.

اثبات این قضیه تا حدود زیادی شبیه قضیه مشابه در فصول قبل می‌باشد.

نتیجه ۵. هر گراف زوج همبند G یک درخت پوشای T دارد به‌طوری‌که $\pi(G) = \pi(T)$.

قضیه ۲۶. به وضوح شرط $\pi(T) \geq \pi(G)$ برای هر درخت پوشای T از G صادق است. برای اثبات طرف دیگر نامساوی، گراف H که با یال‌های مسیرهای شبه‌تطابق کامل مینیمال G پوشیده شده است را در نظر بگیرید. برطبق قضیه قبل H بدون دور است، لذا H می‌تواند به یک درخت پوشای T از G توسعه یابد. پس $\pi(G) = \pi(H) \geq \pi(T)$.

نتیجه ۶. هر گراف زوج G یک شبه‌تطابق کامل مینیمال دارد که در آن تمام مسیرها به طول حداکثر ۲ می‌باشند.

اثبات. گراف H ایجادشده از شبه‌تطابق مینیمال کامل G بدون دور است و لذا زوج است. با به‌کاربردن قضیه ۲۳ برای هر مولفه از H، شبه‌تطابق کامل خواسته شده حاصل می‌شود. □

خواسته بعدی ما ارایه‌کردن حد پایین‌تر از $n-1$ برای حداکثر تعداد یال‌ها در شبه‌تطابق مینیمال کامل می‌باشد. قضیه زیر را داریم:

قضیه ۲۷. اگر G یک گراف زوج با n راس و حداقل n یال باشد، آن‌گاه $\pi(G) \leq n-2$ و یک شبه‌تطابق کامل G با حداکثر $n-2$ یال و مسیرهایی با طول حداکثر ۲ وجود دارد.

فرض کنید $f(n,k)$ را به‌صورت زیر تعریف کنیم:

$$f(n,k) = max\{\binom{2k}{2} + n - 2k, \binom{k}{2} + k(n-k)\}$$

با استفاده از فرمول تات—برگ[۲] برای تطابق بیشینه گراف می‌توان ثابت کرد که اگر گراف زوج n راسی، $f(n,k)+1$ یال داشته باشد آن‌گاه $\alpha'(G) \geq k+1$ خواهد بود. حال قضیه زیر را که از قضیه قبل نیز قویتر است، داریم.

قضیه ۲۸. فرض کنید که k و n اعداد صحیح، n زوج و $1 \leq k < \frac{n}{4}$ باشد، اگر G یک گراف زوج با تعداد رئوس n و تعداد یال‌های $f(n,k)+1$ باشد، آن‌گاه $\pi(G) \leq n-k-1$ است. بعلاوه یک

[۲] Tutte-Berge

شبه‌تطابق کامل از G با حداکثر $n-k-1$ یال با استفاده از مسیرهایی به طول حداکثر ۲ وجود دارد. نتیجه برای تمام n های زوج و k های در بازه‌ی ذکرشده، مرزی [۳] است.

اثبات. برای دیدن این‌که مقدار $f(n,k)$ درحالت کلی برای برقراری شرط $\pi(G) \leq n-k-1$ کافی نیست، کافی است از قضیه‌ی ۲۴ که در آن ثابت کردیم $\pi(G) \geq n-\alpha'(G)$ استفاده کنیم. برای اثبات استقرایی $\pi(G) \leq n-k-1$ نیاز به تعدادی تعریف و ادعا داریم که از ذکر آن‌ها خودداری می‌کنیم. برای اثبات دقیق می‌توانید به [۲۵] مراجعه کنید. □

هدف بعدی ما دیدن این مطلب ست که بدانیم چند یال تنها[۴] می‌توانند در یک شبه‌تطابق کامل از یک گراف زوج شرکت داشته باشند. حالت کلی می‌تواند به‌راحتی در مورد درخت‌ها، بیان شود. فرض کنید T یک درخت زوج است. آن‌گاه به‌وضوح حداکثر $\alpha'(G)$ یال تنها، می‌توانند در مسیرهای شبه‌تطابق کامل T ظاهر شوند.

مثال زیر نشان می‌دهد که در حالت کلی حداکثر $\lceil \frac{\alpha'(T)}{2} \rceil$ یال تنها می‌توانند در یک شبه‌تطابق کامل بیایند.

مثال ۲. درخت زیر را با $5k+2$ راس در نظر بگیرید (برای هر k مثبت زوج). یک ستاره[۵] S با k یال و k نسخه از یک پنجه [۶] (یک ستاره با سه یال) را در نظر بگیرید که همه از هم متمایزند. هر راس انتهایی S را به یک راس انتهایی یک نسخه پنجه وصل کنید. سپس یک یال و یک راس به مرکز S اضافه کنید. به‌راحتی می‌توان چک‌کرد که $\alpha'(T) = 2k+1$ و هر تطابق کامل T حداکثر $k+1$ یال تنها دارد.

قضیه زیر نشان می‌دهد که مثال بالا بهترین حداست.

---

[۳] Sharp

[۴] Single edge

[۵] Star

[۶] Claw

قضیه ۲۹. اگر T یک درخت زوج باشد، آن‌گاه T یک شبه‌تطابق کامل با حداقل $\lceil \frac{\alpha'(G)}{۲} \rceil$ یال تنها دارد.

اثبات با استقرا روی $\alpha'(T)$ صورت می‌گیرد.

با توسعه‌ی یک تطابق بیشینه از یک گراف زوج به یک جنگل پوشا، می‌توان سریعاً نتیجه زیر را گرفت.

نتیجه ۷. هر گراف زوج G یک شبه‌تطابق کامل با حداقل $\lceil \frac{\alpha'(G)}{۲} \rceil$ یال تنها دارد.

۲.۱.۵ شبه‌تطابق‌های کامل دوبخشی

شبه‌تطابق‌ها در درخت‌ها ویا گراف‌های دوبخشی در حالت کلی، خاصیت دوبخشی بودن گراف را در نظر نمی‌گیرند، لذا تعریف زیر را داریم:

تعریف ۱۰. اگر B یک گراف دوبخشی باشد، آن‌گاه یک شبه‌تطابق دوبخشی یک شبه‌تطابق است که تمام رئوس انتهایی مسیرها در دو بخش متفاوت باشند. عبارات k—شبه‌تطابق دوبخشی و شبه‌تطابق دوبخشی کامل هم به‌صورت قبل تعریف می‌شوند.

به‌وضوح، شبه‌تطابق و شبه‌تطابق دوبخشی در مورد گراف‌های دوبخشی یکسان نیستند. هر درخت با تعداد رئوس زوج، یک شبه‌تطابق کامل دارد ولی ممکن است که شبه‌تطابق دوبخشی کامل نداشته باشد. برای مثال یک ستاره با حداقل ۳ یال، شبه‌تطابق دوبخشی کامل ندارد. هم‌چنین، تطابق‌ها و شبه‌تطابق‌های دوبخشی هم یکسان نیستند. چنان‌که قضیه زیر را داریم:

قضیه ۳۰. اگر B یک گراف دو بخشی با n راس در هر بخش باشد که یک شبه‌تطابق دوبخشی کامل داشته باشد، آن‌گاه B یک تطابق با حداقل $\lceil ۲\sqrt{n+۱} - ۲ \rceil$ یال دارد. هم‌چنین وقتی‌که $n+۱$ مربع کامل باشد، این حد مرزی [۷] است.

---

[۷] Sharp

اگر B یک گراف دوبخشی با بخش‌های X و Y باشد و G گرافی باشد که با اضافه کردن دو راس x و y که در آن $x(y)$ با تمام رئوس X (رئوس Y) همسایه است، به‌دست آمده باشد، آن‌گاه واضح است که B یک k–شبه‌تطابق دوبخشی دارد اگر وفقط اگر k مسیر مجزای یالی از x به y وجود داشته باشند و این راه را برای استفاده شبکه‌ی جریان باز می‌کند [۹]، ولی بدون استفاده از آن نیز می‌توان قضایای زیر را اثبات کرد.

تعریف ۱۱. فرض کنید T یک درخت زوج باشد. یک یال e از T زوج (فرد) نامیده می‌شود اگر دومولفه $T-e$ تعداد زوج (فرد)ی راس داشته باشند. یال e تقریباًمتعادل نامیده می‌شود اگر هر بخش $T-e$، یا(t,t) ویا یک $(t,t+1)$–گراف دوبخشی باشند. یال e متعادل نامیده می‌شود اگر تقریباً متعادل بوده و در ضمن اگر گراف دو بخشی $(t,t+1)$ به‌وجود آید، راس مجاور e در مولفه با تعداد بیشتر راس بیافتد. (منظور از (h,m)–گراف دوبخشی، گراف دوبخشیی است که بخش‌هایش m و n راسی باشند.)

حال قضایای زیر را داریم که بدون اثبات آن‌ها را می‌آوریم:

قضیه ۳۱. اگر درخت T زوج باشد، یک تطابق کامل دارد اگر وفقط اگر هر یال T، متعادل باشد.

قضیه ۳۲. اگر T_n درختی باشد که در آن هر یال فرد است، آن‌گاه T_n یک شبه‌تطابق دوبخشی کامل دارد اگر وتنها اگر هر یال T تقریباً متعادل بوده و $n \equiv ۲(mod ۴)$

قضیه ۳۳. اگر T_n یک درخت با تعداد رئوس زوج باشد، آن‌گاه T_n یک شبه‌تطابق دوبخشی کامل دارد اگر وتنها اگر هر یال آن تقریباً متعادل بوده و هر زیر درخت به‌وجود آمده از حذف یال‌های زوج دارای تعداد رئوس ۲ به‌هنگ ۴ باشد.

از قضایای بالا، قضیه اصلی زیر نتیجه می‌شود:

قضیه ۳۴. اگر T_n یک درخت باشد که در آن هر یال فرد و تقریباً متعادل باشد، آن‌گاه برای هر k با شرط $\frac{n+۲}{۴} \leq k \leq \alpha'(T_n)$، یک شبه‌تطابق دوبخشی کامل با k مسیر بطول ۱ وجود دارد.

۲.۵ مسیریابی چندرنگی در ابرمکعب بدون‌جهت

فرض کنید که شبکه‌ی اتصال داخلی با یک گراف بدون‌جهت G مدل شده است. ما هم‌چنین مسیریابی‌های خواسته شده در G را با گراف بدون‌جهت R برروی مجموعه رئوس $V(G)$ مدل می‌کنیم. یک یال $\{u,v\}$ از R، یک مسیر بدون‌جهت بین u و v در G نیاز دارد. لذا یک جواب مساله‌ی مسیریابی (G,R) یک مجموعه‌ی P از مسیرهای بدون جهت از G می‌باشد به‌طوری‌که برای هر یال $\{u,v\}$ در R، P شامل یک مسیر بین u و v باشد. در این حالت ما می‌گوییم که P، R را در G تحقیق می‌بخشد. هم‌چنین ما G را گراف میزبان و R را گراف مسیریابی می‌نامیم. چون همیشه مجموعه رئوس گراف مسیریابی R همان مجموعه رئوس گراف G است ما فقط یال‌های R را مشخص می‌کنیم.

در این بخش روی حالتی که گراف R یک تطابق است تاکید شده است. این یک حالت خاص مهم است چون با قضیه رنگ‌آمیزی یالی Vizing هر گراف مسیریابی با بیشترین درجه d می‌تواند به d یا $d+۱$ تطابق تجزیه شود.

با خاصیت خواسته شده یک جواب برای مساله‌های مسیریابی بالا، این است که مسیرهای جواب دوبه‌دو مجزای یالی باشند. یعنی یال‌ها در حداکثر یک مسیر از مجموعه مسیرها ظاهر شوند. لذا یک سوال طبیعی آن است که بپرسیم آیا هر تطابق روی $V(G)$ می‌تواند در G توسط مسیرهای مجزای یالی تحقق یابد. متاسفانه، جواب این سوال برای اکثر شبکه‌های ارتباطی پیشنهاد شده، منفی است. لذا ما خواسته‌ی مجزای یالی بودن را رها کرده و می‌پرسیم که چند مجموعه مسیرهای مجزای یالی برای تحقق یک تطابق لازم است. به‌طور دقیق‌تر خواسته این سوال، در مورد مسیریابی چندرنگی می‌باشند. ما یک مجموعه از مسیرهای G را k—رنگ‌پذیر می‌گوییم، اگر هر مسیر در این مجموعه بتواند با یکی از k رنگ، رنگ شود به‌طوری‌که مسیرهایی با رنگ یکسان، مجزای یالی باشند. ما می‌گوییم گراف مسیریابی با k رنگ تحقق یافته است اگر با یک مجموعه ازمسیرهای k—رنگ‌پذیر تحقق یافته باشد. لذا سوال ما می‌تواند این‌گونه مطرح شود که کوچکترین kی که هر تطابق در $V(G)$ بتواند با k رنگ

تحقق یابد، چند است؟

ما سوال بالا را در مورد ابرمکعب که یکی از توپولوژی‌های بسیار معروف شبکه‌های اتصال داخلی است بررسی می‌کنیم. منظور از ابرمکعب همان مکعب n—بعدی (Q_n) است. نشان داده شده است که هر تطابقی، اگر n زوج باشد، نمی‌تواند در Q_n تحقق یابد. برای مثال، فرض کنید $A_n = \{\{v, \overline{v}\} | v \in V(Q_n)\}$ یک تطابق متقاطر باشد که دراین‌جا $\overline{v}$ راسی است که با نقیض کردن تمام بیت‌های v به‌دست آمده است(و لذا فاصله‌اش از v برابر n است). به‌راحتی دیده می‌شود که $A_۲$ نمی‌تواند توسط یک رنگ در $Q_۲$ تحقق یابد و هم‌چنین نشان داده شده است که A_n اگر n زوج باشد، نمی‌تواند با یک رنگ تحقق یابد. کار Aumann و Rabani برروی مسیریابی جایگشتی نشان داد که هر تطابق روی $V(Q_n)$ می‌تواند با ۱۶ رنگ در Q_n تحقق یابد. در این مقاله نشان داده‌شده است که این هدف می‌تواند با ۴ رنگ نیز برآورده شود [۲۱].

لازم به‌ذکر است که این‌گونه مسایل مسیریابی از نوع مسیریابی تعویض مدار [۸] است که در آن یک مسیر به یک جفت گره که می‌خواهند باهم ارتباط برقرارکنند، تخصیص می‌یابد و متضاد با مسیریابی تعویض بسته [۹] است که ارتباط با ارسال یک بسته انجام شده و هر یال مسیر به‌محض این‌که بسته از آن گذشت آزاد می‌شود. در مسیریابی تعویض مدار، مجزای یالی بودن مسیرهای ارتباطی در صورتی‌که هر یال بتواند فقط یک کانال ارتباطی را پشتیبانی کند، لازم است. اگر شبکه بتواند طوری مالتی پلکس کند که k کانال را پشتیبانی کند (مثلاً با طول موج‌های متفاوت و یا با تقسیم زمانی) آن‌گاه ما می‌توانیم خواسته‌ی مجزای یالی بودن را با k—رنگ‌پذیر بودن عوض کنیم.

مساله مسیریابی مجزای یالی یا مسیریابی چندرنگی در مورد گراف‌های جهت‌دار هم تحت عنوان مدل مسیریابی جهت‌دار بررسی شده است. یک مساله مسیریابی جهت‌دار یک جفت (G, R) است که هر دو گراف میزبان و مسیریابی جهت‌دار باشند. مسیریابی چندرنگی در مدل جهت‌دار نیز مشابه مدل بدون‌جهت تعریف شده است. ما یک گراف مسیریابی جهت‌دار R را یک جایگشت [۱۰] می‌نامیم، اگر درجه ورودی و خروجی هر راس دقیقاً یک باشد. این حالت خاص مسایل مسیریابی

[۸] Circuit-switched routing

[۹] Packet-switched routing

[۱۰] Permutation

جهت‌دار، همان حالتی را داراست که تطابق‌ها در حالت بدون‌جهت دارا می‌باشند.

H_n را ابرمکعب جهت‌دار تعریف می‌کنیم که $V(H_n) = V(Q_n)$ و هر یال بدون‌جهت Q_n با یک جفت یال با جهت‌های متضاد در H_n تعویض شده‌اند. Szymanski حدس زد که هر جایگشت در H_n می‌تواند با مسیرهای جهت‌دار مجزای یالی تحقق یابد. این حدس تاکنون بازمانده است. در حال حاضر نشان داده شده است که هر جایگشت می‌تواند در H_n با ۲ رنگ تحقق یابد. لازم به‌ذکر است که اگر حدس Szymanski درست باشد، یک پهنای باند حدی [۱۱] است. جایگشت متقاطر $B_n = \{(v, \overline{v}) | v \in V(H_n)\}$ روی $V(H_n)$ را در نظر بگیرید. هر راه‌حل برای مساله مسیریابی (H_n, B_n) باید حداقل $n * ۲^n$ یال داشته باشد که دقیقاً تمام یال‌های H_n است. در حالت بدون‌جهت ما تاکنون نتیجه‌ای به این اندازه دقیق و حدی درمورد پهنای باند نداشته‌ایم، زیرا همانطور که ذکر شد تطابق متقاطر نمی‌تواند با یک رنگ در ابرمکعب بدون‌جهت زوج‌بعدی تحقق یابد. از طرفی اگر ما پهنای باند H_n را با تعویض هر یال جهت‌دار H_n با دویال جهت‌دار در همان جهت، دوبرابر کنیم، آن‌گاه دوجایگشت (یک گراف مسیریابی جهت‌دار که درجه ورودی و خروجی هر راس در آن دقیقاً دو می‌باشد) می‌تواند با یک رنگ در آن تحقق یابد. در واقع حدس Szymanski یک حالت میانی بین این دو قضیه است. ما احساس می‌کنیم که بهبود نتیجه این مقاله به دو رنگ به‌جای چهار رنگ نیاز به‌روش جدیدی غیر از روش این مقاله دارد ولی در مورد سه‌رنگ هنوز این روش جای بهبود دارد.

Aumann وRabani هم‌چنین مسایلی را که در آن گراف میزبان، ابرمکعبی بدون‌جهت است ولی گراف مسیریابی یک گراف جهت‌دار و در حالت خاص یک جایگشت است بررسی کرده‌اند. آن‌ها نشان داده‌اند که هر جایگشت می‌تواند با ۱۶ رنگ روی ابرمکعب بدون‌جهت تحقق یابد ولی چون هر جایگشت متناظر دوتطابق است، نتیجه ما حد ارایه شده توسط اینان را به ۸ رنگ به‌جای ۱۶ رنگ تقلیل می‌دهد [۲۱].

---

[۱۱] Band-width tight

۱.۲.۵ ایده‌هایی در مورد اثبات قضیه

نتایج این مقاله بیشتر براساس کارهای نویسندگان قبلی درمورد مسیریابی چندرنگی و مجزای یالی صورت گرفته است. اثبات، بیشتر در قالب مسیریابی Benes صورت گرفته است. درابتدا علایمی برای راحتی کار تعریف می‌شود که از آن جمله عبارت است از:

$u^{(i)}$ نشان دهنده همسایه راس u در بعد iام می‌باشد. هم‌چنین زیر گراف‌های Q_n با مجموعه رئوس $\{\circ v|v \in V(Q_{n-1})\}$(متناظراً $\{1v|v \in V(Q_{n-1})\}$)، $\circ$—زیرمکعب (1—زیرمکعب) نامیده شده و با علامت Q°_{n-1} (Q^{1}_{n-1}) نمایش داده می‌شوند که هر دو همریخت با Q_{n-1} هستند. هم‌چنین یک تابع تصویر به‌صورت $\pi_n(a_n, a_{n-1}, \cdots, a_1) = a_{n-1}, \cdots, a_1$ که $\pi_n : V(Q_n) \to V(Q_{n-1})$ تعریف می‌شود، مورد استفاده قرار می‌گیرد. هم‌چنین برای راحتی بیان اثبات، گراف مسیریابی جهت‌دار برای ساختن جواب مسایل مسیریابی بدون‌جهت استفاده می‌شود در حالیکه ابرمکعب Q_n را همچنان بدون جهت در نظر می‌گیریم. دقت کنید که جهت‌دار کردن صرفاً یک وسیله بیان است واگر R و R' گراف‌های مسیریابی جهت‌دار با گراف زمینه یکسان باشند، آن‌گاه یک جواب برای R دقیقا یک جواب برای R' نیز می‌باشد.

ایده اصلی اثبات، همانند ایده اثبات اکثر مسایل در ابرمکعب، ایده‌ی استقرا می‌باشد و درواقع حضور تابع تصویر در این اثبات خود گواهی برای این مطلب می‌باشد. هم‌چنین برای قویترکردن استقرا گراف مسیریابی جهت‌دار R را گراف چندتایی تعریف می‌کنیم که می‌تواند شامل طوقه و یال چندتایی هم باشد. و R را یک $h-k$ گراف مسیریابی می‌گوییم اگر بیشترین درجه خروجی آن h و بیشترین درجه ورودی آن k باشد.

اثبات ابتدا بالم زیر آغاز می‌شود:

لم ۳. فرض کنید که R یک $k-k$ گراف مسیریابی روی مجموعه رئوس V باشد، آن‌گاه R می‌تواند به k گراف مسیریابی روی V که هر کدام $1-1$ هستند تبدیل شود.

سپس اثبات با سعی در کاهش مساله مسیریابی از Q_n به Q_{n-1} و Q_{n-2} بدین صورت که

یال‌های R را تبدیل به یال‌هایی در Q°_{n-1} و Q^{1}_{n-1} کرده تا از استقرا بتوان استفاده نمود، ادامه می‌یابد و بعد از اثبات ۸ لم و نتیجه به قضیه زیر می‌رسد:

قضیه ۳۵. هر تطابق در $V(Q_n)$ می‌تواند با چهار رنگ در Q_n تحقق یابد.

اثبات این قضیه یک الگوریتم کارا را نیز برای ساختن این مسیرها می‌دهد که از $O(N \log N)$ ($N = 2^n$ و تعداد رئوس Q_n است) می‌باشد.

با توجه به این اثبات این سوال که حداقل k برای تحقق یک تطابق درQ_n چند است، بین ۲ و ۴ قرار می‌گیرد و می‌تواند پایه‌ی مقاله‌های بعدی در این زمینه شود.

۳.۵ NP-complete بودن بعضی مسایل مسیرهای مجزای یالی

مسایل مسیرهای مجزای یالی طبیعتاً از تحلیل سوال‌ها و تعمیم مسایل شبکه‌ی جریان حاصل می‌شوند. دلیل دیگر برای جلب علاقه در این زمینه کاربردهای مختلف آن نظیر طراحی VLSI است. چون این مساله در حالت کلی NP-Hard است، محققان بسیاری سعی در شناسایی کلاس‌هایی از این مساله که الگوریتم چندجمله‌ای برای آن‌ها وجود دارد، نموده‌اند. ما در این‌جا نتایج Rothschild، Whinston، Nash-williiams، Ibaraki و دیگران را بررسی می‌کنیم.

در این بخش تمام گراف‌ها اعم از جهتدار و بدون‌جهت ممکن است که یال‌های چندگانه داشته باشند. اگر G یک گراف باشد، ما با علامت $\tilde{G}$، زیر گراف ساده ماکزیمال G را نشان می‌دهیم. اگر G و H دو گراف روی یک مجموعه رئوس باشند، $G+H$ گرافی را که مجموعه یال‌های آن اجتماع $E(G)$ و $E(H)$ خواهد بود، نشان می‌دهد(یال‌های چندگانه ممکن است به‌وجود آیند).

اگر G یک گراف جهتدار باشد، ما با علامت $(\delta_G(v))\rho_G(v)$ درجه ورودی(خروجی) یک راس را نشان می‌دهیم. یک گراف بدون‌جهت(جهت‌دار) اویلری است اگر درجه تمام رئوس آن زوج باشد(برای هر راس $v \in V(G)$ داشته‌باشیم: $\delta_G(v) = \rho_G(v)$).

مانند بخش قبل یک نمونه از مساله مسیرهای مجزای یالی را با یک جفت (G,H) از گراف‌ها روی یک مجموعه از رئوس نشان می‌دهیم. یک جواب (G,H) مشابهاً خانواده $P=\{P_f|f\in E(H)\}$ از مسیرهای مجزای یالی در G است به‌طوری‌که برای هر $f\in E(H)$، P_f با f یک دور تشکیل می‌دهند .(اندکی با تعریف بخش قبل متفاوت است.)

در زیربخش ۱ این بخش ما با کلاسی از مسایل سروکار داریم که $G+H$ اویلری است. ما ابتدا به اثبات NP-complete بودن درحالت جهت‌دار می‌پردازیم. سپس یک لم زیبا ما را قادر می‌سازد که نتایج مشابهی برای حالت بدون‌جهت بیابیم. آن‌گاه در زیربخش ۲ مسایل مسیرهای مجزای یالی را هنگامی که G یک گراف جهت‌دار مسطح است ملاحظه می‌کنیم. سخت‌ترین اثبات در این بخش NP-complete بودن مساله مسیرهای مجزای یالی وقتی‌که گراف G بدون‌جهت، بدون دور و مسطح است، می‌باشد.

۱.۳.۵ مسایل اویلری

ما نخست روی حالتی که G جهتدار و بدون دور باشد تمرکز می‌یابیم. اولین نتیجه NP-complete به‌قرار زیر است:

قضیه ۳۶. [Even et al.] مساله مسیرهای مجزای یالی جهت‌دار NP-complete است، حتی اگر G بدون دور بوده و $|E(\tilde{H})|=2$.

درحقیقت اثبات کنندگان اصلی قضیه خود را بدون این محدودیت که G بدون دور است، بیان نموده‌اند، اما گرافی که آن‌ها ساخته‌اند (در تبدیل مساله ارضاشدن[۱۲]) در حقیقت بدون دور است.

یک الگوریتم چند جمله‌ای برای مساله مسیرهای مجزای یالی وقتی که $G+H$ اویلری باشد و یکی از دو شرط $|E(H)|=3$ یا $|E(\tilde{H})|=2$ برقرار باشد، وجود دارد ولی به‌هرحال ما می‌توانیم قضیه زیر را ثابت کنیم:

---

[۱۲] Satisfiability

قضیه ۳۷. (Vygen) مساله مسیرهای مجزای یالی جهت‌دار NP-complete است، حتی اگر G بدون دور باشد، $G+H$ اویلری باشد و $|E(\tilde{H})| = ۳$.

اثبات این قضیه با تبدیل مساله قضیه ۳۶ به این مساله صورت می‌گیرد.

حال ما به مساله مسیرهای مجزای یالی در حالت بدون‌جهت می‌پردازیم.

قضیه ۳۸. (Middendor and Pfeiffer) مساله مسیرهای مجزای یالی بدون‌جهت NP-complete است، حتی اگر $G+H$ مسطح باشد و یااین‌که $G+H$ اویلری باشد.

یک الگوریتم چند جمله‌ای برای مساله مسیرهای مجزای یالی بدون‌جهت اگر $G+H$ اویلری باشد و $|E(\tilde{H})| = ۲$ باشد، وجود دارد.

اماقضیه قویتر زیر را داریم:

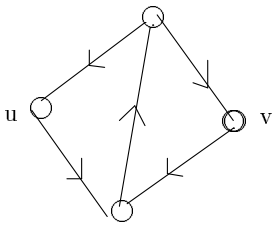
قضیه ۳۹. (Vygen) مساله مسیرهای مجزای یالی بدون‌جهت NP-complete است، حتی اگر $G+H$ اویلری و$|E(\tilde{H})| = ۲$ باشد.

اثبات.

ما قضیه ۳۷ را برای حالت بدون‌جهت به‌کار می‌بریم تا قضیه ۳۹ حاصل شود. ولی قبل از آن لم زیر را لازم داریم:

لم ۴. (Vygen) فرض کنید (G,H) یک نمونه از مساله مسیرهای یالی جهت‌دار باشد که G بدون دوراست و $G+H$ اویلری می‌باشد. نمونه $(G'+H')$ از مساله مسیرهای مجزای یالی بدون‌جهت را که با نادیده گرفتن جهت‌ها حاصل می‌شود، در نظر آورید. هر جواب (G,H) در همان زمان یک جواب (G,H) هم می‌باشد و برعکس.

اثبات. اثبات این‌که هر جواب (G,H) یک جواب (G',H') هم می‌باشد واضح است. طرف دیگر را هم با استقرا روی $|E(G)|$ می‌توان ثابت نمود. □



شکل ۱.۵: زیرگراف استفاده شده

برای کامل کردن اثبات فقط کافی است که لم رادر مورد قضیه ۳۷ به‌کار بریم. □

لازم به‌ذکر است که لم بالا در جهت عکس هم مورد استفاده قرار می‌گیرد. یعنی ما را قادر می‌سازد که نسخه جهت‌دار بعضی قضایای گراف‌های بدون‌جهت را ثابت کنیم. برای مثال قضیه Okamura-Seymour یا قضیه ۵—پایانه را می‌توانید ببینید.

۲.۳.۵ مسایل مسطح

اگر $G+H$ مسطح باشد و G بدون دور باشد، یک نسخه دوگان مسطح قضیه Lucchesi-Younger به ما یک الگوریتم چند جمله‌ای می‌دهد، اما بهرحال قضیه زیر را داریم.

قضیه ۴۰. مساله مسیرهای مجزای یالی جهت‌دار NP-complete است، حتی اگر $G+H$ مسطح باشد.

اثبات. بااستفاده از قضیه ۳۸ ما می‌توانیم مساله مسیرهای مجزای یالی بدون‌جهت را که در این‌جا $G+H$ مسطح است، به نسخه جهت‌دار تبدیل کنیم؛ کافی است در نمونه بدون‌جهت (G,H) یال‌های H را به‌صورت دل‌خواه جهت‌دار کرده و هر یال u,v از G را با زیر گراف نشان داده شده در شکل ۱.۵ جایگزین کنیم. واضح است که نمونه جهت‌دار به‌دست آمده معادل نمونه بدون‌جهت اصلی باشد. بداهتاً، مسطح بودن هم حفظ شده است. □

لازم به‌ذکر است که تنها این موضوع که هر یال بدون‌جهت را با یک جفت یال مختلف‌الجهت تعویض کنیم کافی نیست. ما این بخش را با مشکل‌ترین قضیه این بخش خاتمه می‌دهیم.

قضیه ۴۱.

مساله مسیرهای مجزای یالی جهت‌دار NP-complete است حتی اگر G مسطح و بدون دور باشد.

اثبات. درستی این قضیه از تبدیل مساله ارضاشدن [۱۳] به این مساله حاصل می‌شود. برای اثبات این قضیه از تکنیک بسیار زیبایی استفاده می‌شود و مانند بسیاری ازمسایلی که به مساله ارضا شدن تبدیل می‌شوند، برای هر حرف[۱۴] و برای هر کلمه[۱۵] ساختار ساده‌ای می‌گذارد ولی برای وجود یک حرف در یک کلمه از ساختار نسبتاً پیچیده‌ای استفاده می‌کند که از ذکر آن خودداری می‌کنیم. به‌هرحال برای دیدن شیوه دقیق اثبات می‌توانید به [۲۲] مراجعه کنید. □

قضیه آخر می‌تواند به حالتی که G یک مش جهت‌دار یکنواخت باشد، نیز تعمیم یابد. این‌که مساله هنگامی که $G + H$ اویلری هم باشد آیا NP-complete است یا خیر، یک مساله باز است. هرچند، که مثال‌هایی وجود دارند که یک راه‌حل نیمه‌تمام و نه کامل را می‌دهند.

۴.۵ داده‌پراکنی جزیی در چندپردازنده‌های ابرمکعب

تاکنون هرچه سخن راندیم، بیشتر حرفمان در مورد جنبه‌های نظری شبه‌تطابق بود و کمتر به منشا عملی آن که این بحث ازآن شکل گرفته است، پرداختیم. در این بخش هدف آن است که دین مطلب را به‌جا آوریم و به جنبه‌های عملی که می‌توانند منشا تولید مسایل جدید نظری در این زمینه باشند روی می‌آوریم. برای این‌کار باز به ابرمکعب برمی‌گردیم تااین‌بار مساله داده‌پراکنی جزیی را روی آن بیشتر مطالعه کنیم.

ازذکر تعریف‌های تکراری خودداری می‌کنیم وصرفاً تعاریف جدیدی که در این بخش مورد استفاده قرار می‌گیرند را می‌آوریم.در یک سیستم انتقال پیغام که از توپولوژی ابرمکعب استفاده می‌کند، هرپردازنده در یکی از گره‌های ابرمکعب قرار می‌گیرد و انتقال پیغام تنها وسیله ارتباطات بین پردازنده‌ها

[۱۳] Satisfiability
[۱۴] Literal
[۱۵] Clause

می‌باشد. هم‌چنین فرض می‌شود که پردازنده‌ها حافظه مشترک ندارند. در این‌جا زمان و ترافیک دو پارامتر اصلی هستند که بعداً به تعریف آن‌هامی‌پردازیم.

ماسه نوع داده‌پراکنی داریم: داده‌پراکنی واحد(یک به یک)، داده‌پراکنی جزیی (یک به چند) و داده‌پراکنی‌کلی(یک به همه) که همانطور که قبلاً گفتیم، داده‌پراکنی جزیی مفهوم هر سه نوع داده‌پراکنی را می‌پوشاند، لذا تنها به‌ذکر آن می‌پردازیم. مادر زیربخش ۱ تعاریف مورد نیاز این بخش را ارایه می‌کنیم، آن‌گاه زیربخش ۲ ما را به این حدس که داده‌پراکنی کلی حتی در مکعب‌های n بعدی هم NP-Hard است هدایت می‌کند و سپس در زیربخش آخر یک الگوریتم مکاشفه‌ای را که شبیه‌سازی، کارایی بالای آن را تایید می‌کند، ارایه می‌دهیم و این بخش را به اتمام می‌رسانیم.

۱.۴.۵ مدل درخت داده‌پراکنی جزیی بهینه

[۱۶] ما مکعب n—بعدی را با $Q_n(V,E)$ نمایش می‌دهیم که $a(v)$، آدرس n—بیتی راس v را نشان می‌دهد. هم‌چنین تعداد یک‌ها در عدد باینری $a(v)$ را با $||a(v)||$ نشان می‌دهیم. هم‌چنین علامت $\oplus$ رابرای عملگرد دودویی یای انحصاری (XOR) به‌کار می‌بریم. حال $e=(u,v)\in E(Q_n)$ اگر وتنهااگر $||a(u)\oplus a(v)||=۱$ باشد.به‌علاوه اگر $a(u)+a(v)=۲^i$ باشد، گفته می‌شود یال eدر بعدiام است. هم‌چنین رابطه‌ی $d_{Q_n}(u,v)=||a(u)\oplus a(v)||$ برای هر جفت گره $v,u\in V(Q_n)$ صادق است.

دراین‌جا دو پارامتر اصلی زمان و ترافیک را تعریف می‌کنیم. پارامتر زمان با مرحله زمانی اندازه‌گیری می‌شود که یک مرحله زمانی یک زمان لازم برای ارسال یک واحد اطلاعات از یک گره به همسایه‌اش می‌باشد. این زمان را برای تمام جفت همسایه‌ها، ثابت درنظر می‌گیریم. پارامتر ترافیک با تعداد پیغام‌هایی که از یال‌های ارتباطاتی به‌منظور ارسال پیغام از یک مبدا به رئوس مقصد استفاده می‌کنند، اندازه‌گیری می‌شود. توجه کنید که تعداد مراحل زمانی برای مطلع‌کردن یک مقصد حداقل به‌اندازه فاصله مبدا تامقصد می‌باشد. هم‌چنین، تعداد مقاصد یک حد پایینی برای مقدار ترافیک لازم برای کامل‌کردن ارتباطات می‌دهد. واضح است که طرح مسیریابی که هر دوی این پارامترها را کمینه کند، بهینه است ولی همانطور که بعداً خواهیم دید، این دو پارامتر کاملا، از هم مستقل نیستند.

[۱۶] Optimal Multicast Tree

برای مثال، یک داده‌پراکنی جزیی در ۴–مکعب را در نظر بگیرید که درآن گره ۰۰۰۰ مبدا وگره‌های ۱۰۱۱، ۱۰۱۰ و ۱۰۰۱ مقصد می‌باشند. واضح است که این ارتباطات می‌توانند با حداقل ۴ واحد ترافیک صورت گیرند (بااستفاده از مسیر ۱۰۰۱ → ۱۰۱۱ → ۱۰۱۰ → ۰۰۱۰ → ۰۰۰۰)، در این‌جا گره ۱۰۰۱ پیغام را بعد از ۴ مرحله می‌گیرد، در حالی که می‌تواند آن‌را بعد از ۲ مرحله (اگر مسیر ۱۰۰۱ → ۱۰۰۰ → ۰۰۰۰ مورد استفاده قرارگیرد)، دریافت کند. به‌راحتی می‌توان نشان داد که هیچ روش مسیریابی وجود ندارد که با ۴ واحد ترافیک، در کمترین تعداد مراحل این عمل را انجام دهد(اگر به‌جای ۴ واحد از ۵ واحد استفاده کنیم این‌کار از طریق مسیرهای ۱۰۱۱ → ۱۰۰۱ → ۱۰۰۰ → ۰۰۰۰ و ۱۰۱۰ → ۰۰۱۰ → ۰۰۰۰ امکان پذیراست). در این‌جا یک مدل گراف برای ارتباطات داده‌پراکنی جزیی در یک سیستم توزیع‌شده چند پردازنده‌ای می‌تواند تعریف شود. فرض کنید که u_0 راس مبدا و $D = \{u_1, u_2, \cdots, u_k\} \subseteq V$ رئوس مقصد باشند. مجموعه $M = u_0 \cup D$ مجموعه داده‌پراکنی جزیی نامیده می‌شود. مساله داده پراکنی جزیی، مساله پیداکردن یک زیر درخت T از G است که درخت داده‌پراکنی جزیی بهینه (OMT) نامیده می‌شود، به‌طوری‌که:

۱) $M \subseteq V(T)$

۲) شرط $d_T(u_0, u_i) = d_G(u_0, u_i)$ برای هر $1 \leq i \leq k$ برقرار باشد.

۳) $|E(T)|$ کمترین مقدار را داشته باشد.

دقت کنید که شرط ۲ کمترین تعداد مراحل زمانی را تضمین می‌کند و شرط ۳ برای کمترکردن ترافیک به‌کار برده می‌شود. یک زیردرخت G که فقط شرط‌های ۱ و ۲ را برآورده کند، درخت داده‌پراکنی (MT) نامیده می‌شود. یک برگ[۱۷] یک راس مقصد با درجه یک است و بقیه گره‌های غیر برگ، گره‌های جلوبر[۱۸] نامیده می‌شوند. در حالت کلی OMT ممکن است یکتا نباشد. در بخش بعدی

[۱۷] Leaf

[۱۸] Forward nodes

مسایل OMT را وقتی که G یک n—مکعب است، بررسی می‌کنیم و یک الگوریتم مکاشفه‌ای برای به‌دست آوردن آن ارایه می‌دهیم.

۲.۴.۵ پیداکردن OMT در یک ابرمکعب

در این‌جا بدون کاسته‌شدن از کلیت مساله فرض می‌کنیم که راس مبدا آدرسش صفر است و هم‌چنین $\lambda(u)$ را برابر $d_{Q_n}(u_0, u)$ تعریف می‌کنیم. هم‌چنین تعریف زیر را داریم:

تعریف ۱۲. یک راس u جد راس v نامیده می‌شود (یا v نوه u است) اگر وتنها اگر u در کوتاهترین مسیر بین u و v باشد. بنابراین $\lambda(u) \leq \lambda(v)$ و $\lambda(v) - \lambda(u) = d_{Q_n}(u, v)$. دقت کنید که u و v لزومی ندارد که مجزا باشند.

حال قضیه زیر را داریم:

قضیه ۴۲. اگر $X \subseteq V - \{u_0\}$ و $|X| = k (2 \leq k \leq 2^n - 1)$ باشد، آن‌گاه $NCA(x)$ فقط یک عضو دارد؛ یعنی نزدیکترین جد مشترک هر $k (2 \leq k \leq 2^n - 1)$ راس، یکتا می‌باشد. بعلاوه، آدرس این گره می‌تواند با AND بیتی آدرس‌های باینری تمام k گره به‌دست آید.

در این‌جا $NCA(X) = \{u \in V | u \in CA(X) \text{ and for every } v \in CA(X) : \lambda(u) \geq \lambda(v)\}$ و $CA(X) = \{u \in V | \text{ u is an ancestor of every node } v \in X\}$ است.

بااستفاده از قضیه بالا می‌توان OMT رابرای حالات $|M| = 3$ و $|M| = 4$ با حالت‌گیری به‌دست آورد[۲۰]. در حالت کلی نشان داده شده است که پیداکردن یک OMT برای هر k، NP-Hard است، حتی اگر گراف دوبخشی باشد. یک مساله مشابه با این مساله، مساله درخت اشتاینر(ST) [۱۹] می‌باشد، که آنجا پیداکردن کوچکترین زیر درخت از یک گراف مطرح است که هر راس از یک مجموعه داده شده را در برداشته باشد. می‌توان مشاهده کرد که اگر شرط ۲ در تعریف

[۱۹] Steiner Tree

OMT حذف شود، مساله OMT، همان مساله ST خواهد شد. نشان داده‌شده است که مساله ST، NP-completeاست، حتی اگر گراف n—مکعب باشد.

۳.۴.۵ الگوریتم مکاشفه‌ای برای داده‌پراکنی جزیی

درا ینجا از ذکر کامل الگوریتم خودداری می‌کنیم وصرفاً به‌ذکر ایده آن می‌پردازیم. لازم به‌ذکراست که همانطور که گفته شده شبیه‌سازی این الگوریتم صورت گرفته است ونشان داده است که در عمل بسیار کارا می‌باشد. در این‌جا فرض می‌کنیم که هر راس ابرمکعب به‌صورت مستقل و صرفاً با استفاده از اطلاعات ورودی، تصمیم می‌گیرد که پیغام را برای کدام یک از همسایه‌هایش بفرستد. بدین ترتیب که هر راس آدرس تمامی رئوسی را که این پیغام از این راس باید به آن‌ها برود را از راس قبل می‌گیرد. سپس بیتی را که در بیشترین این مقاصد برابر یک است می‌یابد و پیغام را برای تمام مقاصدی که در این بیت یک هستند در بعدی که این بیت قرار دارد می‌فرستد(به همسایه‌اش در این‌بعد) و این‌کار را تا فرستادن پیغام برای تمامی مقاصد تکرار می‌کند.

بعد از اثبات چند لم و قضیه می‌توان ثابت کرد که این الگوریتم درست کار می‌کند[۲۰]. و الگوریتم پیاده‌سازی شده از $O(nk + n^2)$ است که n، بعد ابرمکعب و k، تعداد مقاصد در داده‌پراکنی جزیی می‌باشد.

درپایان این فصل چند بخش که اکثراً خلاصه بسیار کوتاهی از چند مقاله مربوط به این موضوع می‌باشند می‌آوریم.

۵.۵ بیشترین تعداد مسیرهای مجزا متصل کننده‌ی تعداد مشخصی پایانه

در یک گراف ساده بدون‌جهت $G = (V, E)$ یک مجموعه‌ی مشخص از رئوس، T، پایانه‌ها نامیده می‌شوند. یک مسیر بین یک جفت پایانه که از میان هیچ پایانه دیگری نمی‌گذرد، یک T—مسیر نامیده می‌شود. یک اجتماعی از T—مسیرهای مجزای‌راسی یک T—سیستم نامیده می‌شوند. در این مقاله

بیشتر به‌دنبال پیداکردن یک T—سیستم با بیشترین تعداد (MDT-P) برای یک مجموعه مشخص پایانه‌های T در گراف G می‌باشیم.

اگر $T = V$ باشد، آن‌گاه مساله MDT-P همان مساله تطابق بیشینه می‌شود. از طرف دیگر اگر T فقط شامل دوعضو (t, s) باشد می‌توان از قضیه Menger روی مسیرهای s به t مجزای بیشینه استفاده نمود. هم‌چنین به رابطه‌ی بین MDT-P و تعمیم Gallai(از نتایج Menger از دو پایانه به چند پایانه) برای پیداکردن یک خانواده‌ی بیشینه از مسیرهای دوبدو مجزای‌داخلی در این مقاله اشاره شده است. (مسیرها فقط ممکن است که در پایانه‌ها و نه در هیچ جای دیگر با هم تلاقی داشته باشند).

هم‌چنین در این مقاله یک دوگان برای مساله MDT-P ارایه شده و یک قضیه min-max حاصل شده است. و به‌مشابه تعمیم تطابق، گذرهای تناوبی و افزایشی مطرح و بحث شده‌اند. هم‌چنین نشان داده شده است که یک T—سیستم بیشینه است اگر وتنها اگر هیچ گذر افزایشی نداشته باشد و این ما را به سمت یک الگوریتم چندجمله‌ای برای T—سیستم با بیشترین تعداد مسیرها راهنمایی می‌کند که هدف اصلی این مقاله است. هم‌چنین پیچیدگی الگوریتم ارایه شده بررسی شده و نشان داده شده است که برای یک گراف n راسی، $O(n^4)$ زمان لازم است.

٦.۵ مسایل تطابق‌مسیری

در این مقاله نیز مستقیماً بحث شبه‌تطابق(تطابق‌مسیری) مورد بحث و بررسی قرار گرفته است. در این مقاله با گراف‌هایی با تعداد رئوس زوج و وزن‌های مثبت روی یال‌ها سروکار داریم. هم‌چنین روی مساله پیداکردن یک شبه‌تطابق کامل به‌طوری‌که بیشترین وزن یال‌ها، کمینه باشد، بحث شده است ونشان داده شده است که یک الگوریتم چندجمله‌ای برای درخت‌ها وجود دارد ولی برای گراف‌های دل‌خواه مساله NP-complete است.

۷.۵ زمانبندی تماس‌ها برای داده‌پراکنی جزیی در شبکه‌های درخت

در این مقاله، نشان داده شده است که مساله داده‌پراکنی جزیی در درخت‌ها می‌تواند براساس مرتب‌کردن سطرها و ستون‌ها یک ماتریس ۰ و ۱ بیان شود. یک ماتریسی M، $p*q$ با درایه‌های ۰ و ۱ داده شده است. سایه M، بردار۰ و ۱، x از q درایه است به‌طوری‌که $x_i = ۰$ اگر وفقط اگر هیچ درایه‌ی ۱ در ستون iام ماتریس وجود نداشته باشد. با توجه به این فرمول‌بندی، نشان داده شده است که کلید حل مساله داده‌پراکنی در درخت‌ها موضوع زیر است: یک ماتریس M،$p*q$ با درایه‌های ۰ و ۱ داده شده است. ماتریس M^* را پیدا کنید به‌طوری‌که :

۱) M^* حداکثر یک درایه یک در هرستون داشته باشد.

۲) هر سطر r از M^*(که به‌صورت بسط باینری یک عدد دیده‌شده است) از سطر متناظر r در M بزرگ‌تر باشد، برای هر $۱ \leq r \leq p$.

۳) سایه M^*(که به‌صورت عدد صحیح نگریسته شده است) کمینه باشد.

دراین مقاله نشان داده شده است که یک الگوریتم $O(q(p+q))$ برای پیداکردن M^* هر ماتریسی صفر و یک M وجود دارد. کاربرد این نتیجه به‌قرار زیراست:

یک درخت جهت‌دار T که یال‌هایش از ریشه به‌سمت برگ‌ها جهت‌دار شده‌اند و یک مجموعه D از گره‌ها داده شده‌اند. یک الگوریتم چند جمله‌ای وجود دارد که پروتکل داده‌پراکنی جزیی از ریشه به تمام برگ‌ها را فراهم می‌آورد. پروتکل حاصل می‌تواند بر روی مدل‌های ۱–پورتی یا تمام–پورتی سوار شود و امکان تماس‌های با فاصله زیادِ مجزای یالی را فراهم آورد. هم‌چنین در مورد مدیریت کاربردهای چندنقطه‌ای[۲۰] (نظیر پایگاه‌داده [۲۱] و خادمِ ویدیو [۲۲] و ۰۰۰) که به‌شکل شبکه درخت شکل‌ریزی شده‌اند، کاربرد فراوان دارد.

---

[۲۰] Multi-Point applications

[۲۱] Data-Base

[۲۲] Video Server

فصل ٦

الگوریتم ۱ –تقریبی برای مساله‌ی شبه‌تطابق تنگنایی

۱.٦ مقدمه

در فصول قبلی شبه‌تطابق و تعمیم‌هایی از آن را معرفی و بررسی نمودیم. دراین فصل هدف آن است که تعمیم دیگری از این موضوع را بررسی نموده و یک الگوریتم ۱ –تقریبی برای حل آن ارایه دهیم. در این فصل منظور ما از شبه‌تطابق، شبه‌تطابق مجزای یالی است و فقط برای اختصار از نام شبه‌تطابق استفاده می‌نماییم.

همان‌گونه که در فصول قبل گفته شد، تعمیم‌های زیادی روی این‌گونه تطابق وجود دارند که اکثر آن‌ها براساس خواسته‌ها و محدودیت‌هایی است که روی این مساله گذاشته می‌شوند. در این فصل نیز به معرفی دو نوع دیگر ازآن‌ها می‌پردازیم. اولین مساله پیدا کردن یک شبه‌تطابق با مینیمم مجموع وزن مسیرها است که در این‌جا وزن یک مسیر را مجموع وزن یال‌های موجود در آن تعریف می‌کنیم(نسخه‌ی مجموعی). نوع دیگر، پیداکردن شبه‌تطابق با مینیمم ماکزیمم وزن مسیرهااست(نسخه‌ی تنگنایی). الگوریتم‌های کارایی برای حل مساله تطابق ماکزیمم وجود دارد که در فصول قبلی آن‌ها را ذکر کردیم. مساله‌ی شبه‌تطابق با مینیمم مجموع نیز می‌تواند با تبدیل به مساله‌ی معمولی ماکزیمم تطابق به‌صورت کارایی حل شود[۲۷] برای نسخه‌ی تنگنایی شبه‌تطابق،

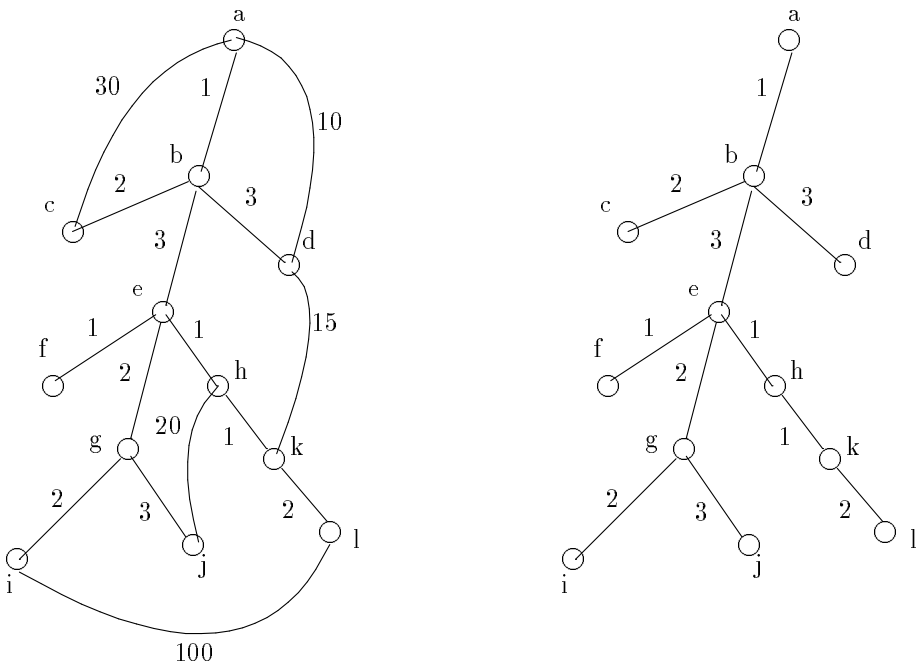
Wu و Manber [٢٧] الگوریتم کارایی را در حالتی که گراف درخت باشد ارایه کردند. هم‌چنین آن‌ها ثابت کردند که این مساله برای گراف‌های کلی NP-complete است[٢٧]. تاکنون این موضوع که آیا این مساله راه‌حل تقریبی دارد یا خیر نامعلوم مانده بود. دراین فصل ما یک الگوریتم ١ –تقریبی ساده با زمان چند جمله‌ای برای مساله‌ی شبه‌تطابق تنگنایی (BDPM) ارایه می‌دهیم. لازم به‌ذکر است که مرجع اصلی این الگوریتم[١۴] است. این الگوریتم در زمان $O(m \log \beta(m,n))$ اجرا می‌شود که $\beta(m,n) = min\{i | \log^{(i)} n \leq m/n\}$. در این‌جا n و m به‌ترتیب تعداد رئوس و یال‌های گراف می‌باشند.

در پایان این فصل به پیاده‌سازی این الگوریتم اشاره‌ای نموده و کد برنامه را در B آورده‌ایم.

٢.٦ بعضی نتایج مقدماتی

در تمام طول این فصل گراف را بدون‌جهت فرض می‌کنیم و هدف را پیداکردن شبه‌تطابق کامل مجزای یالی که به اختصار پیداکردن BDPM می‌گوییم، قرار می‌دهیم. فرض کنید که گراف $G = (V, E)$ دارای تعداد زوجی راس می‌باشد. هم‌چنین $w(u,v)$ را وزن یال (u,v) درنظر بگیرید. در حالتی که تعداد رئوس G فرد باشد، می‌توانیم یک راس به گراف اضافه کرده و این راس را با یال‌های با وزن خیلی زیاد به تمام رئوس دیگر متصل کنیم. حال BDPM گراف G می‌تواند به‌راحتی از BDPM گراف جدید حاصل شود.

فرض کنید که T درخت پوشای کمینه گراف G باشد. این درخت را از ریشه‌ی یک راس دل‌خواه مانند r جهت‌دار می‌کنیم: در T یک راس u پدر v نامیده می‌شود و با $parent(v)$ نشان داده می‌شود اگر u بر روی مسیر بین r و v بوده و u همسایه v در T باشد. یک زیر درخت T، زوج(فرد) نامیده می‌شود، اگر دارای تعداد زوج(فرد)ی راس باشد. یک زیردرختِ درخت T به ریشه v را با $T(v)$ نشان می‌دهیم. فرض کنید: $E_T = \{(u,v) | (u,v) \in T, u = parent(v), T(v) is even\}$.در این‌جا $T - E_T$ یک جنگل پوشای جهت‌دار از G را تعریف می‌کند. فرض کنید $F = \bigcup_i T_i, i = ١, ٢, \cdots, k$ باشد که در این‌جا T_i یک درخت در جنگل پوشای F می‌باشد.(شکل ١.٦ را ببینید).



شکل ١.٦: یک گراف وزندار G (A) و درخت پوشای کمینه آن (B).

مایک نوع خاص BDPM از روی جنگل F می‌سازیم و ثابت می‌کنیم که آن یک راه‌حل ۱–تقریبی برای مساله‌ی شبه‌تطابق تنگنایی کامل است.

لم ۵. هرگراف یک BDPM کامل دارد[۲۷].

لم ٦. فرض کنید $F = \bigcup_i T_i, i = ۱, ۲, \cdots, k$ باشد که r_i ریشه‌ی درخت T_i می‌باشد. اگر $T_i(v)$ زیر درخت T_i با ریشه‌ی v و $v \neq r$ باشد، آن‌گاه $T_i(v)$ فرد است.

اثبات. زیردرخت T_i به ریشه v_i با توجه به ساختار F زوج است. فرض کنید $u = parent(v)$ و $u \in T_i$باشد. بنابراین (u, v) یک یال درخت T می‌باشد. زیردرخت درخت پوشای T به ریشه‌ی v زوج(فرد) است، اگر زیردرخت T_i در F به ریشه v زوج(فرد) باشد. بنابراین یال (u, v) در E_T قراردارد. لذا، چون یال‌های E_T از درخت T برای به‌دست آوردن جنگل پوشای F برداشته شده‌اند، این موضوع ما را به تناقض می‌رساند. □

شبه‌تطابق کامل تنگنایی گراف را با P و بیشترین وزن مسیرها در P را با $b(P)$ نشان می‌دهیم و آن‌را وزن تنگنایی P می‌نامیم.

لم ۷. اگر $P = \bigcup_i P_i$ یک BDPM با وزن تنگنایی $b(P)$ و w^* یال با بیشترین وزن در F باشد، آن‌گاه $b(P) \geq w^*$.

اثبات. فرض کنید (u, v) یالی در F باشد که $w(u, v) = w^*$ باشد. هم‌چنین فرض کنید که $u = parent(v)$ در T_i باشد. براساس لم ۲، $T_i(v)$ فرد است. فرض کنید که $G_i(v)$ زیرگراف القاشده G بر روی رئوس $T_i(v)$ باشد. مشاهده می‌شود که G و $G_i(v)$ به‌ترتیب دارای تعداد فرد و زوجی راس هستند. این بدان معناست که حداقل یک مسیر $P_i = u_i \cdots v_i$ در P وجود دارد، بطوری‌که

$u_i \in G_i(v)$ و $v_i \in G - G_i(v)$ باشد. بنابراین $b(P) \geq w'$ که در این‌جا w' وزن یال با کمترین وزن (u', v')درمجموعه‌ی برشی G که با یال‌های بین رئوس G_i و $G - G_i(v)$ مشخص می‌شوند، باشد.

برای تکمیل اثبات نشان خواهیم داد که $w^* = w'$. فرض کنید که $w' = w(u', v') < w^* =$ $w(u, v)$ باشد. مشاهده می‌شود که T درخت پوشای کمینه G و (u, v) یک یال از T می‌باشد. اگر ما یال (u, v) را از T برداشته و یال (u', v') را به آن اضافه کنیم آن‌گاه مایک درخت پوشای دیگری بنام T' به‌دست می‌آوریم. وزن T' از وزن درخت پوشای کمینه T کمتراست که یک تناقص می‌باشد. لذا $b(P) \geq w^*$. □

دراین‌جا ما یک شبه‌تطابق خاص از روی F می‌سازیم و نشان می‌دهیم که این شبه‌تطابق می‌تواند به‌عنوان یک راه‌حل ١ –تقریبی برای مساله BDPM مورد استفاده قرار گیرد. برای جنگل پوشای ریشه‌دار F از G، که درآن هر زیر درخت باریشه غیر از r_i ها فرد است، ما شبه‌تطابق برادری (SDPM) را شبه‌تطابقی کامل که شرایط زیر را داشته باشد، تعریف می‌کنیم:

١) ریشه‌های $r_i, i = ١, ٢, \cdots, k$ جنگل پوشا باید با یکی از بچه‌های خود از طریق مسیر به‌طول یک جفت شوند.

٢) هرکدام از رئوس جفت نشده باقی‌مانده در هر T_i باید به یکی از برادرانش از طریق مسیر به‌طول دو جفت شود.

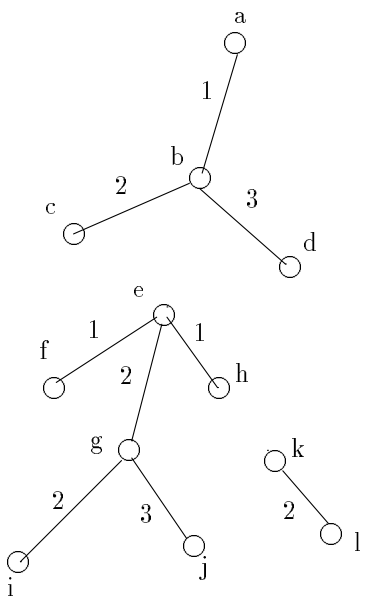
برای مثال برای جنگل پوشای ریشه‌دار شکل ٢.٦ ، SDPM به‌صورت زیر داده‌شده است:

$$\{(a, b), (c, b, d), (e, g), (f, e, h), (i, g, j), (k, l)\}$$

لم ٨.

یک SDPM، P' از F وجود دارد که بیشترین وزن مسیرها در P'، $٢w^*$ است.

اثبات. ما r_i را با یکی ازبچه‌هایش جفت می‌کنیم. برای تمام $i = ١, ٢, \cdots, k$، تمام رئوس باقی‌مانده



شکل ٢.٦: جنگل پوشای ریشه‌دار F که از T کل باحذف یال‌های E_T حاصل شده است.

درهر T_i می‌تواند با یکی از برادرانش جفت شود، تا یک SDPM از F بسازند. این امکان‌پذیر است، چون یک تعداد زوجی بچه برای هر راس داخلی وجود دارد. این یک شبه‌تطابق کامل P'، از F را می‌دهد. هر مسیرتطابقی در P' حداکثر دو یال دارد، هم‌چنین بهترین وزن یک یال در F برابر w^* است. لذا $b(P') = ٢w^*$. □

حال مشاهده می‌شود که $\frac{b(P')-b(P)}{b(P)} = \frac{b(P')}{b(P)} - ١ \leq \frac{٢w^*}{w^*} - ١ = ١$ (*).

چون هر شبه‌تطابق کامل F یک شبه‌تطابق کامل G نیز می‌باشد، داریم:

قضیه ۴۳. P یک جواب ١–تقریبی برای مساله پیداکردن شبه‌تطابق کامل تنگنایی در G است.

الگوریتم ٣ (الگوریتم پیداکردن یک BDPM)

- **ورودی**: یک گراف وزندار بدون جهت $G = (V, E)$.
- **خروجی**: یک جواب ۱–تقریبی برای مساله $BDPM$ در G.
- **مراحل**:

۱) زیردرخت پوشای کمینه‌ی T در G را پیداکن.

۲) یک راس را به‌عنوان ریشه انتخاب کن و تمام رئوس v را براساس این‌که زیردرخت به ریشه v در T دارای تعداد زوج یا فردی راس است، زوج و یا فرد بنام.

۳) برای هرراس v که در مرحله دوم زوج نام گرفته است، پدرش را بیاب و یال $(v, parent(v))$ را ازدرخت T حذف کن، تا جنگل پوشای F حاصل شود.

۴) یک P' $SDPM$ از F بیاب. P' یک جواب مساله با شرایط خواسته شده است.

لم ۹. الگوریتم $BDPM$ به‌درستی یک جواب ۱–تقریبی برای مساله $BDPM$ در زمان $O(m \log \beta(m,n))$ می‌یابد که در این‌جا $\beta(m,n) = min\{i | \log^{(i)} n \leq m/n\}$.

اثبات. درستی الگوریتم از رابطه‌ی (*) اثبات می‌شود. برای محاسبه‌ی پیچیدگی زمانی، می‌بینیم که مرحله‌ی اول الگوریتم درزمان $O(m \log \beta(m,n))$ با استفاده از الگوریتم Gabow برای محاسبه درخت پوشای کمینه به‌دست می‌آید[۱۴]. مراحل ۲ و ۳ نیز می‌توانند در زمان خطی انجام شوند. محاسبه SDPM در F نیز در مرحله ۴ می‌تواند در زمان خطی به‌صورتی روشن پیاده‌سازی شود. □

دراین‌جا لازم به‌ذکراست که الگوریتم ارایه شده می‌تواند به‌راحتی موازی‌سازی شده تا تبدیل به الگوریتم NC برای حل مساله BDPM شود. مرحله ۱ می‌تواند براحتی توسط الگوریتم موازی محاسبه‌ی درخت پوشای کمینه در $O(\log n)$ پیاده‌سازی شود[۷]. مراحل ۲، ۳ و ۴ هم براحتی در $O(\log n)$ قابل پیاده‌سازی می‌باشند. لذا در کل الگوریتم NC می‌باشد.

۳.٦ پیاده‌سازی الگوریتم

همان‌گونه که در بالا ذکرشد پیاده‌سازی مراحل ۲ تا۴ الگوریتم بالا به‌راحتی در زمان خطی قابل اجرا می‌باشد. تنها در مورد مرحله اول الگوریتم بود که اشاره شد باید از الگوریتم Gabow استفاده می‌شود.

برای پیاده‌سازی این بخش از الگوریتم تقریبی ما بجای الگوریتم Gabow از الگوریتم بهینه شده Kruskal استفاده می‌کنیم که درخت پوشای کمینه را در $O(m \log n)$ می‌یابد. بقیه برنامه براساس الگوریتم بالا پیاده‌سازی می‌شود. کد این برنامه در B آمده است.

اگرچه الگوریتم بهینه شده Kruskal از الگوریتم Gabow اندکی کندتر است، ولی در عوض نشان داده شده است که نه‌تنها از لحاظ نظری بلکه در عمل نیز می‌توان این الگوریتم را پیداسازی نمود. برخلاف بعضی الگوریتم‌های دیگر که نیاز به ساختمان‌های داده‌ی پیچیده دارند، این الگوریتم از ساختمان‌داده‌ی نسبتاً ساده و پرکاربرد استفاده می‌کند. در توضیح این الگوریتم باید گفت که اساساً این الگوریتم شبیه الگوریتم Kruskal می‌باشد؛ بدین ترتیب که در ابتدا یال‌های گراف را براساس وزن آن‌ها در $O(m \log n)$ از کوچک به بزرگ مرتب می‌کنیم. سپس از ابتدای لیست شروع کرده و هر یال را به‌ترتیب یک بار مورد آزمایش قرار می‌دهیم، بدین ترتیب که اگر دو سر آن یال در دو مجموعه‌ی همبند یالی متفاوت بودند، آن‌را به درخت اضافه می‌کنیم و دو مجموعه همبند را تبدیل به یک مجموعه می‌کنیم. در غیر این‌صورت بدون انجام عمل خاصی از آن یال می‌گذریم.

حال همان‌طور که دیده می‌شود ما نیاز به ساختمان‌داده‌ای داریم که دو عمل تشخیص مجموعه‌ای که یک عضو در آن آمده است و هم‌چنین عمل اجتماع دو مجموعه را، در زمان خوبی انجام دهد. اگر بخواهیم این موضوع را به‌صورت دقیق‌تر شرح دهیم باید مفهوم مجموعه‌ی مجزا را تعریف کنیم. یک مجموعه‌ی مجزا یک مجموعه $S = \{S_1, S_2, \cdots, S_k\}$ از یک سری مجموعه‌ی پویای مجزا است. هر مجموعه توسط یک نماینده مشخص می‌شود که می‌تواند یکی از اعضای همان مجموعه باشد. در بیشتر کاربردها این نکته که کدام عضو مجموعه به‌عنوان نماینده انتخاب می‌شود، اهمیتی ندارد. تنها این نکته که وقتی ما دوبار در مورد نماینده‌ی یک مجموعه پویا سوال می‌کنیم بدون این‌که در این بین، این مجموعه توسط بعضی درخواست‌ها تغییر کرده باشد، یک جواب بگیریم، اهمیت دارد. ولی در بعضی کاربردهای خاص، قوانین مشخصی برای تعیین نماینده وجود دارد که در این‌جا از

ذکر آن‌ها خودداری می‌کنیم.

اگر x, y را معرف دو شی از مجموعه‌ی جهانی مورد بحث درنظر بگیریم، یک ساختمان داده مجموعه‌ی مجزا باید قادر به انجام اعمال زیر باشد:

- $MAKE-SET(x)$ که یک مجموعه جدید که تنها عضو و هم‌چنین نماینده آن x است، می‌سازد. چون مجموعه‌ها، مجزا می‌باشند ما توقع داریم که x قبلاً در هیچ مجموعه‌ای نبوده باشد.
- $UNION(x, y)$ که اجتماع دو مجموعه‌ی پویای شامل x, y بنام‌های S_x, S_y را به‌جای این دومجموعه به‌وجود می‌آورد. این دو مجموعه باید برطبق فرض مجزا باشند. نماینده‌ی این مجموعه‌ی جدید یک عضو $S_x \cup S_y$ می‌باشد که برطبق پیاده‌سازی عمل $UNION$ اغلب نماینده‌ی یکی از این دو مجموعه به‌عنوان نماینده کل انتخاب می‌شود و چون که باید تمام مجموعه‌ها مجزا باشند دومجموعه S_y و S_x از محدوده‌ی مجموعه‌های مورد بحث برداشته می‌شوند.
- $FIND-SET(x)$ که یک اشاره‌گر به نماینده‌ی مجموعه‌ی یکتای حاوی x برمی‌گرداند.

ساختمان‌داده‌ی مجموعه‌ی مجزا کاربردهای زیادی دارد که از جمله می‌توان از استفاده آن، برای پیاده‌سازی الگوریتم پیدا کردن مولفه‌های همبندی گراف نام برد[۵].

برای پیاده‌سازی یک مجموعه‌ی مجزا راه‌حل‌های مختلفی وجود دارد. یک روش آن، استفاده از لیست پیوندی است بطوری‌که، هر مجموعه را به‌صورت یک لیست پیوندی نگه داریم، ولی در بهترین حالت پیاده‌سازی توسط این ساختار، ما باید زمان $O(m + n \log n)$ برای m عمل $MAKE-SET$، $UNION$ و $FIND-SET$ که nتای آن‌ها عمل $MAKE-SET$ است ، صرف کنیم[۵].

درپیاده‌سازی سریع‌تر مجموعه‌های مجزا ما هرمجموعه را با یک درخت ریشه‌دار نشان می‌دهیم که در آن هر گره نشانگر یک عضو و هر درخت نشانگر یک مجموعه است. در یک جنگل مجموعه‌ی مجزا هر عضو فقط به پدرش اشاره می‌کند. و ریشه‌ی هر درخت نماینده‌ی آن مجموعه و پدر خودش

می‌باشد. در پیاده‌سازی اعمال مورد نیاز، از دو ایده‌ی اجتماع براساس رتبه و فشرده‌سازی مسیر استفاده می‌کنیم. خواهیم دید که فقط استفاده از این دو ایده و هم‌چنین یک پیاده‌سازی ساده ما را به یک پیچیدگی زمانی ایده‌آل می‌رساند.

ایده‌ی اجتماع براساس رتبه به این ترتیب است که ما ریشه‌ی درخت با تعداد گره‌های کمتر را بچه درخت با تعداد گره‌های بیشتر می‌کنیم.

ایده فشرده‌سازی مسیر نیز ایده‌ای بسیار کارا و ساده است. از این ایده درانجام عمل $FIND-SET$ استفاده می‌کنیم: به‌دین ترتیب که بعد از یافتن مسیر از یک گره به ریشه، تمام گره‌های این مسیر را مستقیماً به ریشه اشاره می‌دهیم.

بااستفاده از همین دو ایده‌ی بسیار ساده می‌توان نشان داد که زمان لازم برای m عمل روی مجموعه‌ی مجزا که از میان آن‌ها n تایش عمل $MAKE-SET$ است، درزمان $O(m\alpha(m,n))$ صورت می‌گیرد که در این‌جا $\alpha(m,n)$ یک تابع بسیار کم‌رشد و عکس تابع Ackerman است، به‌طوری‌که در هر حد قابل تصور از کاربرد مجموعه‌ی مجزا داریم: $\alpha(m,n) \leq ۴$ (اثباتِ تاحدی ضعیف‌تر این موضوع را می‌توانید در [۵] مشاهده نمایید.

حال با توجه به نکات گفته شده زمان اجرای الگوریتم پیاده‌سازی شده از $O(m\log n+m)$ خواهد بود که اندکی با حد ارایه شده در الگوریتم Gabow تفاوت دارد.

۴.٦ ورودی و خروجی

توضیحات مربوط به ورودی و خروجی الگوریتم در ذیل آمده است.

ورودی در فایل APPROX.IN می‌باشد و شامل n (تعداد رئوس) در یک سطر و سپس در $n-۱$ سطربعدی شامل ماتریس مجاورت گراف ورودی به‌صورت بالامثلثی (بدون قطر اصلی) است. ضمناً فرض شده است که تعداد رئوس از ۱۰۰ کمتر می‌باشد.

خروجی در فایل APPROX.OUT قرار دارد که در آن ابتدا تعداد مسیرها و سپس خود مسیرها— هر کدام در یک سطر— آمده‌اند.

```
APPROX.IN                                   APPROX.OUT
12
 1 30 10  0  0  0  0  0  0  0  0            The number of pathes is:6
    2  3  3  0  0  0  0  0  0  0            and the pathes are ...
       0  0  0  0  0  0  0  0  0            3 2 4
          0  0  0  0  0  0 15  0            1 2
             1  1  2  0  0  0  0            5 6
                0  0  0  0  0  0            9 8 10
                   0  0 20  1  0            7 5 8
                      2  3  0  0            11 12
                         0  0 99
                            0  0
                               2
```

شکل ٣.٦: یک نمونه از فایل خروجی برنامه تقریبی

مثال: نمونه‌ای از فایل ورودی و خروجی برای شکل ١.٦، در شکل ٣.٦ آمده است.

فصل ۷

خلاصه و نتیجه‌گیری

در این پایان‌نامه یکی سری از مسایل مهم در زمینه‌ی داده‌پراکنی جزیی مورد بحث و بررسی قرار گرفت. ابتدا در فصل دوم به تعریف تطابق و الگوریتم‌های مربوطه پرداختیم. در این فصل، هم‌چنین به اثبات چند قضیه مهم که در اثبات آن‌ها از روش حالتی بحرانی استفاده می‌شد، اشاره کردیم و گفتیم که این روش، یک روش کارا و ساخت‌یافته برای حل بسیاری از مسایل ترکیباتی است که به‌صورت شرط لازم و کافی بیان می‌شوند. لذا در حل مسایل شبه‌تطابق می‌تواند، کاملاً راه‌گشا باشد.

سپس در فصل سوم به معرفی شبه‌تطابق و هم‌چنین تاریخچه‌ای از شکل‌گیری آن اشاره نمودیم و بسیاری از بهینه‌سازی‌های ممکن روی این مساله گفته شد که اکثر این‌گونه مسایل بهینه‌سازی در حالت کلی NP-complete بودند ولی همان‌طور که در انتهای این فصل گفته شد، تحقیقات بسیاری می‌توان روی حالات خاص این مسایل انجام داد کما آنکه در مورد درخت‌ها به‌عنوان یک حالت خاص، در این فصل بحث و بررسی انجام شد. هم‌چنین در این فصل مدل‌های دیگری بجز مدل خطی تعریف شده و اشاره شد که مسایل مشابهی نیز روی این مدل‌ها می‌تواند مورد بررسی قرار گیرند.

در فصل چهارم که اساساً بر روی نتایج به‌دست آمده توسط خود ما تاکید داشت نیز مساله‌های باز زیادی در انتهای فصل مطرح شد که از آن جمله می‌توان از حل مساله شبه‌تطابق برای حالت‌های خاص که طول مسیرها برابر یک kای ثابت است، نام برد. هم‌چنین در این فصل تبدیل‌های بسیار زیبایی برای حل بعضی از مسایل شبه‌تطابق مورد استفاده قرار گرفت که می‌تواند در حل مسایل بعدی

کاربرد داشته باشد.

در فصل پنجم به دیگر کارهای انجام شده در زمینه بحث شبه‌تطابق و به‌طور کلی‌تر مسیرهای مجزا و تعمیم‌های جالبی که در این زمینه وجود دارد، پرداخته‌شد که هر کدام به نوبه‌ی خود جای تحقیقات زیادی دارد.

سرانجام در فصل ششم به‌ذکر یک نوع خاص از مساله شبه‌تطابق، بنام شبه‌تطابق تنگنایی پرداخته شد و یک الگوریتم ۱—تقریب چند جمله‌ای برای این مساله عنوان، اثبات و پیاده‌سازی شد. در پیاده‌سازی این الگوریتم از روشی ابتکاری استفاده شد که از لحاظ زمان اندکی با زمان عنوان شده در شرح الگوریتم تفاوت داشت ولی پیاده‌سازی بسیار جالب و ساده‌ای داشت. این الگوریتم هنوز جای کار دارد و می‌توان سعی کرد که زمان را به درجه‌ی پیچیدگی ذکر شده نزدیک‌تر کرد. ولی هدف از این فصل اساساً یادآوری الگوریتم‌های تقریبی برای حل مسایل شبه‌تطابق بود که می‌تواند به‌صورت بسیار گسترده در مورد اکثر مسایل شبه‌تطابق بکار رفته و با تقریب‌های خوب مساله‌هایی را که اکثراً نیز کاربردی هستند، حل کند. هم‌چنین می‌توان از الگوریتم‌های مکاشفه‌ای و شبیه‌سازی برای اثبات آن‌ها استفاده نمود که در این پایان‌نامه کمتر به این‌گونه از الگوریتم‌ها پرداخته شده است.

پیوست A

متن برنامه‌ی اول

در این پیوست، متن برنامه‌یی که برای پیاده‌سازی الگوریتم تطابق بیشینه یعنی الگوریتمی که در فصل ?? ذکر شد، به کار رفته است خواهد آمد.

```
{MATCH.PAS}
{$A+,B-,D+,E+,F-,G-,I+,L+,N-,O-,P+,Q+,R+,S+,T-,V+,X+}
{$M 16384,0,655360}

program matching_1;
const
  max=53;
type rec=record
             h:integer;
         end;
var
  rec1:rec;
  cycle,g,graph:array[1..2*max,1..2*max]of byte;
  match,path,path1,fil:array[0..2*max]of integer;
  maxn,c,c1,c2,n,m,i,j:integer;
  f:text;
  lencyc,pathtem:array[0..2*max]of integer;


procedure blossem(var s,len:integer);
```

```
var coun,i,j:integer;
    set1:set of 1..100;
begin
  inc(maxn);
  coun:=maxn-n;
  set1:=[s];
  i:=len-1;
  lencyc[coun]:=0;
  while path[i]<>s do begin
    include(set1,path[i]);
    inc(lencyc[coun]);
    cycle[coun,lencyc[coun]]:=path[i];
    path[i]:=0;
    dec(i);
  end;
  inc(lencyc[coun]);
  cycle[coun,lencyc[coun]]:=path[i];
  len:=i;
  path[i]:=maxn;
  s:=maxn;
  for i:=1 to lencyc[coun] do
    for j:=1 to maxn do
      if not(j in set1 ) and (g[cycle[coun,i],j]=1) then begin
        g[maxn,j]:=1;
        g[j,maxn]:=1;
        g[j,cycle[coun,i]]:=0;
        g[cycle[coun,i],j]:=0;
      end;
  fil[maxn]:=1;
end;


function dfs(v,k:integer):integer;
var
  c,i,j:integer;
begin

  for i:=1 to maxn do
    if (g[v,i]=1)and (fil[i]=1) then  begin
      if path[2]>0 then begin
        blossem(i,k);
```

```
        c:=dfs(i,k+1);
        dfs:=c;
        exit;
      end;
    end
    else  if (g[v,i]=1)and (fil[i]=0) then  begin
       path[k]:=i;
       c:=match[i];
       if c=0 then begin
  dfs:=k;
  exit;
       end
       else begin
  fil[i]:=2;
         path[k+1]:=c;
         fil[c]:=2;
  c:=dfs(c,k+2);
         if c=0 then dfs:=0
  else begin
           dfs:=c;
    exit;
         end;
       end;
    end;
  dfs:=0;
end;

procedure open;
var i,j,k,h,coun:integer;
  cyctem:array[1..100]of byte;
begin
  for h:=n+1 to maxn do  begin
    i:=1;
    coun:=h-n;
    while i<=lencyc[coun] do begin
      if cycle[coun,i]>n then begin
        for j:=1 to i-1 do
          cyctem[j]:=cycle[coun,j];
        j:=i-1;
        for k:=1 to lencyc[cycle[coun,i]-n] do begin
          inc(j);
```

```
          cyctem[j]:=cycle[cycle[coun,i]-n,k];
        end;
        c:=j;
        for k:=i+1 to lencyc[coun] do begin
          inc(j);
          cyctem[j]:=cycle[coun,k];
        end;
        lencyc[coun]:=j;
        for k:=1 to j do
          cycle[coun,k]:=cyctem[k];
        i:=c;
      end
      else
        inc(i);
    end;
  end;
end;

procedure getver(var len,s:integer);
var i,j,l,k:integer;
begin

  for i:=1 to len-1 do begin
    if (path[i]>n) and (path[i+1]<=n) then begin
      for j:=1 to lencyc[path[i]-n] do
        if graph[path[i+1],cycle[path[i]-n,j]]=1 then begin
          path1[2*i]:=path[i+1];
          path1[2*i-1]:=cycle[path[i]-n,j];
          break;
        end;
    end;
    if (path[i]<=n) and (path[i+1]<=n) then begin
      path1[2*i]:=path[i+1];
      path1[2*i-1]:=path[i];
    end;
    if (path[i]<=n) and (path[i+1]>n) then begin
      for j:=1 to lencyc[path[i+1]-n] do
        if graph[path[i],cycle[path[i+1]-n,j]]=1 then begin
          path1[2*i-1]:=path[i];
          path1[2*i]:=cycle[path[i+1]-n,j];
          break;
```

```
        end;
    end;
    if (path[i]>n) and (path[i+1]>n) then begin
      for j:=1 to lencyc[path[i]-n] do
        for l:=1 to lencyc[path[i+1]-n] do
        if graph[cycle[path[i+1]-n,k],cycle[path[i]-n,j]]=1 then begin
          path1[2*i]:=cycle[path[i+1]-n,k];
          path1[2*i-1]:=cycle[path[i]-n,j];
          break;
        end;
    end;
  end;
  if path1[1]<>s then begin
    for i:=2*len-2 downto 1 do
      path1[i+2]:=path1[i];
    path1[2]:=s;
    path1[1]:=0;
    inc(len,1);
  end;
end;

function plus(i,b:integer):integer;
begin
  if i=b then plus:=1
  else
    plus:=i+1;
end;

function mins(i,b:integer):integer;
begin
  if i=1 then mins:=b
  else
    mins:=i-1;
end;


function getpath(len,s:integer):integer;
var
  i,j,h,k,m,c1,c2,l:integer;
begin
  i:=1;
```

```
open;
getver(len,s);
j:=0;
l:=0;
c2:=0;
c1:=0;
for i:=1 to 2*len-3 do begin
  if i mod 2=1 then begin
    if path1[i]<>0 then begin
      inc(j);
      pathtem[j]:=path1[i];
    end
    else begin
      l:=1;
      inc(j);
      pathtem[j]:=path1[2];
    end;
  end;
  if i mod 2=0 then
    if path1[i]<>path1[i+1] then begin
      h:=i div 2+1;
      if l=1 then dec(h);
      h:=path[h]-n;
      for m:=1 to lencyc[h] do
        if cycle[h,m]=path1[i] then begin
          c1:=m;
          break;
        end;
      for m:=1 to lencyc[h] do
        if cycle[h,m]=path1[i+1] then begin
          c2:=m;
          break;
        end;
      if c1>c2 then begin
        if (c1-c2) mod 2=0 then begin
          m:=c1-1;
          repeat
            inc(j);
            pathtem[j]:=cycle[h,m];
            dec(m);
          until m=c2;
```

```
          end;
          if (c1-c2) mod 2=1 then begin
            m:=plus(c1,lencyc[h]);
            repeat
              inc(j);
              pathtem[j]:=cycle[h,m];
              m:=plus(m,lencyc[h]);
            until m=c2;
          end;
        end;
        if c1<c2 then begin
          if (c2-c1) mod 2=0 then begin
            m:=c1+1;
            repeat
              inc(j);
              pathtem[j]:=cycle[h,m];
              inc(m);
            until m=c2;
          end;
          if (c2-c1) mod 2=1 then begin
            m:=mins(c1,lencyc[h]);
            repeat
              inc(j);
              pathtem[j]:=cycle[h,m];
              m:=mins(m,lencyc[h]);
            until m=c2;
          end;
        end;
      end;
  end;
  inc(j);
  pathtem[j]:=path1[2*len-2];
  getpath:=j;
end;

procedure afsayesh(l:integer);
var i:integer;
begin
  for i:=1 to l-1 do
    if i mod 2=1 then begin
      match[pathtem[i]]:=pathtem[i+1];
```

```
      match[pathtem[i+1]]:=pathtem[i];
    end;
end;
procedure matching;
var
c,ch,i,j:integer;
begin
  c:=0;
  for i:=1 to n do begin
    ch:=0;
    fillchar(fil,sizeof(fil),0);
    g:=graph;
    maxn:=n;
    for j:=1 to n do  begin
      if (match[j]=0) then begin
 fillchar(path,sizeof(path),0);
 path[1]:=j;
       fil[j]:=1;
 c:=dfs(j,2);
 if c<>0 then begin
          c:=getpath(c,j);
   afsayesh(c);
          ch:=1;
          break;
        end;
      end;
    end;
    if ch=0 then break;
  end;
end;



begin {main}
  assign(f,'Match.in');
  reset(f);
  readln(f,n);
  read(f,c1,c2);
  while c1<>0 do begin
    graph[c1,c2]:=1;
    graph[c2,c1]:=1;
```

```
    read(f,c1,c2);
  end;
  close(f);
  maxn:=n;
  matching;
  Maxn:= 0;
  for i:=1 to n do begin
    if i mod 24=0 then readln;
    writeln(i,'  ',match[i]);
    if match[i]<>0 then inc(j);
    if Match[i]= 0 then inc(Maxn);
  end;
  writeln('The number of match edge is:',j div 2);
  WriteLn(MaxN);
  assign(f,'Match.out');
  rewrite(f);
  writeln(f,'The number of matching edges is:',j div 2);
  for i:=1 to n do begin
    if (Match[i]<> 0) and (i<= Match[i]) then
      writeln(f, i,'  ',match[i]);
    if match[i]<0 then inc(j);
    if Match[i]= 0 then inc(Maxn);
  end;
  Close(f);
end.
```

پیوست B

متن برنامه‌ی دوم

در این پیوست، متن برنامه‌یی که برای پیاده‌سازی الگوریتم ۱—تقریبی شبه‌تطابق تنگنایی یعنی الگوریتمی که در فصل ؟؟ ذکر شد، به کار رفته است خواهد آمد.

```
{APPROX.PAS}
{$A+,B-,D+,E+,F-,G+,I+,L+,N+,O-,P-,Q+,R+,S+,T-,V+,X+,Y+}
{$M 64000,0,655360}

program Approximation;
const
  Max= 100;
  MAXBYTE= 125;
type
  vertex=record
           rank,
           p:integer;
         end;
  treedis=array[1..max]of vertex;

var
  G, G1: Array[1..Max, 1..Max]of Byte;
```

```
  n, m , NPath: Integer;
  E: Array[1..5000,1..2]of Byte;
  Path: Array[1..Max, 1..4]of Byte;
  Fil, Root: Array[1..Max]of Byte;
  settree:treedis;


procedure MakeSet(x:integer);
begin
  settree[x].p:=x;
  settree[x].rank:=0;
end;

function FindSet(x:integer):integer;
begin
  if x<>settree[x].p then
    settree[x].p:=FindSet(settree[x].p);
  FindSet:=settree[x].p;
end;


procedure Link(x,y:integer);
begin
  if settree[x].rank>settree[y].rank then
    settree[y].p:=x
  else begin
    settree[x].p:=y;
    if settree[x].rank=settree[y].rank then
      settree[y].rank:=settree[y].rank+1;
  end;
end;

procedure Union(x,y:integer);
begin
```

```
  Link(FindSet(x),FindSet(y));
end;


procedure Input;
var
  i, j: Integer;
  f: Text;
begin
  Assign(f, 'Approx.in');
  Reset(f);
  ReadLn(f, n);
  m:= 0;
  for i:= 1 to n do begin
    for j:= i+ 1 to n do begin
      Read(f, G[i, j]);
      if G[i, j]<> 0 then begin
        inc(m);
        E[m ,1]:= i;
        E[m, 2]:= j;
      end
      else begin
        G[i, j]:= MAXBYTE;
      end;
      G[j, i]:= G[i, j];
    end;
  end;
  if n mod 2= 1 then begin
    for i:= 1 to n do begin
      G[i, n+ 1]:= MaxByte;
      G[n+ 1, i]:= MaxByte;
      inc(m);
      E[m, 1]:= i;
      E[m, 2]:= n+ 1;
```

```
    end;
    inc(n);
  end;
  Close(f);
end;

procedure QuickSort(Lo, Hi: Integer);

procedure Sort(l, r: Integer);
var
  i, j, x, y: integer;
begin
  i := l; j := r; x := G[E[(l+r) DIV 2, 1],E[(l+r) DIV 2, 2]];
  repeat
    while G[E[i, 1], E[i, 2]] < x do i := i + 1;
    while x < G[E[j, 1], E[j, 2]] do j := j - 1;
    if i <= j then
    begin
      y := E[i, 1]; E[i, 1] := E[j, 1]; E[j, 1] := y;
      y := E[i, 2]; E[i, 2] := E[j, 2]; E[j, 2] := y;
      i := i + 1; j := j - 1;
    end;
  until i > j;
  if l < j then Sort(l, j);
  if i < r then Sort(i, r);
end;

begin {QuickSort};
  Sort(Lo,Hi);
end;

procedure Dfs(v: Integer);
var
  i: Integer;
```

```
begin
  Fil[v]:= 1;
  for i:= 1 to n do begin
    if (G1[v, i]= 1) and (Fil[i]= 0) then
      Dfs(i);
  end;
end;


procedure FindMST;
var
  i, j, sum: Integer;
  c1,c2,coun:integer;
begin
  for i:=1 to n do
    MakeSet(i);

  QuickSort(1, m);
  sum:= 0;
  coun:=0;
  for i:=1 to m do begin
    c1:=FindSet(E[i, 1]);
    c2:=FindSet(E[i, 2]);
    if c1<>c2 then begin
      Union(c1,c2);
      inc(coun);
      sum:= sum+ G[E[i, 1], E[i, 2]];
      G1[E[i, 1], E[i, 2]]:= 1;
      G1[E[i, 2], E[i, 1]]:= 1;
    end;
    if coun=n-1 then break;
  end;
  FillChar(fil, Sizeof(fil), 0);
end;
```

```
procedure Direct(v: Integer);
var
  i: Integer;
begin
  for i:= 1 to n do begin
    if (G1[v, i]= 1) then begin
      G1[i, v]:= 0;
      Direct(i);
    end;
  end;
end;

function Mark(v: Integer): Integer;
var
  i, j: Integer;
begin
  Fil[v]:= 1;
  for i:= 1 to n do begin
    if (G1[v, i]= 1) then begin
      j:= Mark(i);
      Fil[v] :=(fil[v]+ j)mod 2;
      if((j mod 2)= 0) then begin
        G1[v, i]:= 0;
        Root[i]:= 1;
      end;
    end;
  end;
  Mark:= Fil[v];
end;

procedure Create(v: Integer);
var
```

```
  i, j, tmp: integer;
begin
  if Fil[v]= 0 then begin
    tmp:= v;
    Fil[v]:= 1;
  end
  else tmp:= 0;
  for i:= 1 to n do begin
    if G1[v, i]= 1 then begin
      Fil[i]:= 1;
      Create(i);
      if tmp= 0 then tmp:= i
      else begin
        inc(NPath);
        if (tmp= v) then begin
          Path[NPath, 1]:= v;
          Path[NPath, 2]:= i;
        end
        else begin
          Path[NPath, 1]:= tmp;
          Path[NPath, 2]:= v;
          Path[NPath, 3]:= i;
        end;
        tmp:= 0;
      end;
    end;
  end;
end;

procedure FindSDPM;
var
  i: Integer;
begin
  FillChar(Fil, Sizeof(Fil), 0);
```

```
  Root[1]:= 1;
  for i:= 1 to n do begin
    if Root[i]= 1 then begin
      Create(i);
    end;
  end;
end;

procedure Output;
var
  i, j: Integer;
  f: Text;
begin
  Assign(f, 'Approx.out');
  Rewrite(f);
  WriteLn(f, 'The number of pathes is:',NPath);
  WriteLn(f, 'and the pathes are ...');
  for i:= 1 to n do begin
    j:= 1;
    while Path[i, j]<> 0 do begin
      Write(f, Path[i, j], ' ');
      inc(j);
    end;
    WriteLn(f);
  end;
  Close(f);
end;


begin {Main}
  Input;
  FindMST;{The Result (G1) is a Minimum Spanning Tree}
  Direct(1);
  Mark(1);
```

```
  FindSDPM;
  Output;
end.
```

.

پیوست C

واژه‌نامه‌ی فارسی به انگلیسی

آلوده‌شده ..... Saturated
ابرمکعب ..... Hypercube
الگوریتم تقریبی ..... Approximation algorithm
انتقال‌دهنده ..... Transmitter
اندیس تطابقی ..... Matching index
بار راس ..... Vertex load
پوشش راسی ..... Vertex cover
تحقق‌یافته ..... Realized
تطابق ..... Matching
تطابق کامل ..... Perfect matching
تطابق متقاطر ..... Antipodal matching
تطابق مسیری ..... Path-matching
تقریباً متعادل ..... Nearly balanced
جایگشت ..... Permutation
جنگل ریشه‌دار ..... Rooted forest
حرف ..... Literal
داده‌پراکنی جزیی ..... Multicasting
داده‌پراکنی کلی ..... Broadcasting

درخت اشتاینر Steiner tree

درخت پوشای کمینه Minimum spanning tree

درخت داده‌پراکنی جزیی بهینه Optimal multicasting tree

k—رنگ‌پذیر k-Colorable

ستاره Star

شبکه اتصال‌داخلی Interconnection network

شبه‌تطابق Pseudomatching

شبه‌تطابق تنگنایی Bottleneck pseudomatching

شکوفه Blossom

f—فاکتور f-Factor

کلمه Clause

گراف دوبخشی Bipartite graph

گراف مسیریابی Routing graph

گراف میزبان Host graph

مجموعه‌ی مجزا Disjoint-set

مساله ارضاشدن Satisfiability problem

مساله شایعه‌پراکنی Gossiping problem

مسطح Plannar

مسیر افزایشی Augmenting path

مسیر تناوبی Alternating path

مسیر مجزای یالی(راسی) Edge(vertex)-disjoint path

مسیریاب Router

مسیریابی تعویض مدار Circuit-switch routing

پیوست D

واژه‌نامه‌ی انگلیسی به فارسی

مسیر تناوبی ..... Alternating path

تطابق متقاطر ..... Antipodal matching

الگوریتم تقریبی ..... Approximation algorithm

مسیر افزایشی ..... Augmenting path

گراف دوبخشی ..... Bipartite graph

شکوفه ..... Blossom

شبه‌تطابق تنگنایی ..... Bottleneck pseudomatching

داده‌پراکنی کلی ..... Broadcasting

مسیریابی تعویض مدار ..... Circuit-switch routing

کلمه ..... Clause

k—رنگ‌پذیر ..... k-Colorable

مجموعه‌ی مجزا ..... Disjoint-set

مسیر مجزای یالی(راسی) ..... Edge(vertex)-disjoint path

f—فاکتور ..... f-Factor

مساله شایعه‌پراکنی ..... Gossiping problem

گراف میزبان ..... Host graph

ابرمکعب ..... Hypercube

شبکه اتصال‌داخلی ..... Interconnection network

Literal ..... حرف
Matching ..... تطابق
Matching index ..... اندیس تطابقی
Minimum spanning tree ..... درخت پوشای کمینه
Multicasting ..... داده‌پراکنی جزیی
Nearly balanced ..... تقریباً متعادل
Optimal multicasting tree ..... درخت داده‌پراکنی جزیی بهینه
Path-matching ..... تطابق مسیری
Perfect matching ..... تطابق کامل
Permutation ..... جایگشت
Plannar ..... مسطح
Pseudomatching ..... شبه‌تطابق
Satisfiability problem ..... مساله ارضاشدن
Saturated ..... آلوده‌شده
Star ..... ستاره
Steiner tree ..... درخت اشتاینر
Realized ..... تحقق‌یافته
Rooted forest ..... جنگل ریشه‌دار
Router ..... مسیریاب
Routing graph ..... گراف مسیریابی
Transmitter ..... انتقال‌دهنده
Vertex cover ..... پوشش راسی
Vertex load ..... بار راس

کتاب‌نامه

[1] J. A. Mchugh, *Algorithmic Graph Theory*, Prentice-Hall, 1990.

[2] Ian. Anderson, *Combinatorial Design*, Ellis Horwood Limited, 1990.

[3] Michael R. Garey, David S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, 1979.

[4] J. A. Bondy, U. S. R. Murty, *Graph Theory with Applications*, American Elsevier Publishing, 1976.

[5] T. H. Cormen, C. .E. .Leiserson, R. L .Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1989.

[6] D. B. West, *Introduction to Graph Theory*, Prentice-Hall, 1996.

[7] T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*, Morgan Kaufmann, 1992.

[8] L. Lovász, M. D. Plummer, *Matching Theory*, Akadémiai Kiadó, Budapest, 1986.

[9] R. K. Ahuja, T. L. Magnanti and J. B. Orlin. *Network Flows* (Theory, Algorithms, and Applications), Prentice Hall, Englewood Cliffs, NJ, 1993.

[10] E. S. Mahmoodian, *The Pamphlet of Combinatorics*, Sharif University Press, Tehran, 1995.

[11] D. Kleitman, A. Martin-Lof, B. Rothschild, A. Whinston, " A matching theorem for graphs", *J. Combinatorial Theory 8*, 1970, pp. 104-114.

[12] X. Lin, P. K. McKinley, A. H. Esfahanian, " Adaptive Multicast Wormhole Routing in 2D Mesh Multicomputers", *Journal of Parallel and Distributed Computing 28*, 1995, pp. 19-31.

[13] R. Hassin, S. Rubinstein, " An Approximation Algorithm for Maximum Packing of 3-Edge Paths", *IPL 63*, 1997, pp. 63-67.

[14] A. K. Datta, R. K. Sen, " 1-Approximation algorithm for bottleneck disjoint path matching", *Information Processing Letters 55*, 1995, pp. 41-44.

[15] J. Peters, M. Syska, " Circuit-Switched Broadcasting in Torus Networks", *IEEE Transactions on Parallel and Distributed Systems 7*, 1996, pp. 246-255.

[16] P. Slater, E. Cockayne, S. Hedetniemi, " Information Dissemination in Trees", *SIAM J. Computing 10*, 1981, pp. 692-701.

[17] B. Bollobas, I. Leader, " Matching and paths in the cube", *Discrete Applied Mathematics 75*, 1997, pp. 1-8.

[18] H. Yinnone, " Maximum number of disjoint paths connecting specified terminals in a graph", *Discrete Applied Mathematics 55*, 1994, pp. 183-195.

[19] A. Farley, " Minimum-Time Line Broadcast Networks", *Networks 10*, 1980, pp. 59-70.

[20] Y. Lan, A. H. Esfahanian, L. M. Ni, " Multicast in Hypercube", *Journal of Parallel and Distributed Computing 8*, 1990, pp. 30-41.

[21] Q. P. Gu, H. Tamaki, " Multi-color routing in the undirected hypercube", *Algorithms and Computation(Singapore)*, 1997, pp. 72-81.

[22] J. Vygen, " NP-completeness of some edge-disjoint paths problems", *Discrete Applied Mathematics 61*, 1995, pp. 83-90.

[23] C. T. Ho, M. Y. Kao, " Optimal Broadcast in All-port Wormhole-Routed Hypercubes", *IEEE Transactions on Parallel and Distributed Systems 6*, Feb. 1995, pp. 200-318.

[24] J. Cohen, P. Fraigniaud, J. C. Koing, A. Raspaud, " Optimized Broadcasting and Multicasting Protocols in Cut-Through Routed Networks", *IEEE Transactions on Parallel and Distributed Systems 9*, August 1998, pp. 788-802.

[25] R. J. Faudree, A. Gyarfas , " Path-matching in graphs", *Congressus Numerantium 120*, 1996, pp. 83-96.

[26] E. Dahlhaus, M. Karpiński, " Parallel construction of perfect matchings and Hamiltonian cycles on dense graphs", *Theoretical Computer Science 61*, 1988, pp. 121-136.

[27] S. Wu, U. Manber, " Path-matching problems", *Algorithmica 8*, 1992, pp. 89-101.

[28] J. Cohen, P. Fraigniaud, M. Mitjana, " Scheduling calls for multicasting in tree-networks".

Multicasting and Pseudomatching

Abstract

This dissertation adresses the one-to-all broadcasting, usually simply called multicasting in which a node of a network sends the same piece of information to a subset of nodes forms the destination set. This operation has many applications in parallel and distributed computing. In this dissertation, we study this problem in line model which assumes long distance calls between nonneighboring processors.

In the line model, it is well known that one can compute in polynomial time a $\lceil \log_7(d) \rceil$-round(d is the number of destination nodes) multicast protocol for any arbitrary network, but such a protocol does not use the resource of the network in a balanced way. In this dissertation, we not only present a more efficient algorithm, but also show that most of the optimization problems relative to the maximization of efficiency of multicast protocols in terms of switching time, vertex load or path length are NP-complete. After modeling network by graph, most of these problems are reduced to pseudomatching problems where pseudomatching is a set of edge disjoint pathes connecting disjoint pairs of vertices.

We, also, present an approximation algorithm for special type of pseudomatching problem and an algorithm for finding maximum matching in general graphs. We study special types of pseudomatching in hypercube, also.

Keywords: *Broadcasting, MultiCasting, Matching, Pseudomatching(Path-matching), Edge(Vertex) disjoint paths, f-Factor, Approximation algorithms*

Multicasting and Pseudomatching

by

MohammadTaghi Hajiaghayi

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Bachelor of Science
in
Computer Engineering (Software)

Under supervision of
Dr. Mohammad Ghodsi

July 2000

Computer Engineering Department

Sharif University of Technology

Tehran