Review of : Approximating Minimum Bounded Degree Spanning Tree (MBDST)

دانشگاه صنعتی شریف

دانشکده کامپیوتر

محمد تنهایی ۸۷۲۰۲۴۴۶

tanhaei@ce.sharif.edu

چکیده مقاله :

در مسئله MBDST ما یک گراف بدون جهت داریم و می خواهیم با توجه به محدودیت تعداد یال های متصل شده به هر راس که برای هر راس برابر با B_v است ، یک درخت پوشای می نیمم بدست آوریم .

اگر OPT را حل بهینه مسئله بالا در نظر بگیریم :

الگوریتم ما یک گراف G را با محدودیت B_v روی هر راس گرفته و درخت T را به عنوان جواب برمی گرداند که دارای خاصیت های زیر است :

$d_T($

و اینکه مجموع هزینه رئوس T حداکثر برابر با OPT است.

حالت کلی تر مسئله وجود یک کف برای هر راس است که برای آن نیز می توان با اندکی گسترش روی الگوریتم قبلی یک الگوریتم ارائه داد.

۱ دیباچه

مسئله MINIMUM BOUNDED DEGREE SPANNING TREE به صورت زیر تعریف می شود :

یک گراف ساده و بدون جهت G=(V,E) داریم که وزن هر یال آن برابر c است ، برای هر یک از رئوس گراف یک درجه بیشینه به نام B_v داریم ، می خواهیم یک درخت پوشای می نیمم بدست آوریم که در شرایط بالا صدق کند ، یعنی درجه هر راس بزرگتر از B_v نباشد و درخت پوشا و می نیمم باشد .

به دست آوردن چنین درختی از درجه NP است ، به عنوان نمونه در نظر بگیرید که همه Bv ها برابر ۲ باشند ، در این حالت مسئله به مسئله مسیر هامیلتونی تبدیل می شود بنابراین به دنبال الگوریتم هایی می گردیم که به طور تقریبی بتوانند مسئله را حل کنند.

در این مقاله الگوریتمی ارائه می شود که می تواند در زمان چند جمله ای مسئله بالا را در محدوده $d_T(v) \leq B_v + 1$ حل نماید .

ابتدا سعی می شود که تکنیک های تبدیل مسئله به یک مسئله برنامه ریزی خطی بررسی شده و سپس روش هایی برای آسان کردن حل آن به کار برده می شود.

کل الگوریتم حول بازی با برنامه ریزی خطی و ساده کردن حل آن است .

در این الگوریتم ابتدا روش های حل تکراری MST بررسی می شود ، سپس یک الگوریتم B_v+2 ارائه می شود و در آخر ، کارِ اصلی مقاله که الگوریتم تقریبی B_v+1 است ارائه می شود ، با گسترشی ساده می توان الگوریتمی برای حل $B_v\pm1$ تغییر داد.

۱.۱ تکنیک های استفاده شده

در این مقاله از روش های چند سطحی برای طراحی الگوریتم تقریبی استفاده شده است . در روش های چند سطحی ابتدا مسئله را به صورت یک برنامه ریزی خطی تبدیل می کنند و سپس ساده سازی هایی روی آن انجام می دهند . به طور خیلی خلاصه برنامه ریزی خطی برای MBDST را معرفی می کنیم و سپس سعی می کنیم که آنرا گسترش دهیم تا بتوانیم یک الگوریتم کارا به دست آوریم :

$$
\begin{aligned}
& minmize\ c(x) = \sum_{e \in E} c_e x_e \\
& suject\ to\ x(E(V)) = |V| - 1 \\
& \qquad x(E(S)) \leq |S| - 1 \qquad \forall\, S \subset V \\
& \qquad x(\delta(v)) \leq B_v \qquad \forall\, v \in V \\
& \qquad x_e \geq 0 \qquad \forall\, e \in E
\end{aligned}
$$

در این الگوریتم $E(S)$ عبارت است از یال هایی که دو راس آن در داخل مجموعه S قرار دارد . $\delta(S)$ مجموعه یالهایی است که دقیقاً یک راس آن ها در مجموعه S قرار دارد و x(U) هم عملگری است که تعداد اعضای مجموعه را می شمارد.

درستی این برنامه ریزی خطی واضح است . اما نکته قابل تامل تعداد نمایی زیر مجموعه های V است . برای این منظور ، باید از نکته اساسی که Geomans بر اساس آن الگوریتم خود را بنا نهاده است غافل نشویم و آن این است که برای حل این مسئله خطی تنها کافی است که شرط های تساوی که به وسیله laminar family تعریف می شود را ارضا نمود و جواب به دست آمده یک جواب بهینه اولیه خواهد بود .

توضیح اینکه laminar family عبارت است از یک خانواده از زیر مجموعه های یک مجموعه ، که هیچ ۲ تایی آنها با هم اشتراک ندارد.

الگوریتم اصلی به صورت زیر کار می کند : ابتدا مسئله را به وسیله برنامه ریزی خطی (مانند بالا) حل می کند و سپس رئوسی که بالاترین ارزش دارند را انتخاب کرده و به جواب اضافه می کند ، اگر نتوانست چنین راسی را پیدا کند مجدداً مسئله خطی را حل می کند تا یک جواب به دست آورد و الگوریتم به همین شیوه ادامه پیدا می کند .

در ادامه گزارش مقاله سعی می کنیم که ابتدا روش polyhedron برای حل مسئله MST را شرح دهیم ، سپس الگوریتم A+2 را به طور خلاصه بحث می کنیم و در نهایت الگوریتم A+1 و A±1 که کار اصلی مقاله هستند را توضیح می دهیم .

برای خلاصه شدن گزارش سعی می کنیم که بخش های مهم اثبات ها را ذکر کرده و از گفتن جزئیات ریز خودداری کنیم . برای فهم کامل مقاله مراجعه به مرجع اصلی مقاله امری ضروری است.

۲ الگوریتم polyhedron برای MST

همان طور که در تکنیک ها توضیح داده شد ابتدا یک برنامه ریزی خطی را طراحی می کنیم و سپس یک روش تکراری برای حل مسئله به کار می گیریم.

برنامه ریزی خطی ما که عنوان LP-MST(G) را برای آن انتخاب کرده ایم به صورت زیر است :

$$\begin{aligned} &minmize\ c(x) = \sum_{e\in E} c_e x_e \\ &suject\ to\ x\big(E(V)\big) = |V| - 1 \\ &\qquad x\big(E(S)\big) \le |S| - 1 \qquad \forall\, S \subset V \\ &\qquad x_e \ge 0 \qquad \forall\, e \in E \end{aligned}$$

این برنامه ریزی خطی می تواند جواب بهینه اولیه را برای ما به دست آورد حال یک الگوریتم تکراری طراحی می کنیم که از جواب x* که این برنامه ریزی خطی داده است استفاده کند .

F=ϕ.

While F is not a spanning tree

1. Solve LP to obtain an extreme point x*
2. Remove all edges s.t. $x^*_e = 0$
3. If there exists a leaf vertex v, then include the edge incident at v in F and remove v from G.

خلاصه اثبات درستی جواب الگوریتم : در اثبات درستی از روش استقرایی استفاده می کنیم و ثابت می کنیم که درخت F یک درخت می نیمم پوشا است.

اگر در مرحله 2b الگوریتم یال e را انتخاب کند ، ما حتماً داریم $x_e^* = 1$ و درجه راس های آن نیز بالاتر یا مساوی با یک خواهد بود . این راس به درخت ما اضافه می شود .راس متناظر با یال e را v می نامیم . اگر درخت حاصل از حل G\{v} را T' بنامیم در این صورت : T= T'∪{e} فرض کنیم که الگوریتم برای G' درست کار می کند (فرض استقرا) در این صورت یک x_{res}^* را بر می گرداند . حال داریم $c(F') \le c.\, x_{res}^*$ از طرفی داریم :

$c(F) = c(F') + c_e$ و $c(F') \le c.\, x_{res}^*$ که نتیجه می دهد : $c(F) \le c.\, x_{res}^* + c_e = c.x^*$ و این یعنی اینکه جواب ما همچنان کمتر از میزان OPT است زیرا $x_e^* = 1$ است .

مرحله بعدی که در این الگوریتم **داریم اثبات درست بودن الگوریتم و خاتمه** آن است ، یعنی اینکه بعد از هر بار حل مسئله بهینه یک برگ وجود دارد :

ابتدا باید ثابت کنیم حداکثر تعداد شرط های مساوی در الگوریتم LP حداکثر برابر با n-1 است . اثبات از طریق تکنیک uncrossing انجام می شود (این اثبات در Jain وجود دارد) . اگر F را برابر خانواده همه زیر مجموعه های S که شرط مساوی دارند در نظر بگیریم و L را برابر ماکزیمال laminar family رئوس S در نظر بگیریم باید ثابت کنیم که span(L) = span(F) ، این اثبات سر راست است می توانیم دو زیر مجموعه بگیریم که در F هستند ولی در L نیستند و با برهان خلف ثابت می کنیم که در این صورت L ماکزیمال نخواهد بود .(تکنیک uncrossing)

نکته : span(T) برابر است با برداری که حضور یا عدم حضور یک یال را در داخل مجموعه T نشان می دهد. حضور با ۱ و عدم حضور با ۰ مشخص می شود و اندازه بردار برابر با S است.

بعد از اثبات دو لم بالا می توانیم درستی الگوریتم را با برهان خلف اثبات کنیم (در همه جای این مقاله برای اثبات درستی از برهان خلف استفاده خواهیم کرد) . برای این منظور فرض کنیم که در مرحله 2b راس با درجه ۱ (برگ) وجود نداشته باشد بنابراین کلیه رئوس حداقل از درجه ۲ هستند ، مجموع کلیه رئوس برابر با 2|V| است ، از طرفی می دانیم که |E*| از نصف مجموع رئوس بزرگتر است پس |E*|>|V| . چون که x* یک جواب بهینه اولیه است و ماکزیمال laminar family برابر با |L| پس : |E*| = |L| و از لم های قبلی داریم که span(L) = span(F) پس در نتیجه حداکثر تعداد مجموعه های خانواده laminar برابر با |V|-1 است . یعنی |L|≤ |V|-1 پس در نتیجه |E*| ≤ |V| -1 که تناقض است.

۳ الگوریتم تقریبی A+2

این بخش را به طور خیلی خلاصه توضیح می دهیم ، این الگوریتم از کارهای گذشته الهام گرفته شده است . یک روش LP برای حل MBDST ارائه می کنیم و سپس یک روش تکراری که در مقاله عنوان شده را بحث می کنیم ، با توجه به اینکه در الگوریتم های A+1 اثبات های کلی تری ارائه می شود از اثبات ها صرف نظر می کنیم .

LP-MBDST(G,B,W)

$$\begin{aligned} &minmize\ c(x) = \sum_{e\in E} c_e x_e \\ &suject\ to\ x(E(V)) = |V| - 1 \\ &\qquad x(E(S)) \le |S| - 1 \qquad \forall\, S \subset V \\ &\qquad x(\delta(v)) \le B_v \qquad \forall\, v \in W \\ &\qquad x_e \ge 0 \qquad \forall\, e \in E \end{aligned}$$

در اینجا W مجموع رئوسی است که می خواهیم محدودیت $x(\delta(v)) \le B_v$ به آن اعمال شود . الگوریتم تکراری به صورت زیر است :

Initialize F=ϕ.

While F is not a spanning tree

1. Solve LP to obtain extreme point x*.
2. Remove all edges e s.t. $x^*_e = 0$.
3. If there is a leaf vertex v with edge {u,v}, then
    1. Include {u,v} in F.
    2. Decrease B_u by 1. Delete v from G. Delete v from W
4. If there is a vertex v ∈ W such that $\deg_E(v) \le 3$, then remove the degree constraint of v. i.e. Delete v from W

خلاصه اثبات درستی جواب الگوریتم :

اثبات درستی جواب شبیه به اثباتی است که در الگوریتم قبلی انجام دادیم . در مرحله ۳ ما هر بار یک برگ را بر می داریم و درجه B راس متصل به آنرا یکی کم می کنیم . و در مرحله ۴ محدودیت راس با درجه ۳ را بر می داریم ، بدترین حالت این است که $b_v = 1$ باشد که در این حالت نیز شرط $d_v(T) \le B_v+2$ برقرار است . چون که هر بار یک برگ انتخاب می شود جواب حاصل تشکیل یک درخت خواهد داد.

و اما خلاصه اثبات درستی الگوریتم و خاتمه :

ابتدا با تکنیک uncrossing ثابت می کنیم که یک T⊆W و نیز یک خانواده laminar مانند L وجود دارد که x* جواب مسئله خطی زیر است :

$$\begin{cases} x^*(E(S)) = |S| - 1 & \forall\, S \in L \\ x^*(\delta(v)) = B_v & \forall\, v \in T \end{cases}$$

و |E*| = |L| + |T| اثبات حالت کلی این مسئله را در بخش بعدی ارائه می کنیم.

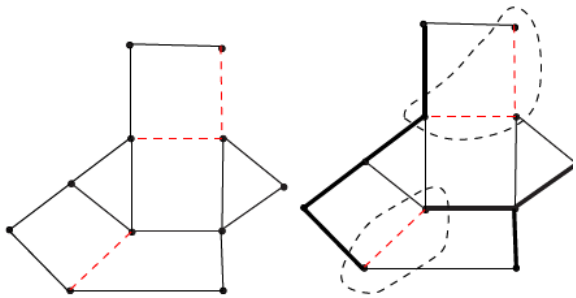
حال ثابت می کنیم که در هر مرحله یا یک راس با درجه ۱ وجود دارد و یا یکی از رئوس متعلق به w دارای درجه کمتر یا مساوی با ۳ است: فرض کنیم که این طور نباشد در این صورت هر راس متعلق به w دارای درجه حداقل ۴ و سایر رئوس دارای درجه حداقل ۲ می باشند چون که تعداد یال های گراف بیشتر یا مساوی نصف مجموع رئوس است در نتیجه : |E*| ≥ (2(n - |W|) + 4|W|)/2 = n+ |W| از طرفی از پاراگراف قبل داشتیم |E*| = |L| + |T| و می دانیم که حداکثر اندازه یک خانواده laminar برابر n-1 است پس : |E*| = |L|+|T| ≤ n-1+|T| ≤ n-1+|W| و این تناقض است . پس فرض ما غلط بوده است.

۴ الگوریتم تقریبی A+1

در این بخش کار اصلی مقاله را توضیح خواهیم داد . در این الگوریتم به جای اینکه هر بار یک برگ انتخاب کنیم هر بار یک راس با درجه یک انتخاب می کنیم . این راهبرد انتخاب باعث می شود که هر بار به جای درخت یک جنگل ایجاد شود . ما این جنگل را F می نامیم .. چون که در این جا یک سری جنگل ایجاد می شود مسئله ما به مسئله MINIMUM BOUNDED-DEGREE CONNECTING TREE تبدیل می شود .

در این مسئله برای گراف G=(V,E) با محدودیت حداکثر درجه B_v برای هر راس و یک تابع هزینه c و با توجه به جنگل F ، می خواهیم یک یک جنگل H پیدا کنیم به طوری که F∪H تشکیل یک درخت پوشای می نیمم بدهد . باید توجه داشته باشیم در اینجا F مجموعه یالهایی است که حتماً باید در جواب درخت پوشا وجود داشته باشند .

الگوریتم ما ابتدا با F=0 شروع کرده و هر بار یال متصل به یک راس با درجه ۱ را انتخاب کرده و به صورت بازگشتی خودش را صدا می زند تا اینکه یک درخت پوشای می نیمم به دست آید.



(a) (b)

شکل a یک گراف با مجموعه F که به صورت نقطه چین مشخص شده اند را نمایش می دهد و در شکل b خطوط پر رنگ نمایش دهنده جنگل H و اجتماع آنها تشکیل یک درخت پوشای می نیمم می دهد .

برای حل این مسئله همه راس های F را که با یک یال به هم متصل شده اند را به صورت یک راس در نظر می گیریم ، در شکل b رئوسی که با خط چین به هم متصل شده اند چنین خاصیتی دارند ، اگر درخت پوشا را در این گراف جدید به دست آوریم در واقع H را محاسبه کرده ایم .

برای ارائه LP مسئله و نیز روش تکراری آن ابتدا چند تعریف را ارائه می دهیم : F(S) عبارت است از مجموعه یال هایی از F که ابتدا و انتهای آنها در مجموعه S قرار دارد . τ(F) را تعریف می کنیم مجموعه رئوسی که با F اشتراک ندارند.

LP-MBDST(G,B,W,F)

$$minmize\ c(x) = \sum_{e \in E} c_e x_e$$

$$suject\ to:$$

$$x(E(V)) = |V| - |F(V)| - 1$$

$$x(E(S)) \le |S| - |F(S)| - 1 \quad \forall\, S \subset \tau(F)$$

$$x(\delta(v)) \le B_v \quad \forall\, v \in W$$

$$x_e \ge 0 \quad \forall\, e \in E$$

و الگوریتم تکراری برای به دست آوردن MBDCT عبارت است از :

MBDCT Algorithm(G, B, W, F)

1. If F is a spanning tree return 0 else let $\hat{F} = 0$
2. Find a basic optimal solution x^* of LP-MBDCT(G,B,W,F) and remove every edge e with $x_e^* = 0$ from G. Let E^* be the support of x^*
3. If there exists an edge $e = \{u, v\}$ such that $x_e^* = 1$, then set $\hat{F} \leftarrow \{e\}$, $F \leftarrow F \cup \{e\}$ and $G \leftarrow G \backslash \{e\}$. Also set $B_u \leftarrow B_u - 1$ and $B_v \leftarrow B_v - 1$
4. If there exists a vertex $w \in W$ such that $degE^*(w) \le B_w + 1$, then update $W \leftarrow W \backslash \{w\}$.
5. Return $\hat{F} \cup$ MBDCT Algorithm(G, B, W, F).

ابتدا **ثابت می کنیم که الگوریتم جواب درستی** را بر می گرداند : به عبارت ریاضی تر الگوریتم تکراری فوق یک درخت H بر می گرداند که شرایط محدودیت درجه $d_H(v) \le B_v+1$ را برای همه رئوس برآورده می کند و هزینه آن نیز حداکثر برابر با OPT است :

اثبات : برای اثبات از روش استقرایی استفاده می کنیم. اگر H=0 باشد جواب درست است ، زیرا F حتماً یک درخت پوشای می نیمم بوده است . فرض کنیم که x^* جواب بهینه الگوریتم LP در دور اول باشد ، در مرحله ۳ ما یک یال e=(u,v) پیدا می کنیم که $X_e^* = 1$ دارد . فرض کنیم که $F' = F \cup \{e\}$ و $G' = G \backslash \{e\}$ و B' محدودیت های ویرایش شده در مرحله ۳ الگوریتم باشد . بر اساس استقرا فرض می کنیم که H' جواب G' در حل LP-MBDCT(G′,B′,W,F′) باشد ، و برای هر راس عضو W داشته باشیم $d_{H'}(v) \le B'_v+1$ ، ثابت می کنیم که با افزودن e همچنان الگوریتم جواب درستی را بر می گرداند:

در نظر بگیرید که $H = H' \cup \{e\}$ و در نهایت می دانیم که x^*ای که به وسیله حل G' به دست می آید را x_{res}^* بنامیم . داریم :

$$c(H) = c(H') + c_e \le c.\, x_{res}^* + c_e$$

و چون که $x_e^* = 1$ است هزینه H حداکثر به اندازه OPT خواهد بود .

افزودن یال e به جواب باعث افزایش درجه دو راس u و v به اندازه یک می شود . یعنی $B_u = B'_u + 1$ و $B_v = B'_v + 1$. داریم : $d_H(u) = d_{H'}(u) + 1 \le B'_u + 1 + 1 = B_u + 1$ که رابطه $d_{H'}(u) \le B'_u + 1$ همچنین داریم $d_H(v) \le B_v + 1$.

برای بقیه رئوس هم شرط محدودیت رئوس برقرار است.

حال در نظر بگیرید که در مرحله ۴ ما به جای حذف یال یک محدودیت را از برخی راس های $w \in W$ برداشته ایم . فرض کنید که x' جواب LP ایی است که از برداشتن محدودیت برای برخی رئوس w به وجود آمده است. تعریف می کنیم : $W' = W \backslash w$. با استقرا الگوریتم یک جنگل H برمی گرداند که هزینه آن حداکثر برابر با $c.x'$ و شرط $d_H(v) \le Bv+1$ برای همه رئوس عضو W' برقرار است . واضح است که $c.x' \le c.x^*$ و هزینه آن هنوز کمتر از OPT است .چون که هر بار راسی برای برداشتن محدودیت انتخاب می شود که درجه آن حداکثر یکی بزرگتر از B_v باشد پس برای همه رئوس شرط $d_H(w) \le B_w+1$ برقرار است .

حال **اثبات می کنیم که الگوریتم خاتمه می یابد** به عبارت بهتر در هر مرحله یا یک راس با $x_e^* = 1$ و یا یک راس با درجه کمتر از B_v+1 وجود دارد .

لم ۴.۱ : فرض کنیم که x^* جواب اولیه برای مسئله LP باشد ، مجموعه جواب را E^* می نامیم . در این صورت وجود دارد یک مجموعه T که زیر مجموعه W است و همچنین یک مجموعه laminar به نام L که زیر مجموعه τ(F) است و داریم x^* جواب یکه مسئله خطی سازی زیر است :

$$\begin{cases} x(\delta(v)) = B_v & \forall\, v \in T \\ x(E(S)) = |S| - |F(S)| - 1 & \forall\, S \in L \end{cases}$$

و داریم $|E^*| = |L| + |T|$.

اثبات : جواب بهینه اولیه خطی سازی جواب یکه m تا شرط مساوی مستقل از هم است . فرض کنیم که

$U = \{v \in W : x^*(\delta(v)) = B_v\}$

$M = \{S \subseteq V : \sum_{e \in E(S)} x_e * = |S| - |F(S)| - 1\}$

و فرض کنیم که R,S ای داریم که عضو M هستند و اشتراک آنها تهی نیست . داریم :

$$
\begin{aligned}
&(|R\cap S| - |F(R\cap S)| - 1) + (|R\cup S| - |F(R\cup S)| - 1)\\
&\geq x^*(E(R\cap S)) + x^*(E(R\cup S))\\
&\geq x^*(E(R)) + x^*(E(S))\\
&= |R| - |F(R)| - 1 + |S| - |F(S)| - 1\\
&= (|R\cap S| - |F(R\cap S)| - 1) + (|R\cup S| - |F(R\cup S)| - 1),
\end{aligned}
$$

با توجه به این استدلال هر دو مجموعه R∩S و R∪S باید در M باشند .

اکنون با توجه به تکنیک uncrossing اثبات می شود که یک laminar ماکزیمال مانند L در داخل M وجود دارد که span(L) = span(M) . همچنین یک خانواده ماکزیمال T زیر مجموعه U وجود دارد که span(T) = span(U)

نامساوی های مربوط به T و همچنین نامساوی های مربوط به L یک جواب اولیه feasible به دست می دهند بنابراین |E*| = |L|+|T| .

قضیه ۴.۲ : در هر مرحله از روش حل LP-MBDCT(G,B,W,F) حتماً یکی از شرایط زیر برقرار است:

- یک راس با xe* =1 وجود دارد
- یک راس با درجه کمتر از B_w+1 وجود دارد

با توجه به این قضیه الگوریتم حتماً خاتمه می یابد و درستی آن ثابت می شود.

اثبات : فرض کنیم که شروط اول و دوم قضیه برقرار نباشد ، در نتیجه هر راس حداقل دارای درجه ۳ است ، حال فرض کنیم که L و T مانند قضایای گذشته تعریف شده باشند ، نشان می دهیم که در این صورت |E*| > |T| + |L|.

یک راس را فعال می نامیم اگر حداقل یک یال به آن متصل باشد با استفاده از استدلال شمارش [۱] برای هر یال متصل به هر راس یک مهره به راس فعال می دهیم . در نتیجه تعداد کل مهره ها برابر با 2|E*| است . می توانیم این مهره ها را طوری توزیع مجدد کنیم که هر راس در T و هر راس از مجموعه L دارای ۲ مهره باشند و هنوز تعدادی مهره اضافی وجود داشته باشد . که این نشان می دهد که 2|E*| = 2|L| + 2|T|+2 و در نتیجه |E*|>|L|+ |T| و این تناقض با لم ۴.۱ است .پس فرض خلف ما غلط بوده است .

به علت طولانی بودن اثبات توزیع مجدد از آوردن آن خودداری می شود اما نکاتی درباره چگونگی انجام آن ارائه می شود :

- در این توزیع مجدد سعی می شود که یک مجموعه راس مانند S را طوری توزیع مجدد کنیم که راس ریشه دارای ۴ مهره و بقیه رئوس دارای ۲ مهره باشند.
- ممکن است که ریشه یک برگ باشد در این حالت اگر ۴ راس فعال در کنار آن باشد که مسئله حل است . اگر ۳ راس باشد چندین حالت پیش می آید که جزئیات آن در مقاله اصلی موجود است . و حالتی که ۲ راس فعال موجود باشد وجود ندارد زیرا در این صورت یک راس یکه خواهیم داشت و می دانیم که این راس قبلاً انتخاب شده است.
- ممکن است که S دارای حداقل ۱ فرزند باشد اگر بیشتر یا مساوی ۲ فرزند داشته باشد که مسئله حل است . اگر تنها یک فرزند داشته باشد نیز می توان با روش هایی که در مقاله اصلی ذکر شده توزیع مجدد را انجام داد.

۵ الگوریتم تقریبی A±1

این الگوریتم الهامی از روش قبلی است . در این الگوریتم برای هر راس علاوه بر محدوده بالایی یک محدود پایین نیز داریم . این محدوده را با Av نشان می دهیم . یک برنامه خطی برای حل این مسئله ارائه می دهیم :

LP-MBDST(G,A,B,U,W,F)

$$
\begin{aligned}
&minmize\ c(x) = \sum_{e\in E} c_e x_e\\
&suject\ to:\\
&x(E(V)) = |V| - |F(V)| - 1\\
&x(E(S)) \leq |S| - |F(S)| - 1 \quad \forall\, S \subset \tau(F)\\
&x(\delta(v)) \geq A_v \quad \forall\, v \in U\\
&x(\delta(v)) \leq B_v \quad \forall\, v \in W\\
&x_e \geq 0 \quad \forall\, e \in E
\end{aligned}
$$

الگوریتم تکراری که برای این منظور تهیه شده به صورت زیر است :

MBDCT Algorithm2(*G, A, B, U, W, F*)

1. If *F* is a spanning tree return *0* else let $\hat{F}$ = 0
2. Find a basic optimal solution *x** of LP-MBDCT(G, A, B, U, W, F) and remove every edge *e* with x_e* = 0 from *G*
3. If there exists an edge *e* = {*u, v*} such that x_e* = 1, then set $\hat{F} \leftarrow \{e\}$, $F \leftarrow F \cup \{e\}$ and $G \leftarrow G\backslash\{e\}$. Also set $B_u \leftarrow B_u - 1$, $B_v \leftarrow B_v - 1$, $A_u \leftarrow A_u - 1$ and $A_v \leftarrow A_v - 1$
4. If there exists a vertex $w \in U \cup W$ such that $degE^*(w) \leq B_w + 1$, then update $W \leftarrow W \backslash\{w\}$ and $U \leftarrow U \backslash\{w\}$
5. Return $\hat{F} \cup$MBDCT Algorithm2(*G, A, B, U, W, F*)

اثبات درستی جواب این الگوریتم مانند حالت قبل است ، با استفاده از استقرا می توان ثابت کرد که الگوریتم یک درخت پوشای می نیمم تولید می کند .

[۱] counting argument

برای اثبات خاتمه الگوریتم مانند حالت قبل ابتدا ثابت می کنیم که:

لم ۵.۱ ، وجود دارد مجموعه T_U و T_W زیر مجموعه U و W و یک مجموعه laminar به نام L که زیر مجموعه $\tau(F)$ است که x* جواب یکه تساوی های زیر است :

$$\begin{cases} x(\delta(v)) = A_v & \forall\, v \in T_U \\ x(\delta(v)) = B_v & \forall\, v \in T_W \\ x(E(S)) = |S| - |F(S)| - 1 & \forall\, S \in L \end{cases}$$

و همچنین داریم $|E^*| = |L|+|T_U|+|T_W|$

اثبات همانند حالت قبلی است (الگوریتم A+1)

قضیه ۵.۲ ، در هر مرحله از الگوریتم MBDCT Algorithm2 همیشه یکی از شرایط زیر برقرار است :

- یک یال وجود دارد که $x_e^* = 1$
- یک راس متعلق به U یا W وجود دارد که درجه آن برابر با ۲ است

اثبات : برای اثبات فرض می کنیم که هیچ کدام از شرط های بالا برقرار نباشد ، نشان می دهیم که در این حالت نامساوی لم ۵.۱ برقرار نخواهد بود . برای اثبات مانند حالت قبلی از استدلال شمارشی استفاده می کنیم ، به هر راس فعال (راسی که یالی به آن متصل است) یک مهره می دهیم . این مهره ها را طوری توزیع مجدد می کنیم که به راس ریشه ۳ مهره و به بقیه ۲ مهره برسد . در این حالت داریم : $2|E^*| = 2|L|+2|T_U|+2|T_W|+1$ و در نتیجه $|E^*| > |L| + |T_U|+|T_W|$ و این تناقض با نتیجه لم ۵.۱ است .

برای اثبات چند ادعا و تعریف داریم :

ادعای ۵.۳ : هر supernode دارای یک راس فعال است و درجه این راس برابر با ۳ است .اثبات : هر supernode با توجه به تعریف آن دارای یک راس فعال است و با توجه به فرض خلف درجه راس برابر با ۳ است ، زیرا این راس عضو T است.

تعریف ۵.۴ : هر مجموعه $S \subseteq V$ ویژه محسوب می شود اگر :

- $|\delta(S)| = 3$
- $x^*(\delta(S)) = 1$ or $x^*(\delta(S)) = 2$
- $\chi_{\delta(S)}$ is a linear combination of the characteristic vectors of its descendants (including possibly $\chi_{E(S)}$) and the characteristic vectors $\chi_{\delta(v)}$ of $v \in S \cap T$

لم ۵.۵ : با استفاده از این ۲ تعریف و ادعا می توان اثبات کرد که برای هر مجموعه S می توان مهره ها را طوری توزیع کرد که هر راس در داخل این زیردرخت $T \cap S$ دارای ۲ مهره و ریشه S دارای حداقل ۳ مهره باشد . در حالتی که S برابر V باشد ریشه دارای دقیقاً ۳ مهره خواهد بود .

اثبات : باید بتوانیم برای همه حالات یک توزیع مجدد پیدا کنیم ، برای اثبات نیاز به تعریف عضویت داریم ، هر زیر مجموعه S را یک عضو می نامیم اگر زیر مجموعه دیگر اعضای S نباشد یعنی :

$C \subseteq S$ but $C \not\subset R$ for any R child of S

تعریف می کنیم که D(S) برابر است با تعداد یالهای بین ۲ عضو متفاوت S و ادعا می کنیم که اگر مجموعه S زیر مجموعه L باشد (L خاصیت laminar دارد) و دارای r عضو باشد آنگاه $x^*(D(S)) = r-1$ اثبات با جزئیات در مقاله اصلی موجود است.

ادعا می کنیم که اگر مجموعه S دارای دقیقاً ۳ عضو ویژه باشد (عضوی که دارای یک راس فعال هستند) آنگاه S نیز یک مجموعه ویژه خواهد بود . اثبات این مورد نیز از حوصله گزارش خارج است و در داخل مقاله اصلی وجود دارد . نکات کلیدی برای اثبات : اثبات وجود ۳ شرط تعریف ۵.۴ برای مجموعه S است .

و در آخر اثبات لم ۵.۵ است ، به دلیل طولانی بودن خلاصه ای از اثبات آورده می شود ، در حالت اولیه هر عضو S دارای حداقل یک مهره اضافی است و اگر این عضو ویژه باشد دارای دقیقاً یک مهره اضافی خواهد بود شرایط زیر ممکن است وجود داشته باشد :

- S دارای حداقل ۴ عضو است ، پس می تواند از هر کدام یک مهره بگیرد و در نتیجه S دارای حداقل ۴ مهره خواهد شد .
- S دارای دقیقاً ۳ عضو است ، اگر هر مهره دارای حداقل ۲ مهره اضافی باشد که S آنها را جمع می کند در غیر این صورت هر مهره دارای تنها یک مهره است و در نتیجه (با توجه به تعریف) S یک مجموعه ویژه است . و تنها به ۳ مهره نیاز دارد در غیر این حالت یعنی $x^*(D(S)) = 2$ نیز ثابت می شود که S مجموعه ویژه است.
- S دارای دقیقاً ۲ عضو است ، اگر R1 و R2 هر کدام دارای حداقل ۲ مهره اضافی باشند که مسئله حل است ، در غیر این صورت یکی از آنها مانند R1 دارای دقیقاً یک مهره است و در نتیجه مجموعه ویژه است ، و می توان ثابت کرد که S می تواند ۳ مهره از داخل این مجموعه کسب کند . (جزئیات اثبات ذکر نشده است)

۶ منبع

- Approximating Minimum Bounded Degree Spanning Trees to within One of Optimal , Mohit Singh Lap Chi Lau, *STOC'07,* June 11–13, 2007, San Diego, California, USA.