DOR: 20.1001.1.2538161.1404.10.4.4.6

DOI : 10.22034/csj.2025.237375

زمستان ۱۴۰۴، دوره ۱۰، شماره ۴، پیاپی ۳۹

دریافت مقاله: ۱۴۰۴/۰۲/۲۹

پذیرش مقاله: ۱۴۰۴/۰۷/۲۶

نوع مقاله: پژوهشی

معکوس مسئله مکان‌یابی ۱–مرکز روی گراف دور با افزایش طول یال‌ها

ابوالفضل پورعیدی*

گروه ریاضی کاربردی و علوم کامپیوتر، دانشکده علوم ریاضی، دانشگاه صنعتی شاهرود، شاهرود، ایران

پست الکترونیکی: a.poureidi@shahroodut.ac.ir

جعفر فتحعلی

گروه ریاضی کاربردی و علوم کامپیوتر، دانشکده علوم ریاضی، دانشگاه صنعتی شاهرود، شاهرود، ایران

پست الکترونیکی: fathali@shahroodut.ac.ir

چکیده

در سال‌های اخیر معکوس مسائل مکان‌یابی روی شبکه‌ها موضوع تعداد زیادی از تحقیقات بوده است. مسئله مکان‌یابی ۱–مرکز مطلق در یک شبکه عبارت است از یافتن یک نقطه روی شبکه به عنوان سرویس‌دهنده به طوری که فاصله دورترین نقطه روی گراف تا سرویس‌دهنده حداقل شود. در این مقاله به بررسی معکوس مسئله مکان‌یابی ۱–مرکز مطلق روی یک گراف دور بدون وزن می‌پردازیم. یک گراف دور یک مسیر بسته است به طوری که درجه هر راس گراف ۲ است. در معکوس مسئله مکان‌یابی ۱–مرکز مطلق روی یک گراف دور یک راس در گراف مشخص شده است و هدف این است تا طول یال‌های گراف دور را افزایش دهیم به طوری که با انجام کمترین هزینه راس مشخص شده با طول یال‌های جدید یک ۱–مرکز مطلق گراف دور باشد. در این مقاله یک الگوریتم ترکیبیاتی برای حل این مسئله پیشنهاد می‌کنیم به طوری که پیچیدگی زمانی الگوریتم $O(n^2)$ برای گراف‌های دور با n راس خواهد بود.

واژه‌های کلیدی: مسئله مکان‌یابی ۱–مرکز مطلق شبکه‌ها، بهینه سازی معکوس، الگوریتم ترکیبیاتی، گراف دور.

۱ – مقدمه

مکان‌یابی با توجه به کاربردهایی که دارد یکی از موضوعات جذاب تحقیق در عملیات است. به عنوان مثال اگر بخواهیم بهترین مکان در یک شهر را برای ایجاد یک مرکز آتش نشانی، تاسیس یک مدرسه یا ساخت یک بیمارستان پیدا کنیم با یک مسئله مکان‌یابی مواجه هستیم. مسائل مکان‌یابی در چند دهه گذشته بسیار مورد توجه و تحقیق قرار گرفته‌اند چرا که تصمیم‌گیری در مورد انتخاب مکان یک سرویس‌دهنده برای تعداد مشتری می‌تواند تاثیرات اقتصادی، امنیتی، اجتماعی، فرهنگی و محیط‌زیستی بسیاری را به دنبال داشته باشد. به عنوان مثال ساخت یک یا چند پناهگاه در یک شهر برای زمانی است که شرایط غیرعادی و اضطراری است و افراد برای حفظ جان خود به پناهگاه که یک سازه امن و مستحکم است پناه می‌آورند. یکی از شاخصه‌های محل انتخاب پناهگاه می‌تواند این مورد باشد که هنگام بروز یک حادثه دورترین شهروند از پناهگاه در کوتاه‌ترین زمان ممکن

* نویسندهٔ مسئول

بتوانند خود را به پناهگاه برسانند. در یک مسئله مکان‌یابی کلاسیک فرض می‌کنیم که تعدادی مشتری داریم و هدف این است که بهترین مکان را برای قرار دادن یک یا چند سرویس‌دهنده پیدا کنیم طوری که یک تابع هدف (هزینه حمل و نقل، زمان سرویس‌دهی، سود و غیره) با توجه به محدویت‌های موجود مسئله بهینه شود. به روش‌های مختلفی می‌توان مسائل مکان‌یابی را دسته‌بندی کرد. یکی از این مدل‌ها مکان‌یابی شبکه است که در آن مکان انتخابی برای سرویس‌دهنده و مشتری‌ها روی گره‌ها و یال‌های یک شبکه (گراف) مورد مطالعه قرار می‌گیرند. در یک مسئله مکان‌یابی کلاسیک فرض می‌کنیم که تعدادی مشتری داریم و هدف این است که بهترین مکان را برای قرار دادن یک یا چند سرویس‌دهنده پیدا کنیم طوری که یک تابع هدف (هزینه حمل و نقل، زمان سرویس‌دهی، سود و غیره) با توجه به محدویت‌های موجود مسئله بهینه شود. به روش‌های مختلفی می‌توان مسائل مکان‌یابی را دسته‌بندی کرد. یکی از این مدل‌ها مکان‌یابی شبکه است که در آن مکان نامزدها برای سرویس‌دهنده و مکان مشتری‌ها روی گره‌ها و یال‌های یک شبکه (گراف) مورد مطالعه قرار می‌گیرند. مسئله مکان‌یابی به طور گسترده‌ای چه در زمینه‌های عملی و چه در زمینه‌های تئوری بررسی شده است. برای آشنایی و مطالعه بیشتر مبانی نظریه مسئله مکان‌یابی، مطالعه منابع [۲-۱] توصیه می‌شود.

در سال‌های اخیر مسائل مکان‌یابی معکوس که یکی از مسائل بهینه‌سازی معکوس است بسیار مورد توجه قرار گرفته است. در این مسائل، مکان سرویس‌دهنده‌ها از قبل مشخص است و قابل تغییر نیست. در مسائل مکان‌یابی معکوس در یک شبکه، سعی می‌کنیم با تغییر پارامترهای مسئله داده شده، مانند طول یال‌ها یا وزن راس‌ها، با توجه به محدودیت‌های موجود برای پارامترها، با صرف کردن کمترین هزینه، یک یا چند سرویس‌دهنده مشخص با توجه به مقدار پارامترهای جدید شرایط بهینگی را داشته باشند. مدل‌های مرکز و میانه دو مدل شناخته شده در مسائل مکان‌یابی روی شبکه‌ها هستند که در سال‌های اخیر بسیار مورد توجه قرار گرفته‌اند. در حالی که در مدل مرکز هدف این است که موقعیت سرویس‌دهندگان بهینه را در شبکه موردنظر پیدا کنیم به طوری که فاصله دورترین سرویس‌گیرندگان تا نزدیک‌ترین سرویس‌دهندگان حداقل شود، در مدل میانه هدف این است تا مجموع فاصله‌ها از محل سرویس‌گیرندگان تا نزدیک‌ترین سرویس‌دهنده حداقل شود.

ساختار مقاله به صورت زیر است. در بخش ۲ به بررسی کارهای قبلی در رابطه با موضوع مقاله می‌پردازیم. در بخش ۳ به معرفی مسئله مکان‌یابی معکوس ۱-مرکز روی شبکه‌ها می‌پردازیم و یک فرمول‌بندی برای مسئله به صورت یک مدل بهینه‌سازی پیشنهاد می‌کنیم. در بخش ٤ شرط بهینگی مسئله مکان‌یابی ۱-مرکز روی گراف دور بررسی خواهد شد. در ادامه این بخش مسئله را به تعدادی زیرمسئله تقسیم می‌کنیم به طوری که هر زیر مسئله دارای یک فرمول به صورت یک مدل بهینه‌سازی خطی است. در بخش ٥ برای حل هر زیرمسئله یک الگوریتم پیشنهاد می‌کنیم. در بخش ٦ با استفاده از الگوریتم پیشنهادی بخش قبل الگوریتم خود را برای حل مسئله ارائه می‌کنیم. در بخش ۷ روش حل پیشنهادی این مقاله را به صورت کلی مرور کرده‌ایم و پیشنهادهایی برای کارهای جدید ارائه شده است.

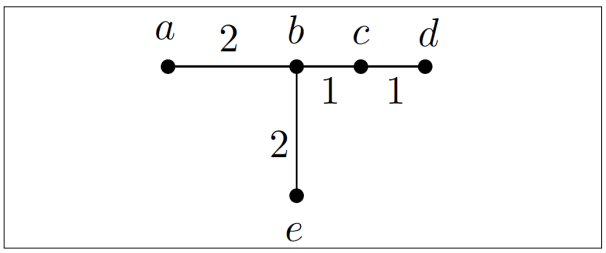
۲- مروری بر ادبیات

نظری و همکاران [۳] با استفاده از الگوریتم‌های فراابتکاری به حل مسئله مکان‌یابی معکوس ۲- میانه روی شبکه‌ها پرداختند. مسئله مکان‌یابی معکوس ۱- میانه روی تعدادی از شبکه‌ها بررسی شده است و برای آنها الگوریتم‌های کارآیی با پیچیدگی زمانی چندجمله‌ای برای به دست آوردن جواب بهینه مسئله روی شبکه پیشنهاد شده است. گلوی [٤] مسئله مکان‌یابی معکوس ۱- میانه روی یک درخت را با وزن‌های غیرمنفی و روی یک مسیر

بـــا وزن‌های مثبت/ منفی بررسی کرد. گان و همکاران [۵] مســـئله مکان‌یابی معکوس ۱- میانـــه روی یک درخت را تحت فاصله همینگ و نـــرم بینهایت بررسی کردند. فام و همکاران [٦] نیز مســـئله مکان‌یابی معکوس ۱- میانه روی درخت‌ها را برای نرم‌های چپیشـــف و مستطیلی بررسی کردند. روی شبکه‌های دور، مسئله مکان‌یابی معکوس ۱- میانه توســـط بوردکارد و همکاران [۷] بررسی شد و یک الگوریتم با پیچیدگی زمانی مربعی برای حل مســـئله ارائه دادند. برای گراف‌های بلوکی نگوین [۸] مســـئله مکان‌یابی معکوس ۱- میانه را بررسی کرد. مسئله معکوس ۲- میانه روی شبکه‌ها توســـط نظری و فتحعلی [۹] مورد بررسی قرار گرفـــت. نگوین و همکاران [۱۰] مســـئله معکوس ۱- میانـــه را روی درخت‌هـــا بررسی کردنـــد و یک الگوریتم بـــا پیچیدگی زمانی $O(n\log n)$ برای حل مســـئله ارائه کردند. مسئله مکان‌یابی معکوس ۱-مرکز روی شبکه‌های جهت‌دار بدون وزن با تغییر دادن طول یال‌ها یک مســـئله NP- کامل اســـت که توســـط کای و همـــکاران [۱۱] ثابت شده است. برای حل مســـئله مکان‌یابی معکوس ۱-مرکز روی درخت‌هـــای بدون وزن وقتی کـــه فقط مجاز باشیم طـــول یال‌ها را افزایش دهیم الگوریتمی با پیچیدگی زمانی $O(n\log n)$ توســـط علیزاده و همکاران [۱۲] ارائه شده اســـت و وقتی که مجاز باشیم طـــول یال‌ها را هم افزایش و هم کاهش دهیم الگوریتمی با پیچیدگی $O(n^2)$ توسط علیـــزاده و همکاران [۱۳] ارائه شـــده اســـت که n تعداد راس‌های درخت اســـت. حسن‌زاده و همکاران [۱٤] مسئله معکوس مرکز را در شبکه درخت‌ها تحت نرم‌های چپیشف و همینگ بررسی کردنـــد و الگوریتم‌های خطی برای حل مسئله ارائه دادند. علیزاده و همکاران [۱۲] برای حل مسئله مکان‌یابی معکوس ۱-مرکز روی درخت‌ها نشان دادند که تعدادی مسیر وجـــود دارد و باید بیـــن مسیرها مسیری را جستجو کنند که بیشـــترین طول را دارد (طولانی‌ترین مسیر) و ســـپس در بین مسیرهـــای باقی‌مانده مسیری را جستجو کنند که هزینه افزایش طول مسیر تا جایی که طول مسیر با طولانی‌ترین مسیر برابر شـــود کمینه باشد. برای هر مسیر یک ســـاختار درخت جســـتجو ساختند تا بتوان در بـــا پیمایش در عمق درخت در زمان $O(\log n)$ هزینه افزایش طول هر مسیر را محاسبه کرد. در این مقاله مسئله مکان‌یابی معکوس ۱-مرکز مطلق را روی گراف دور بدون وزن را بررسی می‌کنیم. این مســـئله وقتی که مجاز باشیم طول یال‌ها را هم افزایش و هم کاهش دهیم توسط نگوین [۱۵] بررسی شده است و یک الگوریتم با پیچیدگی زمانی $O(n^2\log n)$ برای آن ارائه شـــده اســـت، جـــایی که n تعداد راس‌های گراف اســـت. در این مقاله مسئله را برای حالـــتی که فقط مجاز باشیم طول یال‌هـــا را افزایش دهیم بررسی می‌کنیم و در نهایت یک الگوریتم برای حل مسئله با پیچیدگی زمانی $O(n\log n)$ ارائه می‌کنیم و الگوریتم پیشـــنهادی مقاله مســـئله را برای نمونه‌های داده شده به صورت دقیق حل می‌کند. توجه شود که مسئله مکان‌یابی معکوس ۱-مرکز مطلق را روی گراف دور بدون وزن برای حالـــتی که فقط مجاز باشیم طـــول یال‌ها را افزایش دهیم، در گذشـــته بررسی نشده است. اینجا مسئله را به تعدادی زیرمســـئله تبدیل می‌کنیم به طوری که در هر زیرمســـئله باید دو مسیر را بررسی کنیم. تعداد زیرمسئله‌ها از مرتبه تعداد راس‌های گراف دور اســـت. در هر زیرمســـئله طول یال‌های یک مسیر را باید بـــا کمترین هزینه افزایش دهیم تا جایی که با طول مسیر دومی برابر شـــود. برای افزایش طول یال‌های یک مسیر با کمترین هزینه نیاز داریم یال‌ها را بر حسب نسبت هزینه افزایش هر یال بر طول یال مرتب کنیم. برای این که این مرتب‌ســـازی را در هر زیرمســـئله انجام ندهیم برای یک زیرمسئله یال‌ها را مرتب می‌کنیم و برای حل زیرمسئله‌های بعدی با استفاده از فهرست مرتب شـــده یال‌های قبلی در یک زمان خطی یال‌های زیرمسئله جدید را مرتب می‌کنیم. در نهایت با حل همه زیرمســـئله‌ها یک الگوریتم ترکیبیاتی برای حل مســـئله روی گراف دور ارائه می‌کنیم.

۳ - تعریف مسئله مکان‌یابی معکوس ۱-مرکز روی شبکه‌ها و نمادگذاری

در این بخش مسئله مکان‌یابی معکوس ۱-مرکز روی یک شبکه را معرفی می‌کنیم. گراف $G=(V,E)$ را در نظر بگیرید به طوری که V مجموعه رئوس گراف و E مجموعه یال‌های گراف است و فرض کنید که هر یال $e\in E$ دارای طول مثبت $l(e)$ است. یک نقطه روی G یک راس گراف یا یک نقطه روی یک یال گراف است. فاصله بین دو نقطه a و b روی G را با $d_l(a,b)$ نمایش می‌دهیم که طول کوتاه‌ترین مسیر بین دو نقطه a و b روی G است. در اینجا فرض می‌کنیم مشتری‌ها فقط مجاز هستند روی راس‌های شبکه قرار بگیرند ولی یک مرکز به عنوان سرویس‌دهنده مجاز است روی راس‌ها یا یال‌های شبکه قرار داشته باشد. دنبال مرکزی (نقطه‌ای روی راس‌ها یا یال‌های شبکه) هستیم تا سرویس‌دهنده را در آنجا قرار دهیم به طوری که دورترین مشتری (نقطه‌ای روی راس‌ها) کمترین فاصله را تا سرویس‌دهنده داشته باشد. به عنوان مثال در شکل ۱ اگر سرویس‌دهنده در راس b قرار گیرد فاصله دورترین راس شبکه تا سرویس‌دهنده ۲ است و اگر سرویس‌دهنده در راس c قرار گیرد فاصله دورترین راس شبکه تا سرویس‌دهنده ۳ است.



شکل (۱): یک گراف با ۵ راس (درخت) که هر یال $e\in E$ دارای برچسب $l(e)$ است.

می‌توانیم مسئله مکان‌یابی ۱-مرکز روی گراف G را به صورت زیر فرمول‌بندی کنیم:

$$\min_{p\in G}\max_{v\in V} d_l(p,v) \qquad (۱)$$

نقطه $p^*\in G$ را یک ۱-مرکز مطلق گوییم اگر و فقط اگر p^* یک جواب بهینه برای (۱) باشد.

در مقابل مسئله کلاسیک مکان‌یابی ۱-مرکز روی یک شبکه، مسئله مکان‌یابی معکوس ۱-مرکز روی یک شبکه را معرفی می‌کنیم: فرض کنید s یک راس مشخص شده روی گراف G است. هدف این است که طول یال‌های گراف G را با کمترین هزینه افزایش دهیم به طوری که s یک ۱-مرکز مطلق با توجه به طول‌های جدید در گراف G باشد. همچنین فرض کنید که اگر $l(e)$ را به اندازه یک واحد افزایش دهیم هزینه‌ای برابر با $c^+(e)$ دارد. علاوه بر این برای افزایش طول هر یال $e\in E$ یک کران بالا $l_p(e)$ در نظر می‌گیریم. در نتیجه می‌توانیم مسئله مکان‌یابی معکوس ۱-مرکز روی گراف G را به صورت زیر بیان کنیم:

برای هر یال $e\in E$، طول یال $l(e)$ را به مقدار جدید $\tilde{l}(e)$ تغییر می‌دهیم به طوری که شرایط سه گانه زیر روی دهند:

۱) راس s یک ۱-مرکز مطلق روی گراف G با توجه به طول یال‌های جدید $\tilde{l}$ باشد، به عبارت دیگر

$$\max_{v\in V} d_{\tilde{l}}(v,s)\le \max_{v\in V} d_{\tilde{l}}(v,p)$$

۲) تابع هزینه

$$\sum_{e\in E} c^+(e)x(e)$$

حداقل شود که در آن $x(e)$ مقداری است که طول $l(e)$ را افزایش می‌دهیم.

۳) طول‌های جدید هر یال بین کران‌های تعیین شده قرار می‌گیرند؛ به عبارت دیگر برای هر $e\in E$ داریم

$$0\le \tilde{l}(e)\le l_p(e)$$

در نتیجه می‌توانیم مسئله مکان‌یابی معکوس ۱-مرکز

روی گراف G را به صورت زیر به عنوان یک مدل بهینه‌سازی فرمول‌بندی کنیم:

$$\begin{aligned} \min \quad & \sum_{e\in E} c^+(e)x(e) \\ s.t. \quad & \max_{v\in V} d_{\tilde{l}}(v,s) \le \max_{v\in V} d_{\tilde{l}}(v,p) \quad \forall p \in G \\ & \tilde{l}(e) = l(e) + x(e) \quad \forall e \in E \\ & 0 \le x(e) \le l^+(e) \quad \forall e \in E \end{aligned}$$

که در آن $l^+(e) = l_{\psi}(e) - l(e)$ حداکثر مقداری است که $l(e)$ می‌تواند افزایش یابد.

۴ - شرط بهینگی برای مسئله مکان‌یابی معکوس

۱-مرکز روی گراف دور

یک گراف دور یک مسیر بسته است که هر راس درجه ۲ دارد. در ادامه فرض کنید $C=(V,E)$ یک گراف دور با n راس است. به آسانی دیده می‌شود که اگر دور دو راس داشته باشد مسئله مکان‌یابی معکوس ۱-مرکز روی دور C جواب ندارد. بنابراین در ادامه مقاله فرض می‌کنیم که $n \ge 3$. دور C را طوری در صفحه در نظر بگیرید که راس‌های C روی محیط یک دایره باشند. راس‌های C را در جهت ساعتگرد شماره‌گذاری می‌کنیم به طوری که $V=\{v_1,\dots,v_n\}$ و $e_i=(v_i,v_{i+1})\in E$ برای هر $i\in\{1,\dots,n-1\}$ و $e_n=(v_n,v_1)\in E$. برای یک یال $e\in E$ مسیر $C(e)$ با حذف یال e از دور C به دست می‌آید. توجه کنید که $C(e)$ یک مسیر با n راس است. بدون کاستن از کلیت فرض کنید v_1 راس مشخص شده در دور C است و می‌خواهیم طول یال‌های دور C را طوری افزایش دهیم که v_1 یک ۱-مرکز برای گراف C باشد.

لم ۱. گراف دور $C=(V,E)$ را در نظر بگیرید. راس v_1 یک ۱-مرکز برای C است اگر و فقط v_1 یک نقطه وسط $C(e^*)$ است به طوری که $l(e^*)=\max_{e\in E} l(e)$.

برهان. اثبات این لم در [۱۵] آمده است.

در مسئله مکان‌یابی معکوس ۱-مرکز روی گراف دور C هدف این است که طول یال‌های دور C را طوری افزایش دهیم که راس v_1 (بدون کاستن از کلیت) شرط بهینگی لم ۱ را داشته باشد. برای یک یال $e_i=(v_i,v_{i+1})$ از دور C که $i\in\{2,\dots,n-1\}$ فرض کنید P_c^i مسیر $v_1,v_2,\dots,v_i$ و $P_{\overline{c}}^i$ مسیر $v_1,v_n,v_{n-1},\dots,v_{i+1}$ باشد. به آسانی دیده می‌شود که اگر v_1 یکی از رئوس یال e_i باشد مسئله یک جواب شدنی ندارد. برای این که شرط بهینگی لم ۱ برای راس v_1 و یال e_i برقرار شود باید طول یال‌های دور C را طوری افزایش دهیم به طوری که با توجه به طول یال‌های جدید، e_i دارای بزرگترین طول بین همه یال‌های دور C شود و v_1 نقطه وسط $C(e_i)$ باشد یا به عبارت دیگر طول‌های دو مسیر P_c^i و $P_{\overline{c}}^i$ برابر شوند. بنابراین برای حل مسئله مکان‌یابی معکوس ۱-مرکز روی گراف دور C باید e_i را هر کدام از یال‌های دور C در نظر بگیریم و شرط بهینگی لم ۱ را در این حالت برقرار کنیم و کمترین هزینه تغییرات طول یال‌ها را به دست آوریم. بین همه حالت‌های ممکن برای یال e_i، حالتی که حداقل هزینه را دارد جواب مسئله مکان‌یابی معکوس ۱-مرکز روی گراف دور C است. بنابراین زیرمسئله مکان‌یابی معکوس ۱-مرکز روی دور C و یال e_i را می‌توانیم به صورت زیر فرمول‌بندی کنیم.

$$\begin{aligned} \min \quad & \sum_{e\in E} c^+(e)x(e) \\ s.t. \quad & \sum_{e\in E(P_c)} \tilde{l}(e) = \sum_{e\in E(P_{\overline{c}})} \tilde{l}(e) \\ & \tilde{l}(e) \le \tilde{l}(e_i) \quad \forall e \in E\setminus\{e_i\} \\ & 0 \le x(e) \le l^+(e) \quad \forall e \in E \end{aligned}$$

فرض کنید که

$l_c^i = l(P_c^i) = l(e_1)+l(e_2)+\cdots+l(e_{i-1})$،

$l_{\overline{c}}^i = l(P_{\overline{c}}^i) = l(e_n)+l(e_{n-1})+\cdots+l(e_{i+1})$ و

$l^* = \max_{e\in E} \tilde{l}(e)$. با توجه به مقادیر l_c^i و $l_{\overline{c}}^i$ یکی از سه حالت زیر روی می‌دهد.

۱) فرض کنید $l^i_{cc} = l^i_c$. در ایـن حالت نیازی به تغییر طول یال‌های دو مسیر P^i_c و P^i_{cc} نداریم و فقط کافی است اگر $l^* - l(e_i) > 0$ طول یـال e_i را به اندازه $l^* - l(e_i)$ افزایـش دهیم، در غیر این صورت (یعنی اگر $l^* - l(e_i) \le 0$) لازم نیسـت طول هیچ یالی تغییر کنـد و v_1 یک ۱-مرکز برای C است.

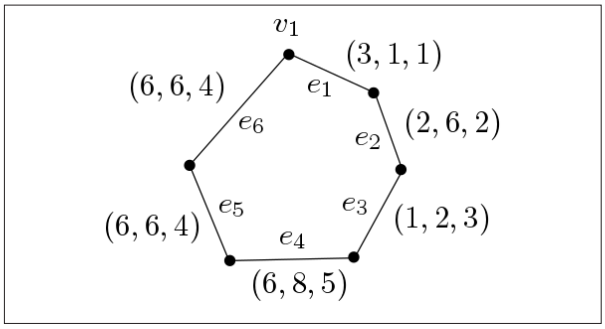
۲) فـرض کنید $l^i_{cc} < l^i_c$. در این حالت باید طول مسیر P^i_{cc} و طول یال e_i (در صورت نیاز) را به ترتیب به اندازه $l^i_c - l^i_{cc}$ و $l^* - l(e_i)$ (اگر $l^* - l(e_i) > 0$) افزایش دهیم. در واقع باید مدل برنامه‌ریزی خطی زیر را حل کنیم:

$$
\begin{aligned}
Inc^i_c : \min \quad & \sum_{e\in E(P_c)} c^+(e)x(e) \\
s.t. \quad & \sum_{e\in E(P_c)} \tilde{l}(e) = | l^i_c - l^i_{cc} | \\
& \tilde{l}(e) \le \tilde{l}(e_i) \quad \forall e \in E \setminus \{e_i\} \\
& 0 \le x(e) \le l^+(e) \quad \forall e \in E
\end{aligned}
$$

۳) فـرض کنید $l^i_{cc} > l^i_c$. در این حالت باید طول مسیر P^i_c و طول یال e_i (در صورت نیاز) را به ترتیب به اندازه $l^i_{cc} - l^i_c$ و $l^* - l(e_i)$ (اگر $l^* - l(e_i) > 0$) افزایش دهیم. در واقع باید مدل برنامه‌ریزی خطی زیر را حل کنیم:

$$
\begin{aligned}
Inc^i_{cc} : \min \quad & \sum_{e\in E(P_{cc})} c^+(e)x(e) \\
s.t. \quad & \sum_{e\in E(P_c)} \tilde{l}(e) = | l^i_c - l^i_{cc} | \\
& \tilde{l}(e) \le \tilde{l}(e_i) \quad \forall e \in E \setminus \{e_i\} \\
& 0 \le x(e) \le l^+(e) \quad \forall e \in E
\end{aligned}
$$

قبل از آن که الگوریتم خود را برای حل مسئله مکان‌یابی معکوس ۱-مرکز روی گراف دور C ارائه کنیم با یک مثال ایده الگوریتم را بیان می‌کنیم. شکل ۲ را ببنید. در شکل ۲ یک گراف دور با ٦ یال داریم و می‌خواهیم زیرمسئله مکان‌یابی معکوس ۱-مرکز روی یال e_4 را حل کنیم. دیده می‌شود که $l^4_c = 6$ و $l^4_{cc} = 12$ است. بنابراین، باید طول سه یال e_1، e_2 و e_3 را در مجموع به اندازه ٦ افزایش دهیم.



شکل (۲). یک گراف دور با ۶ یال که هر یال $e \in E$ دارای برچسب $(l(e), l^+(e), c^+(e))$ است.

قبل از افزایش طول یال‌های e_1، e_2 و e_3 می‌بینیم که e_4 بزرگترین طول را در بین همه یال‌های گراف دارد. با توجه بـه $l(e_1) + l^+(e_1) < l(e_4)$ و $c^+(e_1) < c^+(e_2) < c^+(e_3)$ ابتدا طول e_1 را به اندازه $l^+(e_1) = 1$ واحد افزایش می‌دهیم. در ادامه باید طول دو یال e_2 و e_3 را در مجموع به اندازه ٥ افزایـش دهیـم. با توجه بـه اینکه $c^+(e_2) < c^+(e_3)$ و $l(e_2) + 4 \le l(e_4)$ طـول e_2 را به انـدازه ٤ واحد افزایش می‌دهیـم. توجه کنید که اگر طـول e_2 را بیش از ٤ واحد افزایـش دهیم آنـگاه e_2 بزرگترین طـول را در بین همه یال‌هـای گراف خواهد داشـت و در نتیجه طول e_4 را نیز بایـد افزایش دهیم که افزایش هـر واحد طول e_4 ٥ واحد هزینه خواهد داشت. به عبارت دیگر افزایش هر واحد طول e_2 تـا ٤ واحد ۲ واحد هزینه دارد و افزایش هر واحد طول e_2 بعـد از ٤ واحد ۷ واحـد هزینه دارد. با توجه به این که $l(e_3) + l^+(e_3) \le l(e_4)$ و هزینـه افزایش هر واحد طول e_3 ۳ واحد اسـت که کمتر از ۷ واحد اسـت طول e_3 را ۱ واحد افزایش می‌دهیم. بنابراین جواب مسئله افزایش طول یال‌هـای e_1، e_2 و e_3 به ترتیب به انـدازه ۱، ٤ و ۱ واحد اسـت. با توجه به آن چه در بالا گفتیم برای حل زیرمسئله مکان‌یـابی معکوس ۱-مرکـز روی دور C و یال e_i دو هزینـه $c^+(e)$ و $c^+(e) + c^+(e_i)$ را باید برای هر یال e در نظر بگیریم.

۵. یک الگوریتم برای حل Inc_c^i و Inc_{cc}^i

در ایـــن بخش یک الگوریتم ترکیبیـــاتی (الگوریتم ٤,١) برای حـــل دو مدل برنامه‌ریزی خطی Inc_c^i و Inc_{cc}^i برای هر $i \in \{2,\ldots,n-1\}$ ارائه می‌کنیم.

لـــم ۲. فـــرض کنیـــد کـــه $b \in \{2,\ldots,n-1\}$، $l^* = \max\{l(e_1),\ldots,l(e_n)\}$، $cap = |l_{cc}^{k+1} - l_c^{k+1}|$ و $T = (T_1,\ldots,T_n)$ به طوری کـــه $T_i = \langle l(e_i)\ l^+(e_i)\ c^+(e_i)\rangle$ برای هر $i = 1,\ldots,n$. اگر $k = b-1$ و $A = (A_1,\ldots,A_{2k})$ به طوری که $A_j = \langle A_j^1, A_j^2\rangle$ بـــرای هـــر $j = 1,\ldots,2k$ و $A_j^1 \in \{1,\ldots,k\}$ و $A_j^2 \in \{0,1\}$ برای هر $c^+(e_{A_j^1}) + c^+(e_b)\times A_j^2 \le c^+(e_{A_{j+1}^1}) + c^+(e_b)\times A_{j+1}^2$، $j = 1,\ldots,2k-1$، آنـــگاه الگوریتم ٤,١ بـــا این ورودی‌ها زیرمسئله Inc_c^i را برای دور C و یال e_b حل می‌کند.

برهان. فـــرض کنید کـــه $l^* > l(e_b) + l^+(e_b)$. در این صـــورت حتی اگـــر طول یـــال e_b را به اندازه بیشـــترین مقـــدار ممکن یعنی $l^+(e_b)$ افزایش دهیـــم طول جدید e_b از مقدار l^* کمتر اســـت و در نتیجه شـــرایط لم ۱ برقرار نمی‌شـــود و زیرمسئله جواب شـــدنی ندارد. همچنین اگر $l^+(e_1) + \cdots + l^+(e_k) < cap$ زیرمســـئله جـــواب شـــدنی ندارد. در ادامه فرض می‌کنیم این دو شرط روی نمی‌دهند. الگوریتم ٤,١ این شرطها را در خط ۱ بررسی می‌کند.

Algorithm ٤,١: FK(b, k, T, A, cap, l^*)

Input: $b, k \in \{1,\ldots,n\}, T = (T_1,\ldots,T_n), A = (A_1,\ldots A_{2k}), T_i = \langle l(e_i), l^+(e_i), c^+(e_i)\rangle$ for $i \in \{1,\ldots,n\}$, $A_j = \langle \alpha_j, \beta_j\rangle$ for $j \in \{1,\ldots,2k\}$, where $\alpha_j \in \{1,\ldots,n\}$ and $\beta_j \in \{1,\ldots,n\}$, and $cap, l^* \ge 0$

Output: An optimal solution of the inverse 1-center location sub-problem on C

```
1  if (l* > l(e_b) + l+(e_b)) ∨ (l+(e_1) + ··· + l+(e_k) < cap) then
2        return No solution;
3  Z = max{l* − l(e_b), 0} × c+(e_b);
4  l(e_b) = l*;
5  C = 0;
6  for i = 1 to 2k do
7        t = α_i;
8        if β_i = 0 then {
9              if (l+(e_t) ≤ cap − C) ∧ (l+(e_t) ≤ l(e_b) − l(e_t)) then
10                   C = C + l+(e_t);
11                   Z = Z + l+(e_t) × c+(e_t);
12                   l+(e_t) = 0; }
13             else if l(e_b) − l(e_t) < cap − C then {
14                   C = C + l(e_b) − l(e_t);
15                   Z = Z + (l(e_b) − l(e_t)) × c+(e_t);
16                   l+(e_t) = l+(e_t) − (l(e_b) − l(e_t)); }
17             else if l(e_b) − l(e_t) ≥ cap − C then
18                   return Z + (cap − C) × c+(e_t);
19       Else
20             if (l+(e_t) ≤ cap − C) ∧ (l+(e_t) ≤ l+(e_b)) then
21                   C = C + l+(e_t);
22                   Z = Z + l+(e_t) × (c+(e_t) + c+(e_b));
23                   l+(e_t) = l+(e_b) − l+(e_t); }
24             else if l+(e_b) < cap − C then {
25                   C = C + l+(e_b);
26                   Z = Z + l+(e_b) × (c+(e_t) + c+(e_b));
27                   l+(e_b) = 0; }
28             else if cap − C ≤ l+(e_b) then
29                   return Z + (cap − C) × (c+(e_t) + c+(e_b));
30 if cap = C then
31       return Z;
32 Else
33       return No solution;
```

فرض کنید $l^* > l(e_b)$. با توجه به لم ۱ ضلع e_b باید بزرگترین طول را بین یال‌های گراف دور C داشته باشد. بنابراین الگوریتم ٤,١ در خط ٤ $l(e_b)$ را به اندازه $l^* - l(e_b)$ افزایش می‌دهد و این افزایش هزینه‌ای برابر با $(l^* - l(e_b)) \times c^+(e_b)$ دارد که الگوریتم ٤,١ در خط ٣ این هزینه را در متغیر Z ذخیره می‌کند. توجه کنید که اگر $l^* > l(e_b)$ روی ندهد الگوریتم ٤,١ در خط ٣ متغیر Z را به صفر مقداردهی می‌کند. الگوریتم ٤,١ در خط ٥ متغیر C را به صفر مقداردهی می‌کند که این متغیر نشان می‌دهد چقدر از کوله پر شده است و در نتیجه $cap - C$ ظرفیت خالی کوله است. با استقراء ثابت می‌کنیم که تا برقرار شدن شرایط لم ۱ الگوریتم ٤,١ یکی از یال‌های مسیر P_c^{k+1} را انتخاب می‌کند و طول آن یال و طول یال e_b (در صورت نیاز) را افزایش می‌دهد به طوری که این افزایش کمترین هزینه را دارد.

با توجه به این که برای هر $j = 1,\ldots,2k-1$ داریم $c^+(e_{A_j^1}) + c^+(e_b) \times A_j^2 \le c^+(e_{A_{j+1}^1}) + c^+(e_b) \times A_{j+1}^2$ یال‌های مسیر P_c^{k+1} با شروع از اندیس A_1^1 تا A_{2k}^1 بررسی می‌کنیم همانگونه که الگوریتم یال‌ها را بررسی می‌کند. ابتدا یالی را که دارای اندیس $t = A_1^1$ است بررسی می‌کنیم؛ یعنی وقتی که در الگوریم $i = 1$ است. در اولین تکرار حلقه داریم $A_1^2 = 0$. در اولین تکرار حلقه الگوریتم یالی را بررسی می‌کند که دارای کمترین هزینه افزایش واحد طول است. بنابراین در این مرحله $l^+(e_t)$ حداکثر مقداری است که می‌توانیم با آن کوله را پر کنیم. اگر بخواهیم کوله را با $l^+(e_t)$ پر کنیم باید دو شرط را بررسی کنیم. شرط اول این است که آیا ظرفیت خالی کوله به اندازه $l^+(e_t)$ است و شرط دوم این است که با اضافه کردن $l^+(e_t)$ به طول یال e_t، طول جدید یال e_t از طول یال e_b بیشتر نشود. در خط ۹ الگوریتم ٤,١، این شرایط را بررسی می‌کنیم. اگر شرط خط ۹ الگوریتم ٤,١ روی دهد در خط ۱۰ مقدار $l^+(e_t)$ را به کوله و در خط ۱۱ هزینه این افزایش طول را به متغیر Z اضافه می‌کنیم. در خط ۱۲ $l^+(e_t)$ را به صفر مقداردهی می‌کنیم که نشان می‌دهد الگوریتم در ادامه دیگر نمی‌تواند طول یال e_t را افزایش دهد.

فرض کنید شرط خط ۹ روی ندهد. بنابراین یا ظرفیت خالی کوله کمتر از $l^+(e_t)$ است یا با افزایش طول یال e_t به اندازه $l^+(e_t)$ طول جدید یال e_t از طول یال e_b بیشتر می‌شود. در نتیجه حداکثر افزایش طول e_t برابر با مقدار $cap - C$ یا $l(e_b) - l(e_t)$ است. فرض کنید که $l(e_b) - l(e_t) < cap - C$؛ به عبارت دیگر شرط خط ۱۳ روی دهد. در نتیجه طول یال e_t را تا جایی افزایش می‌دهیم که حداکثر برابر با طول یال e_b شود؛ به عبارت دیگر طول یال e_t را به اندازه $l(e_b) - l(e_t)$ افزایش می‌دهیم. مقدار $l(e_b) - l(e_t)$ را در خط ١٤ به ظرفیت کوله و هزینه این افزایش را در خط ۱۵ به متغیر Z اضافه می‌کنیم. در خط ١٦ مقدار $l(e_b) - l(e_t)$ را از $l^+(e_t)$ کسر می‌کنیم که نشان می‌دهد الگوریتم در ادامه می‌تواند طول یال e_t را حداکثر به اندازه $l^+(e_t) - (l(e_b) - l(e_t)$ افزایش دهد. فرض کنید که $l(e_b) - l(e_t) \ge cap - C$؛ به عبارت دیگر شرط خط ۱۷ روی دهد. در نتیجه طول یال e_t را تا جایی افزایش می‌دهیم که ظرفیت کوله پر شود؛ به عبارت دیگر یال e_t را به اندازه $cap - C$ افزایش می‌دهیم. با پر شدن کوله الگوریتم پایان می‌یابد. در خط ۱۸ الگوریتم، هزینه این افزایش را به متغیر Z اضافه می‌کنیم و سپس الگوریتم این مقدار را برگشت می‌دهد و در نتیجه الگوریتم پایان می‌یابد. با این بررسی پایه استقراء ثابت می‌شود. فرض کنید که الگوریتم در حلقه خود تا تکرار i-ام به کوله مقداری را اضافه می‌کند که کمترین هزینه را دارد. در این مرحله تکرار $i+1$-ام حلقه الگوریتم را بررسی می‌کنیم. اگر $cap - C$ برابر با صفر باشد؛ یا به عبارت دیگر ظرفیت کوله پر شده است، آنگاه به آسانی دیده می‌شود که در ادامه تا انتهای اجرای الگوریتم مقدار صفر را به ظرفیت کوله و متغیر Z اضافه می‌کند. در ادامه فرض کنید $cap - C$ مثبت باشد. فرض کنید $t = A_{i+1}^1$. در اینجا A_{i+1}^2 می‌تواند صفر یا یک باشد. اگر A_{i+1}^2 صفر باشد اثبات مانند قسمت پایه استقراء

خواهد بود. فرض کنید که A^2_{i+1} یک باشد. در این حالت در یکی از تکرارهای قبلی الگوریتم طول یال e_t مقداری افزایش یافته است و $l^+(e_t)$ مقداری را نشان می‌دهد که هنوز به طول یال e_t اضافه نشده است و می‌تواند به ظرفیت کوله اضافه شود. در اینجا؛ یعنی وقتی که A^2_{i+1} یک است، طول جدید دو یال e_t و e_b برابر هستند و هر مقدار که طول یال e_t را افزایش دهیم به همان اندازه باید طول یال e_b را نیز افزایش دهیم. برای این که طول یال e_t را افزایش دهیم باید مواظب باشیم که این افزایش حداکثر به اندازه ظرفیت خالی کوله و $l^+(e_b)$ باشد. در خط ۲۰ الگوریتم این شرایط را بررسی می‌کنیم. اگر شرط خط ۲۰ روی دهد در خط ۲۱ مقدار $l^+(e_t)$ را به کوله و در خط ۲۲ هزینه افزایش طول یال‌های e_t و e_b را به متغیر Z اضافه می‌کنیم. در ادامه دیگر یال e_t را بررسی نخواهیم کرد. در خط ۲۳ $l^+(e_t)$ را از $l^+(e_b)$ کسر می‌کنیم چرا که این مقدار را به طول یال e_b اضافه کرده‌ایم. فرض کنید که شرط خط ۲۰ روی ندهد؛ به عبارت دیگر مقدار $l^+(e_t)$ بیشتر از ظرفیت خالی کوله یا مقدار $l^+(e_b)$ است. بنابراین حداکثر افزایش طول e_t برابر با مقدار $cap - C$ یا $l^+(e_b)$ است.

در ادامه دو حالت را وابسته به این که $cap - C \le l^+(e_b)$ یا $cap - C > l^+(e_b)$ بررسی می‌کنیم. فرض کنید که $cap - C > l^+(e_b)$ روی دهد. این شرط در خط ۲٤ الگوریتم بررسی می‌شود. در این حالت می‌توانیم طول یال e_t را به اندازه $l^+(e_b)$ افزایش دهیم. در خط ۲۵ مقدار $l^+(e_b)$ را به کوله و در خط ۲٦ هزینه افزایش طول یال‌های e_t و e_b را به متغیر Z اضافه می‌کنیم. در ادامه دیگر یال e_t را بررسی نخواهیم کرد. در خط ۲۷ $l^+(e_b)$ را به صفر مقداردهی می‌کنیم چرا که این مقدار را به طول یال e_b اضافه کرده‌ایم. فرض کنید که $cap - C \le l^+(e_b)$. در این حالت شرط خط ۲۸ الگوریتم روی می‌دهد. با پر کردن کوله به اندازه $cap - C$ با یال e_t کوله پر می‌شود و شرایط لم ۱ برقرار می‌شود. در این حالت الگوریتم در خط ۲۹ هزینه این افزایش یال را به متغیر Z اضافه می‌کند و با برگشت دادن این مقدار به اجرای الگوریتم پایان می‌دهد. بعد از پایان یافتن حلقه الگوریتم اگر $cap = C$ روی دهد هزینه مسئله Z است و با برگشت دادن این مقدار اجرای الگوریتم پایان می‌یابد؛ در غیر این صورت مسئله جواب شدنی ندارد. این اثبات لم را کامل می‌کند. به طور مشابه می‌توانیم لم زیر را ثابت کنیم.

لم ۳. فرض کنید که $b \in \{2,\ldots,n-1\}$، $l^* = \max\{l(e_1),\ldots,l(e_n)\}$، $cap = |\, l^{k+1}_{cc} - l^{k+1}_{c} \,|$ و $T = (T_1,\ldots,T_n)$ به‌طوری‌که $T_i = \langle l(e_i),\, l^+(e_i),\, c^+(e_i)\rangle$ برای هر $i = 1,\ldots,n$. اگر $k = n - b$ و $A = (A_1,\ldots,A_{2k})$ به طوری که $A_j = \langle A^1_j, A^2_j\rangle$، $A^1_j \in \{k+1,\ldots,n\}$ و $A^2_j \in \{0,1\}$ برای هر $j = 1,\ldots,2k$ و $c^+(e_{A^1_j}) + c^+(e_b)\times A^2_j \le c^+(e_{A^1_{j+1}}) + c^+(e_b)\times A^2_{j+1}$ برای هر $j = 1,\ldots,2k-1$، آنگاه الگوریتم ٤,۱ با این ورودی‌ها زیرمسئله Inc^i_{cc} را برای دور C و یال e_b حل می‌کند.

۶. یک الگوریتم برای حل مسئله

در این بخش یک الگوریتم کارآ (الگوریتم ٥,۱) برای حل مسئله مکان‌یابی معکوس ۱-مرکز روی دور C پیشنهاد می‌کنیم. الگوریتم ٥,۱ برای هر یال دور C با توجه زیرمسئله مربوطه را با استفاده از الگوریتم ٤,۱ حل می‌کند. بعد از بررسی همه یال‌ها زیرمسئله‌ای که کمترین هزینه را دارد جواب مسئله را به ما می‌دهد.

توجه کنید که در بخش قبل الگوریتم ٤,۱ برای حل زیرمسئله مکان‌یابی معکوس ۱-مرکز روی دور C و یال e_k ارائه شد. همانگونه که در الگوریتم ٥,۱ دیده می‌شود برای $i = 2$ تا $i = n-1$ الگوریتم زیرمسئله مکان‌یابی معکوس ۱-مرکز روی دور C و یال e_i را حل می‌کند و مقدار هزینه زیرمسئله را در متغیر Z_i ذخیره می‌کند و بعد از بررسی همه $i \in \{2,\ldots,n-1\}$ کوچکترین مقدار Z_i را به عنوان جواب برمی‌گرداند. جواب بهینه Inc^c_i و Inc^c_i را با استفاده از الگوریتم ٤,۱ محاسبه می‌کنیم.

Algorithm ۵،۱: I1CL(C)

Input: A cycle graph $C=(V,E)$ with $V=\{v_1,\dots v_n\}$ and $E=\{e_1,\dots e_n\}$.

Output: An optimal solution of the inverse ۱-center location problem on C

$l^*=\max\{l(e_b): e\in E\}$;

$l_c=l(e_b)$;

$l_{cc}=\sum_{i=3}^{n} l(e_b)$;

$b=2$;

$B_c=(\langle 1,0\rangle)$;

$B'_c=(\langle 1,1\rangle)$;

$A_c=(\langle 1,0\rangle,\langle 1,1\rangle)$;

such that $\alpha_1,\dots,\alpha_{n-2}\in\{3,\dots,n\}$ and $c^+(e_{\alpha_1})\le\cdots\le c^+(e_{\alpha_{n-2}})$. $B_{cc}=(\langle\alpha_1,0\rangle,\dots,\langle\alpha_{n-2},0\rangle)$;

$B'_{cc}=(\langle\alpha_1,1\rangle,\dots,\langle\alpha_{n-2},1\rangle)$;

such that $\beta_1,\dots,\beta_{2(n-2)}\in\{3,\dots,n\}$, $\gamma_1,\dots,\gamma_{2(n-2)}\in\{0,1\}$ and $A_{cc}=(\langle\beta_1,\gamma_1\rangle,\dots,\langle\beta_{2(n-2)},\gamma_{2(n-2)}\rangle)$; $c^+(e_{\beta_1})+\gamma_1\times c^+(e_b)\le\cdots\le c^+\left(e_{\beta_{2(n-2)}}\right)+\gamma_{2(n-2)}\times c^+(e_b)$.

$T_c=(\langle l(e_1),l^+(e_1),c^+(e_1)\rangle)$;

$T_{cc}=(\langle l(e_2),l^+(e_2),c^+(e_2)\rangle,\dots,\langle l(e_n),l^+(e_n),c^+(e_n)\rangle)$;

for $i=1$ **to** $n-1$ **do** {

if $l_c=l_{cc}$ **then**

if $l^*>l(e_i)$ **then**

$Z_i=(l^*-l(e_i))\times c^+(e_i)$;

Else

return 0;

else if $l_c<l_{cc}$ **then**

$Z_i=FK(b,b-1,T_c,A_c,|l_c-l_{cc}|,l^*)$;

Else

$Z_i=FK(b,n-b,T_{cc},A_{cc},|l_c-l_{cc}|,l^*)$;

$l_c=l_c+l(e_i)$;

$l_{cc}=l_{cc}-l(e_{i+1})$;

$b=b+1$;

Add $\langle i,0\rangle$ to B_c and $\langle i,1\rangle$ to B'_c;

Merge B_c and B'_c to form A_c;

Remove $\langle i+1,0\rangle$ from B_{cc} and $\langle i+1,1\rangle$ to B'_{cc};

Merge B_{cc} and B'_{cc} to form A_{cc}; }

return $\min\{Z_i: 2\le i\le n-1\}$;

دیده می‌شود که اگر ورودی‌های الگوریتم ٤،١ آماده باشند زمان اجرای الگوریتم ٤،١ $O(n)$ است.

قضیه ٤. الگوریتم ٥،١ مسئله مکان‌یابی معکوس ١-مرکز روی دور C را در زمان $O(n^2)$ حل می‌کند.

برهان. فرض کنید که $M=\max\{l(e): e\in E(C)\}$ و $T=(T_1,\dots,T_n)$ به طوری که $T_i=\langle l(e_i)\ l^+(e_i)\ c^+(e_i)\rangle$ برای هر $i=1,\dots,n$. نشان می‌دهیم برای $i=2$ الگوریتم زیرمسئله مکان‌یابی معکوس ١-مرکز روی دور C و یال e_2 را حل می‌کند. یال‌های دو مسیر P_c^2 و P_{cc}^2 را در نظر بگیرید. البته P_c^2 فقط یک یال دارد. فرض کنید که $b=2$. قبل از حلقه تکرار الگوریتم ٥،١، مقادیر $c^+(e)$ را برای یال‌های مسیر P_{cc}^2 به صورت غیرنزولی مرتب می‌کنیم و اندیس یال‌ها را به ترتیب در دنباله $B_{cc}=(\langle\alpha_1,0\rangle,\dots,\langle\alpha_{n-2},0\rangle)$ قرار می‌دهیم. (مقدار صفر در مولفه دوم نشان می‌دهد که مولفه اول مقدار $c^+(e)$ برای یک یال e مسیر P_{cc}^2 است.) این کار در زمان $O(n\log n)$

قابل انجام است. در واقع داریم $\alpha_1,\ldots,\alpha_{n-2} \in \{3,\ldots,n\}$ به طوری که $c^+(e_{\alpha_1}) \le \cdots \le c^+(e_{\alpha_{n-2}})$. واضح است که $c^+(e_{\alpha_1}) + c^+(e_b) \le \cdots \le c^+(e_{\alpha_{n-2}}) + c^+(e_b)$. فرض کنید که $B_{cc} = (\langle \alpha_1,0\rangle,\ldots,\langle \alpha_{n-2},0\rangle)$. (مقدار یک در مولفه دوم نشان می‌دهد که مولفه اول مقدار $c^+(e) + c^+(e_b)$ برای یک یال e مسیر P_{cc}^2 است.) این کار در زمان $O(n)$ قابل انجام است. سپس دو دنباله B_{cc} و B'_{cc} را با توجه به مولفه اول آنها به ترتیب غیرنزولی با هم ادغام می‌کنیم و در دنباله جدید A_{cc} قرار می‌دهیم. این کار در زمان $O(n)$ قابل انجام است. در واقع داریم $A_{cc} = (\langle \alpha'_1, \beta'_1\rangle,\ldots,\langle \alpha'_{2(n-2)}, \beta'_{2(n-2)}\rangle)$ به طوری که برای هر $j = 1,\ldots,2(n-2)-1$ داریم $c^+(e_{\alpha'_j}) + c^+(e_b) \times \beta'_j \le c^+(e_{\alpha'_{j+1}}) + c^+(e_b) \times \beta'_{j+1}$ و برای هر $j = 1,\ldots,2(n-2)$ داریم $\beta'_j \in \{0,1\}$. همچنین برای یال‌های مسیر P_c^2 سه دنباله $B_c = (\langle 1,0\rangle)$، $B'_c = (\langle 1,1\rangle)$ و $A_c = (\langle 1,0\rangle,\langle 1,1\rangle)$ را در نظر بگیرید. این کار در زمان $O(1)$ قابل انجام است. بنابراین زمان اجرای این دستورات $O(n\log n)$ است. تمام دنباله‌های بالا را با استفاده از لیست پیوندی پیاده‌سازی می‌کنیم.

اگر $l_{cc} < l_c$ آنگاه با توجه به لم ۳ برای محاسبه جواب بهینه Inc_{cc}^2 الگوریتم ٤,١ را با ورودی‌های $b = 2$، $k = n-2$، T، $A = A_{cc}$، $cap = |l_c - l_{cc}|$ و $l^* = M$ فراخوانی می‌کنیم و در غیر این صورت (اگر $l_{cc} < l_c$) آنگاه با توجه به لم ۲ برای محاسبه جواب بهینه Inc_c^2 الگوریتم ٤,١ را با ورودی‌های $b = 2$، $k = 1$، T، $A = A_c$، $cap = |l_c - l_c|$ و $l^* = M$ فراخوانی می‌کنیم.

سپس متغییر i با شروع از ۲ تا $n-2$ مقداردهی می‌شود. در این حالت داریم $b = i+1$. در B_{cc} و B'_{cc} عنصر $\langle b,0\rangle$ را جستجو و سپس حذف می‌کنیم و عنصر $\langle i,0\rangle$ را به B_c و B'_c اضافه می‌کنیم. توجه کنید مرتب بودن لیست‌ها باید حفظ شود. با توجه به مرتب بودن این دنباله‌ها، این کار در زمان $O(n)$ قابل انجام است. سپس عناصر دو دنباله B_{cc} و B'_{cc} را به ترتیب غیرنزولی ادغام می‌کنیم و در دنباله A_{cc} قرار می‌دهیم. به طور مشابه عناصر دو دنباله B_c و B'_c را به ترتیب غیرنزولی ادغام می‌کنیم و در دنباله A_c قرار می‌دهیم. این کار در زمان $O(n)$ قابل انجام است.

اگر $l_{cc} < l_c$ آنگاه با توجه به لم ۳ برای محاسبه جواب بهینه Inc_{cc}^2 الگوریتم ٤,١ را با ورودی‌های b، $k = n-b$، T، $A = A_{cc}$، $cap = |l_c - l_c|$ و $l^* = M$ فراخوانی می‌کنیم و در غیر این صورت (اگر $l_{cc} < l_c$) آنگاه با توجه به لم ۲ برای محاسبه جواب بهینه Inc_c^3 الگوریتم ٤,١ را با ورودی‌های b، $k = b$، T، $A = A_c$، $cap = |l_c - l_c|$ و $l^* = M$ فراخوانی می‌کنیم. این اثبات قضیه را کامل می‌کند.

۷. نتیجه‌گیری و کار بیشتر

در این مقاله به بررسی مسئله معکوس ۱-مرکز مطلق روی یک شبکه دور پرداختیم. در مسئله معکوس مکان‌یابی ۱-مرکز مطلق روی یک گراف دور یک راس در گراف مشخص شده است و هدف این است تا طول یال‌های گراف دور را افزایش دهیم به طوری که با انجام کمترین هزینه راس مشخص شده با طول یال‌های جدید یک ۱-مرکز مطلق گراف دور باشد. در این مقاله یک مدل برنامه‌ریزی به صورت بهینه‌سازی برای مسئله ارائه دادیم و برای حل این مدل برنامه‌ریزی بهینه‌سازی تعدادی مدل برنامه‌ریزی خطی با تعداد قیود کمتر پیشنهاد دادیم. برای حل مدل‌های برنامه‌ریزی خطی با قیود کمتر یک الگوریتم کوله‌پشتی کسری طراحی کردیم. در نهایت با استفاده از الگوریتم کوله‌پشتی کسری پیشنهادی یک الگوریتم ترکیبیاتی از مرتبه مربعی بر حسب تعداد راس‌های دور برای حل مسئله پیشنهاد دادیم. در ادامه پیشنهادهایی برای کار بیشتر ارائه می شود. کاهش زمان اجرای الگوریتم و طراحی یک الگوریتم برای مسئله با مرتبه زمانی کمتر از مربعی پیشنهاد می‌شود. همچنین بررسی این مسئله مطرح شده در مقاله برای گراف‌هایی که خواصی شبیه گراف دور (گراف‌های تک‌دور) دارند پیشنهاد می‌شوند. همچنین

آیـــا می‌توان روش حل این مقاله را برای حالتی از مســـئله معکوس ۱-مرکز روی یک شـــبکه دور که مجاز به کاهش طول یال‌ها هســـتیم نیز به کار ببریم و بتوان زمان اجرای الگوریتم نگوین [۱۵] را کاهش داد.

مراجع

[1] Daskin, M. (1997). Network and discrete location: Modeles, algorithms and applications. Wiley, New York.

[2] Drezner, Z., Hamacher, H. W. (Eds.). (2001). Facility location: applications and theory. Springer Science and Business Media.

[3] Nazari, M., Fathali, J., Nazari, M., Varedi Koulaei, S.M. (2018). Inverse of Backup 2-Median Problems with Variable Edge Lengths and Vertex Weight on Trees and Variable Coordinates on the Plane. Production and Operations Management. 9(2), 115-137.

[4] Galavii, M. (2010). The inverse 1-median problem on a tree and on a path, Electron. Notes Discrete Math. 36, 1241-1248.

[5] Guan, X., & Zhang, B. (2011). Inverse 1-median problem on trees under weighted Hamming distance, J. Glob. Opt. 54, 75-82.

[6] Pham, V.H., & Nguyen, K.T. (2019). Inverse 1-median problem on trees under mixed rectilinear and Chebyshev norms, Theoret. Comput. Sci. 795, 119-127.

[7] Burkard, R.E., Pleschiutschnig, C., & Zhang, J.Z. (2008). The inverse 1-median problem on a cycle, Discrete Optim. 5, 242-253.

[8] Nguyen, K.T. (2016). Inverse 1-median problem on block graphs with variable vertex weights, J. Optim. Theory Appl. 168, 944-957.

[9] Nazari, M., & Fathali, J. (2023). Inverse and reverse 2-facility location problems with equality measures on a network. Iranian Journal of Mathematical Sciences and Informatics. 18(1), 211-225.

[10] Nguyen-Thu, H., Nguyen, K. T., & Nguyen T.T. (2024). The max-sum inverse median location problem on trees with budget constraint, Applied Mathematics and Computation. 128296

[11] Cai, M.C., Yang, X.G., & Zhang, J.Z. (1999). The complexity analysis of the inverse center location problem, J. Glob. Optim. 15, 213-218.

[12] Alizadeh, B., Burkard, R.E., & Pferschy, U. (2009). Inverse 1-center location problems with edge length augmentation on trees, Computing. 86, 331-343.

[13] Alizadeh, B., & Burkard, R.E. (2011). Combinatorial algorithms for inverse absolute and vertex 1-center location problems on trees, Networks. 58, 190-200.

[14] Hasanzadeh, M., Alizadeh, B., & Baroughi, F. (2024). Optimal algorithms for inverse obnoxious center location problems under the weighted Chebyshev and Hamming cost norms on networks, Optimization 73(3), 545-574.

[15] Nguyen, K. T. (2019). The inverse 1-center problem on cycles with variable edge lengths, Central European Journal of Operations Research. 27, 263-274.