الگوریتم تقریبی برای رنگ آمیزی رأسی گراف‌ها

محمّد علی آبام ***محمد قدسی***
دانشکدهٔ مهندسی کامپیوتر، دانشگاه صنعتی شریف

چکیده

در این مقاله به بررسی الگوریتم‌های تقریبی برای رنگ آمیزی راسی گراف‌ها پرداخته‌ایم. برای بدست آوردن ضریب تقریب بهتر، الگوریتم هایی که اخیراً طراحی شده‌اند الگوریتم‌های احتمالاتی هستند چرا که الگوریتم های حریصانه ارائه شده نتوانسته‌اند ضریب تقریب بدست آمده برای رنگ آمیزی راسی را بهبود بخشند. در این مقاله یک الگوریتم حریصانه برای رنگ آمیزی راسی گراف ها با ضریب تقریب $O(n/\log n)$ ارائه شده است.

کلیدواژه‌ها: رنگ آمیزی گراف‌ها، الگوریتم تقریبی، تحلیل الگوریتم.

۱ مقدمه

رنگ آمیزی راسی گراف $G = (V, E)$، نسبت دادن رنگ ها به رئوس گراف می باشد به طوری که هیچ دو راس مجاوری همرنگ نباشند به عبارت دیگر رنگ آمیزی گراف G با k رنگ، افراز رئوس به k مجموعه مستقل می باشد. کمترین تعداد رنگی که برای رنگ آمیزی راسی گراف G نیاز می باشد را عدد رنگی [۱] G می نامند که با $\chi(G)$ نمایش داده می شود. تعیین عدد رنگی یک گراف مساله NP-hard می‌باشد. الگوریتم های تقریبی زیادی برای رنگ آمیزی راسی گراف ها مطرح شده اند. ضریب تقریب الگوریتم رنگ آمیزی برابر ماکزیمم نسبت، برای همه گراف‌ها با n راس، تعداد رنگ های استفاده شده به مینیمم تعداد رنگ مورد نیاز می باشد.

رنگ آمیزی راسی گراف ها کاربردهای زیادی دارد که به عنوان نمونه می توان زمان بندی امتحان‌ها را نام برد. در بسیار از کاربردها مانند مثالی که ذکر شد جواب دقیق مورد نیاز نمی باشد و جواب تقریبی برای حل مساله کفایت می‌کند.

[۱] Chromatic Number

اولین الگوریتم تقریبی توسط johanson در سال ۱۹۷۴ مطرح شد که این الگوریتم، الگوریتمی حریصانه با ضریب تقریب $O(n/\log n)$ می باشد. wigderson در سال ۱۹۸۳ الگوریتمی را مطرح کرد که از $O(n^{1-1/(k-1)})$ رنگ برای رنگ آمیزی گراف k رنگ پذیر G استفاده می کرد. الگوریتمی دیگری توسط blum طراحی شد که از $O(n^{3/8}\log^{8/5} n)$ رنگ برای رنگ آمیزی برای گراف ۳ رنگ پذیر G استفاده می کرد که این الگوریتم در حالت کلی از $O(k^{1-1/(k-4/3)}\log^{8/5} n)$ رنگ برای رنگ آمیزی گراف k رنگ پذیر G استفاده می کند. widerson الگوریتم دیگری با ضریب تقریب $O(n(\log\log n/\log n)^2)$ مطرح کرد. بهترین ضریب تقریب بدست آمده الگوریتم halldersson[۱] می باشد که دارای ضریب تقریب $O(n(\log\log n)^2/\log^3 n)$ می باشد. الگوریتم دیگری که توسط karger[۲] مطرح شد از $min\{O(\Delta^{1-2/k}\log^{1/2}\Delta\log n), O(n^{1-3/(k+1)}\log^{1/2} n)\}$ رنگ برای رنگ آمیزی گراف k رنگ پذیر G استقاده می کند. این الگوریتم یک الگوریتم احتمالاتی است و تنها الگوریتمی است که تعداد رنگ ها براساس Δ، ماکزیمم درجه گراف، می باشد. علاوه بر الگوریتم های ارائه شده چندین حدس نیز مطرح شده است که از جمله می توان حدس زیر را نام برد :

حدس بهترین تقریبی که می توان برای رنگ آمیزی راسی گراف ها بدست آورد، $O(n/\log^c n)$ برای یک $c \geq 3$ می باشد.

در بخش ۲ الگوریتمی حریصانه برای رنگ آمیزی راسی ارائه خواهد شد و در بخش ۳ تحلیل الگوریتم بیان خواهد شد و سرانجام در بخش ۴ خلاصه و نتیجه گیری بحث آورده شده است.

۲ الگوریتمی حریصانه برای رنگ آمیزی رأسی

فرض می کنیم گراف $G=(V,E)$، k رنگ پذیر باشد. در این صورت مجموعه V را می‌توان به زیرمجموعه های مستقل $V_1,\cdots,V_k$ افراز کرد. اگر تعداد رئوس G را n بنامیم طبق اصل لانه کبوتری یکی از زیرمجموعه های $V_1,\cdots,V_k$ دارای n/k راس می باشد؛ این زیرمجموعه را V_l می‌نامیم. مجموعه رئوس گراف را به دلخواه به زیرمجموعه هایی افراز می کنیم که هر زیرمجموعه $k*\log_k n$ راس را دارا باشد؛ تعداد این زیرمجموعه‌ها برابر $\frac{n}{k*\log_k n}$ می باشد. طبق اصل لانه کبوتری یکی از این زیرمجموعه ها

$$\frac{n/k}{\frac{n}{k*\log_k n}} = \log_k n$$

راس از زیر مجموعه V_l را دارا می باشد که چون مجموعه V_l مستقل است این $\log_k n$ راس نیز مستقل می باشند. از آنجا که هر زیر مجموعه $k*\log_k n$ راس دارد و $C^{k*\log_k n}_{\log_k n} \approx n$ می توان $\log_k n$ راس مستقل را در زمان $(k*\log_k n)*n$ پیدا کرد. به $\log_k n$ راس مستقل بدست آمده یک رنگ اختصاص می دهیم و از گراف حذف می کنیم؛ گراف باقی مانده همچنان k رنگ پذیر است. مراحل فوق را برای گراف باقی مانده تکرار می کنیم تا در نهایت همه رئوس گراف رنگ شوند. این الگوریتم را برای $k=1,\cdots,n$ تکرار می کنیم به ازای هر k الگوریتم از یک تعداد رنگ استفاده می کند(ممکن است برای $k<\chi(G)$ الگوریتم به تناقض برسد یا نرسد اما برای $k\geq\chi(G)$ حتما یک تعداد رنگ بر می گرداند) که کمترین این اعداد به عنوان خروجی الگوریتم در نظر می گیریم. بطور دقیق می توان الگوریتم را به شکل زیر بیان کرد :

۱) برای $k = ۱, \cdots, n$ مراحل زیر را انجام بده ($m = n, no\_color_k = ۰$) :

(a) مجموعه رئوس باقی مانده از گراف را به دلخواه به مجموعه هایی افراز کن که هر زیرمجموعه $k * \log_k m$ راس داشته باشد.

(b) با جستجو در هر زیرمجموعه $\log_k m$ راس را پیدا کن که مستقل باشند.

(c) به $\log_k m$ راس بدست آمده یک رنگ اختصاص بده و no_color_k را یکی افزایش بده. $\log_k m$ راس را از گراف حذف کن و $m = m - \log_k m$. اگر راسی باقی مانده است به مرحله 1a برو.

۲) از بین $no\_color_k (k = ۱, \cdots, n)$ کوچکترین عدد را به عنوان خروجی الگوریتم در نظر بگیر.

۳ تحلیل الگوریتم

۱.۳ ضریب تقریب الگوریتم

به ازای هر k می توان یک رابطه بازگشتی از تعداد رئوس باقی مانده در هر مرحله از الگوریتم بدست آورد؛ این رابطه بازگشتی به شکل زیر است :

$$a_۱ = n; a_{i+۱} = a_i - \log_k a_i$$

اگر j کوچکترین عددی باشد که $a_j < ۱$ در این صورت برای رنگ آمیزی از j رنگ استفاده می شود. حل این رابطه بازگشتی به سادگی امکانپذیر نیست. اما از آنجا که اعدادی که استفاده می کنیم اعداد صحیح هستند به عبارت دیگر وقتی می گوییم $\log_k n$ راس در حقیقت منظور قسمت صحیح آن است می توان تعداد رنگ های استفاده شده را با فرض $n = k^{x+۱}$ به شکل زیر بدست آورد :

$$\frac{k^{x+۱} - k^x}{x} + \cdots + \frac{k^۲ - k^۱}{۱}$$

که این عبارت را می توان به شکل زیر نوشت :

$$(k - ۱) * (\frac{k^x}{x} + \cdots + \frac{k}{۱})$$

اما از آنجا که داریم

$$\frac{k^x}{x} + \cdots + \frac{k}{۱} = \int_۰^k \frac{t^x - ۱}{t - ۱} dt \leq \int_۰^k t^x dt = \frac{k^{x+۱}}{x + ۱} = O(n/\log n)$$

و k از ۱ تا n تغییر می کند در یک جا $k = \chi$ خواهد بود بنابراین می توان گفت با حداکثر $O((\chi - ۱) * \frac{n}{\log n})$ رنگ، گراف را می توان رنگ آمیزی کرد و چون جواب بهینه برابر χ می باشد ضریب تقریب الگوریتم برابر

$$O(\frac{\chi - ۱}{\chi} * \frac{n}{\log n}) = O(\frac{n}{\log n})$$

می شود.

۲.۳ زمان اجرای الگوریتم

قسمت ۱ الگوریتم n بار تکرار می شود. به ازای هر تکرار قسمت 1a زمان اجرایی از $O(1)$ دارد و مرحله 1b زمان اجرایی از $O(\frac{n}{k*\log_k n} * n)$ دارد چرا که تعداد زیر مجموعه ها برابر $\frac{n}{k*\log_k n}$ می باشد و جستجو در هر زیر مجموعه $O(n)$ زمان نیاز دارد. در قسمت 1c در صورت باقی ماندن راسی به مرحله 1a می رود و از آنجا که در هر مرحله حداقل یک راس رنگ می شود تعداد این پرش ها حداکثر n بار می باشد بنابراین زمان اجرای مراحل 1a، 1b و 1c

$$O(\frac{n}{k*\log_k n} * n * n) = O(\frac{n^3}{k*\log_k n})$$

می باشد. پس می توان گفت مرحله ۱ الگوریتم در زمان $O(\sum_{k=1}^{n} \frac{n^3}{k*\log_k n})$ انجام می شود و چون مرحله ۲ الگوریتم در زمان $O(n)$ انجام می شود(پیدا کردن مینیمم) در کل زمان اجرای الگوریتم از

$$O(\sum_{k=1}^{n} \frac{n^3}{k*\log_k n})$$

می باشد که اگر مجموع فوق با تقریب محاسبه کنیم این الگوریتم زمان اجرایی از $O(n^4)$ خواهد داشت.

۴ خلاصه و نتیجه گیری

پیدا کردن الگوریتم تقریبی برای رنگ آمیزی راسی گراف ها به خاطر کاربردهای زیاد رنگ آمیزی راسی چندین سال است که مورد توجه قرار گرفته است. الگوریتم اولیه، الگوریتم های حریصانه بوده اند اما از این الگوریتم ها ضریب تقریب خوبی حاصل نمی شد به همین دلیل الگوریتم هایی که اخیراً مطرح شده اند الگوریتم های احتمالاتی هستند که تا حدودی توانسته اند ضریب تقریب بهتری را حاصل کنند. در این مقاله خلاصه ای از ضریب تقریب های بدست آمده تا کنون همراه با الگوریتم karger که یک الگوریتم احتمالاتی است ارائه شده است و در نهایت یک الگوریتم حریصانه با ضریب تقریب $O(n/\log n)$ بیان شده است. این الگوریتم از فرمول مهم $C_{\log_k n}^{k*\log_k n} \approx n$ استفاده می کند که در حقیقت پایه طراحی این الگوریتم بوده است.

مراجع

[1] Halldersson, "A still performance guarantee for approximate graph coloring",Information Process Letter, vo.45 , 1993, p.19-23.

[2] Karger, Motwaini and Sundan, "Approximate graph coloring semidefinite programming", Proc. 36th Ann,IEEE Symo on foundations of comput. Sci. IEEE computer society, 1994, p.2-13