الگوریتم موازی برای مساله برچسب‌گذاری نقشه‌ها

شروین دانش‌پژوه[۱]، محمد قدسی[۲]

دانشگاه صنعتی شریف، دانشکده مهندسی کامپیوتر

daneshpajouh@ce.sharif.edu
ghodsi@sharif.edu

چکیده

مساله برچسب‌گذاری نقشه‌ها[۳] یکی از مساله‌های قدیمی نقشه‌کشی[۴] است. نقشه‌ای حاوی مجموعه‌ای از نقاط داریم و هر نقطه در این مجموعه نقاط دارای تعدادی کاندیدا(مربع) است. هدف یافتن اندازه بهینه کاندیداها است بنحوی که کاندیداها با یکدیگر تداخل نداشته باشند و هر نقطه دارای حداقل یک کاندیدا باشد. نقشه می‌تواند یک نقشه معمولی(نقشه یک کشور)، نمودار، گراف یا هر شکل دیگری که نیاز به برچسب‌گذاری دارد، باشد. چند الگوریتم تقریبی برای این مساله وجود دارند. یکی از این الگوریتم‌ها دارای زمان اجرا و تقریب بهینه است و در عمل هم خوب کار می‌کند. در این مقاله ما از این الگوریتم بعنوان الگوریتم پایه استفاده کرده و یک الگوریتم موازی برای مساله برچسب‌گذاری نقشه‌ها ارائه می‌کنیم. الگوریتم موازی که ارائه می‌شود اولین الگوریتم موازی برای مساله برچسب‌گذاری نقشه‌ها است. این الگوریتم دارای افزایش سرعت برابر با $\log_2^p$ نسبت به الگوریتم غیر موازی است.

واژه‌های کلیدی: برچسب‌گذاری نقشه‌ها، پردازش موازی، هندسه محاسباتی، الگوریتم‌های تقریبی

۱-مقدمه

یکی از کارهایی که تهیه‌کنندگان نقشه بهنگام تهیه نقشه انجام می‌دهند، برچسب‌گذاری نقاط نقشه است. بدین معنی که در کنار هر نقطه اطلاعاتی راجع به آن نقطه می‌نویسند. اطلاعات باید به گونه‌ای برروی نقشه قرارگیرند که اطلاعات نقاط مختلف برروی هم قرار نگیرند. انجام خودکار برچسب‌گذاری نقشه توسط رایانه باعث سهولت و تسریع در انجام اینکار می‌شود. برای رایانه‌ای کردن این محاسبه نیاز داریم که برچسب‌گذاری نقشه را بصورت دقیق‌تر بیان کنیم.

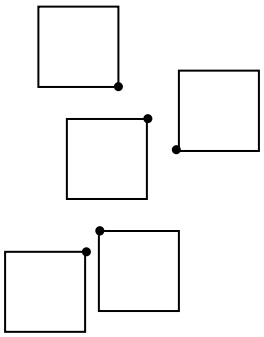
---

[1] دانشجوی کارشناسی ارشد کامپیوتر-نرم‌افزار

[2] استاد دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف

[3]. Map Labeling

[4]. Cartography



شکل ۱: نمونه‌ای از برچسب‌گذاری معتبر

تعریف ۱. برچسب‌گذاری معتبر: در یک صفحه n نقطه جدا از هم داده شده‌اند. می‌خواهیم یک مجموعه n تایی از مربع‌ها به طول σ داشته باشیم بطوریکه هر نقطه کنج یک مربع(و نه بیشتر) قرار گیرد و کلیه مربع‌ها با یکدیگر تداخل نداشته باشند(شکل ۱).

تعریف ۲. برچسب‌گذاری بهینه: در یک صفحه n نقطه جدا از هم داده شده‌اند. به‌ازای کلیه اعداد حقیقی σ می‌خواهیم سوپریمم σ_{opt} را بنحوی پیدا کنیم که یک برچسب‌گذاری معتبر داشته باشیم(شکل ۲).

در سال ۹۲، فورمن و واگنر نشان دادند که مساله برچسب‌گذاری بهینه NP-hard است[۱]. آنها همچنین الگوریتم تقریبی ارائه‌دادند که برچسب‌گذاری معتبری به اندازه‌حداقل نصف اندازه بهینه می‌یافت. همچنین آنها نشان دادند که اگر P≠NP آنگاه الگوریتم تقریبی با جواب چندجمله‌ای وجود ندارد که پاسخ بهتری را تضمین کند. پیچیدگی الگوریتم آنها $O(n \log n)$ است. نتیجه مشابهی نیز در [۲] و [۳] گزارش شده است. در سال ۹۵ واگنر و ولف الگوریتم جدیدی ارائه دادند که زمان اجرا و کیفیت پاسخ الگوریتم قبلی را تضمین می‌کرد[۴]. نقطه قوت این الگوریتم نسبت به الگوریتم قبلی این است که الگوریتم [۴] پاسخ‌های نزدیکتری به اندازه بهینه برچسب‌گذاری (در عمل) می‌دهد.

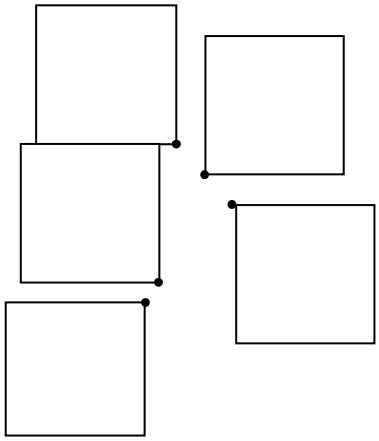
در این مقاله ما یک الگوریتم موازی مبتنی بر الگوریتم [۴]، برای افزایش سرعت اجرای برچسب‌گذاری، ارائه می‌دهیم. در بخش ۲ الگوریتم غیرموازی [۴] را بیان می‌کنیم. در بخش ۳ الگوریتم موازی را معرفی می‌کنیم و پیچیدگی آنرا حساب می‌کنیم و در بخش ۴ هم نتیجه‌گیری آمده است. این الگوریتم دارای افزایش سرعت برابر با $\log_2^p$ نسبت به الگوریتم غیر موازی است.

۲- الگوریتم غیرموازی برچسب‌گذاری نقشه‌ها

الگوریتمی که شرح آن در ادامه می‌آید، الگوریتمی است که توسط واگنر و ولف در سال ۱۹۹۵ معرفی شده است[۴]. ابتدا چند تعریف، که در ادامه از آنها استفاده خواهیم نمود را بیان می‌کنیم.

تعریف ۳: برای هر نقطه مانند p واقع در نقشه، σp_i بیانگر مربعی به طول σ است که نقطه p در جنوب غربی، جنوب شرقی، شمال غربی یا شمال شرقی آن قرار دارد. $i \in \{1,2,3,4\}$ و $\sigma \geq 0$ یک مقدار حقیقی است. همچنین p_i را کاندیدای iام نقطه p می‌نامیم.

تعریف ۴: برچسب‌گذاری با اندازه برچسب σ عبارتست از برچسب‌گذاری معتبری که در آن کاندیداها دارای طول یکسان و برابر σ باشند.



شکل ۲: نمونه‌ای از برچسب‌گذاری بهینه

۲-۱- ساختار الگوریتم غیر موازی

مرحله ۱: پیدا کردن کلیه اندازه‌های تداخل با ارزش.

مرحله ۲: انجام جستجوی دودویی برروی اندازه‌های تداخل بدست آمده از مرحله قبل. برای هر اندازه تداخل بررسی می‌کنیم که آیا پاسخی برای این اندازه برچسب وجود دارد یا خیر. اینکار را با اعمال سه گام زیر انجام می‌دهیم.

گام الف: پیش پردازش.

گام ب: حذف کاندیداهای غیر ممکن و سپس انجام آزمایش 2-SAT برروی زیر مجموعه باقی مانده.

گام ج: برای نقاطی که بیشتر از دو کاندیدا دارند، دو کاندیدا را انتخاب کرده و سپس توسط 2-SAT بررسی می‌کنیم که آیا پاسخی وجود دارد یا خیر.

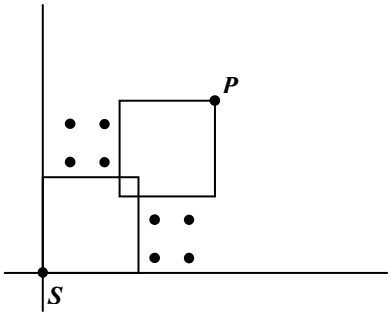
۲-۲- پیدا کردن اندازه‌های تداخل با ارزش

روشی که برای پیدا کردن اندازه‌های تداخل توسط ولف و واگنر معرفی شده است مبتنی بر دو قضیه است که آنها را اثبات نموده‌اند[۴].

قضیه ۱: کلیه اندازه‌های تداخل با ارزش کوچکتر از اندازه برچسب بهینه(σ_{opt}) هستند. بعبارت دیگر اندازه‌های تداخل بی‌ارزش بزرگتر از اندازه برچسب بهینه هستند.

قضیه ۲: تنها اندازه‌های تداخل بین نقطه p و نزدیکترین هشت نقطه همسایه‌اش در هر یک از چهارسمت نقطه p با ارزش هستند. شکل ۳ نمونه‌ای از اندازه تداخل بی‌ارزش را نشان می‌دهد.

طبق قضیه ۲، برای بدست آوردن اندازه‌های تداخل نقطه‌ای مانند p، تنها باید تعداد ثابتی از نزدیکترین همسایه‌های آن نقطه مورد بررسی قرار گیرند. در [۵] الگوریتمی برای پیدا کردن نزدیکترین k همسایه n نقطه ارائه شده است که دارای پیچیدگی زمانی $O(kn \log n)$ است. مرتب‌سازی اندازه‌های تداخل نیز $n \log n$ زمان نیاز دارد. بنابراین بدست‌آوردن اندازه‌های تداخل و مرتب‌سازی دارای پیچیدگی زمانی $O(n \log n)$ خواهد بود.



شکل ۳: تداخل بین دو کاندیدای s1 و p3 با ارزش نیستند

۲-۳- بدست آوردن اندازه برچسب بهینه

در این مرحله برای بدست آوردن اندازه برچسب بهینه، در فهرست اندازه‌های تداخل به جستجوی دودویی می‌پردازیم. به ازای هر اندازه تداخل(که جستجوی دودویی آنرا مشخص می‌کند) سه گامی که در ادامه می‌آید را انجام می‌دهیم.

۲-۳-۱- گام الف: پیش پردازش

برای کلیه کاندیداهای مانند p_i بررسی می‌کنیم، اگر σp_i حاوی یک نقطه دیگر باشد، آنگاه کاندیدای p_i را حذف می‌کنیم در غیر اینصورت فهرست جدیدی از اطلاعات تداخل برای کاندیدای p_i می‌سازیم. این فهرست شامل اندازه‌های تداخلی خواهد بود که از اندازه برچسب σ کوچکتر هستند.

۲-۳-۲- گام ب: حذف کاندیداهای غیر ممکن

در این مرحله برای کلیه نقاط مانند p چهار حالت زیر را بررسی می‌کنیم:

- اگر کلیه کاندیداهای نقطه p حذف شده بودند، آنگاه برای اندازه برچسب σ مورد بررسی، پاسخی وجود ندارد. تابع مجری این قسمت در اینجا متوقف شده و به بخشی از برنامه که جستجوی دودویی را انجام می‌دهد باز می‌گردد.
- اگر نقطه p کاندیداهای بدون تداخل با سایر کاندیداها داشته باشد، یکی از آنها را بدلخواه انتخاب کرده و سایر کاندیداهای نقطه p را حذف می‌کنیم. قبل از انجام عمل حذف کاندیدای مربوطه را از فهرست تداخل سایر کاندیداها نیز حذف می‌کنیم.
- اگر تنهای یک کاندیدا برای نقطه p باقی مانده باشد، آنگاه کاندیداهای تداخل‌دار با این کاندیدا را حذف می‌کنیم.
- اگر p دارای کاندیدای p_i باشد که با دو کاندیدای باقی‌مانده نقطه q تداخل داشته باشد، آنگاه p_i را حذف می‌کنیم.

تعریف ۵: کاندیدای p_i را $\sigma - dead$ می‌نامیم اگر σp_i حاوی نقطه دیگری باشد.

لم ۱: پس از اجرای گام ب، کلیه نقاط بیش از دو کاندیدا که $2\sigma - dead$ نباشند نخواهند داشت.

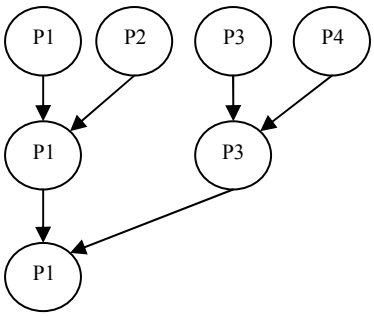
بنابراین طبق لم ۱، می‌توانیم 2-SAT را برروی مجموعه‌ای از کاندیداها که هنوز تداخل دارند و $2\sigma - dead$ نیستند، اجرا کنیم. اگر 2-SAT جوابی برای این مجموعه یافت، کافی است کلیه کاندیداهای بدون تداخل را به جواب اضافه کنیم و این جوابی برای اندازه برچسب مورد بررسی خواهد بود.

۲-۳-۲- گام ج: اعمال الگوریتم مکاشفه‌ای

اگر آزمون 2-SAT مرحله قبلی جوابی را پیدا نکند آنگاه الگوریتم مکاشفه‌ای را اعمال می‌کنیم. الگوریتم مکاشفه‌ای که در اینجا معرفی می‌شود، الگوریتم مکاشفه‌ای I است که در عمل پاسخ بهتری را می‌دهد. ابتدا به سراغ نقاطی که چهار کاندیدا دارند می‌رویم. از میان آنها کاندیدایی را که بیشترین تداخل را دارد حذف می‌کنیم. سپس به سراغ نقاطی که سه کاندیدا دارند می‌رویم و همین عمل را برای آنها نیز تکرار می‌کنیم. بنابراین نقاط باقی‌مانده دوکاندیدا بیشتر نخواهند داشت. با انجام آزمایش 2-SAT بررسی می‌کنیم که پاسخی وجود دارد یا خیر.

۲-۴- تحلیل پیچیدگی الگوریتم غیر موازی

مرحله ۱ الگوریتم، پیدا کردن اندازه‌های تداخل با ارزش، دارای پیچیدگی $O(n \log n)$ است. گام‌های الف، ب و ج از مرحله ۲، در زمان خطی قابل انجام هستند. از آنجایی که ما باید برروی اندازه‌های تداخل جستجوی دودویی را انجام دهیم، مرحله ۲ در زمان $O(n \log n)$ انجام می‌شود. بنابراین پیچیدگی کل الگوریتم غیرموازی برچسب‌گذاری نقشه‌ها $O(n \log n)$ خواهد بود.



شکل ۴: مراحل مرتب‌سازی نقاط توسط ۴ پردازنده

۳- الگوریتم موازی برچسب‌گذاری نقشه‌ها

الگوریتم موازی که برای مساله برچسب گذاری نقشه معرفی می‌کنیم بر پایه الگوریتم غیر موازی گفته شده طراحی شده است. فرض ما این است که تعداد پردازنده‌ها توانی از ۲ هستند و الگوریتم خود را طبق این فرض طراحی می‌کنیم. P_i را پردازنده i ام می‌نامیم. اگر p پردازنده داشته باشیم آنگاه پردازنده‌ها را بصورت $P_1, P_2, ..., P_i, ..., P_{p-1}, P_p$ می‌نامیم. از میان این پردازنده‌ها پردازنده P_1 پردازنده مبصر خواهد بود.

۳-۱- ساختار الگوریتم موازی

مرحله ۱: پیداکردن کلیه اندازه‌های تداخل با ارزش بصورت موازی

گام ۱: مرتب‌سازی نقاط بصورت موازی

گام ۲: محاسبه اندازه‌های تداخل توسط پردازنده‌ها

گام ۳: مرتب‌سازی اندازه‌های تداخل بصورت موازی

مرحله ۲: انجام جستجوی موازی برای پیدا کردن اندازه برچسب بهینه

۳-۲- پیداکردن اندازه‌های تداخل با ارزش بصورت موازی

برای پیدا کردن اندازه‌های تداخل n نقطه بصورت موازی، باید نقاط را بین p پردازنده تقسیم کنیم. از آنجایی که فرض این است که نقاط نامرتب هستند، ابتدا آنها را به بصورت موازی مرتب می‌کنیم. بدین ترتیب می‌توان محاسبه اندازه‌های تداخل نقاط واقع در یک محدوده خاص را به پردازنده مورد نظر محول نمود. مرتب‌سازی نقاط می‌تواند به دو صورت طولی یا عرضی صورت گیرد. با توجه به پراکندگی نقاط می‌توان راجع به نحوه مرتب‌سازی تصمیم‌گیری نمود. اگر پیش‌فرضی در این مورد نداشته باشیم می‌توان با یکبار پیمایش نقاط(توسط پردازنده مبصر) محدوده کمینه و بیشینه نقاط را بدست آورد و با توجه به آن نسبت به نحوه مرتب‌سازی تصمیم‌گیری نمود. اگر محور طولها بزرگتر از محور عرض‌ها باشد، نقاط را برحسب مختصات طولی مرتب می‌کنیم و بالعکس.

۳-۲-۱- مرتب‌سازی نقاط بصورت موازی

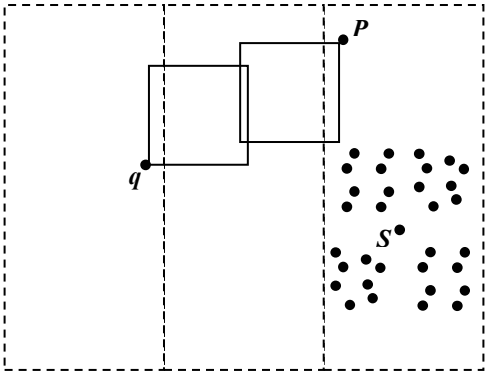
پردازنده مبصر(P_1) از نقطه شماره یک تا نقطه شماره n/p در فهرست نقاط را کنار می‌گذارد که خود مرتب کند. سپس از نقطه شماره $n/p+1$ تا نقطه شماره $2n/p$ در فهرست نقاط را برای پردازنده P_2،... و از نقطه شماره $(p-1)n/p+1$ تا نقطه شماره n را برای پردازنده P_p ارسال می‌کند. هر پردازنده بصورت مستقل نقاط رسیده را مرتب می‌کند. سپس پردازنده‌های زوج p، نقاط مرتب شده را برای پردازنده فرد به شماره $p-1$ ارسال می‌کند. در هر مرحله پردازنده‌ای که داده ارسال می‌کند از دور محاسبه خارج می‌شود و غیر فعال می‌شود(تا زمانی که پردازنده مبصر داده‌ای را به آن ارسال کند). پردازنده‌هایی که داده‌ای دریافت کرده‌اند با استفاده از الگوریتم مرتب‌سازی ادغامی، نقاط را مرتب می‌کند. سپس مجددا پردازنده‌های زوج(زوج در میان پردازنده‌های فعال) نقاط مرتب شده را به پردازنده‌های فرد(فرد در میان پردازنده‌های فعال) ارسال می‌کند و مراحل قبلی مجددا تکرار می‌شود. اینکار $\log p$ مرحله تکرار می‌شود. شکل ۴ مراحل انجام اینکار را نشان می‌دهد. پس از $\log p$ مرحله پردازنده مبصر(P_1) فهرست کامل مرتب شده نقاط را خواهد داشت. این پردازنده فهرست مرتب شده نقاط را به سایر پردازنده‌ها ارسال می‌کند. کلیه پردازنده‌هایی که غیر فعال شده بودند با دریافت این فهرست از پردازنده مبصر فعال شده و آماده محاسبه اندازه‌های تداخل می‌شوند.

پیچیدگی زمانی این الگوریتم با p پردازنده و n نقطه برابر $n/p*(\log n/p+2^{(\log p+1)}-2)$ است. زیرا در ابتدای کار هر پردازنده تعداد n/p نقطه را در زمان $n/p \log n/p$ مرتب می‌کند. در مرحله بعد نیمی از پردازنده‌ها بیکار شده و نیم دیگر پردازنده‌ها 2n/p کار برای ادغام نقاط انجام می‌دهند. بدین ترتیب در مرحله i ام پردازنده‌های فعال هرکدام $2^i*n/p$ کار انجام می‌دهند. اینکار $\log p$ مرحله به طول می‌انجامد. بنابراین پیچیدگی کل الگوریتم مرتب‌سازی بصورت موازی برابر می‌شود با:

$$n/p*\log n/p+2*n/p+4*n/p+...+p*n/p$$
$$=n/p*\log n/p+\sum_{i=0}^{\log p}(2^i*n/p)-n/p$$
$$=n/p(\log n/p+2^{\log p}-2)$$

۳-۲-۲- محاسبه اندازه‌های تداخل توسط پردازنده‌ها

در این مرحله هر پردازنده فهرست مرتب شده نقاط را به $p+1$ ناحیه تقسیم می‌کند. سپس هر پردازنده P_i اندازه‌های تداخل بین نقاط شماره $(i-1)*n/(p+1)$ و $(i+1)*n/(p+1)$ در فهرست مرتب شده نقاط را طبق روش گفته شده در ۲-۲ بصورت غیر موازی محاسبه می‌کند. چون هر پردازنده اندازه‌های تداخل دو ناحیه را محاسبه می‌کند و هر دو پردازنده همسایه یک ناحیه مشترک دارند، بنابراین اندازه‌های تداخل داخل هر ناحیه و بین دو ناحیه همسایه را هم بدست می‌آوریم. حال کافی است نشان دهیم که بین دو ناحیه غیرهمسایه اندازه تداخل با ارزش وجود ندارد.



شکل ۵: بدلیل وجود نقطه s، اندازه تداخل $q1$ و $p3$ بی‌ارزش است

لم ۲: اگر تنها در یک ناحیه از این $p+1$ ناحیه، نقطه‌ای باشد که در هر چهار سمت خود هشت همسایه(در همان ناحیه) داشته باشد، آنگاه اندازه‌های تداخل بین نقاط یک ناحیه و نقاط واقع در نواحی غیرهمسایه آن بی‌ارزش خواهند بود.

اثبات: طبق قضیه ۲ می‌دانیم که تنها اندازه‌های تداخل بین نقطه q و نزدیکترین هشت نقطه همسایه‌اش در هر یک از چهارسمت نقطه q با ارزش هستند. بنابراین اگر تنها در یک ناحیه از این $p+1$ ناحیه، نقطه q باشد که در هر چهار سمت خود هشت همسایه (در همان ناحیه) داشته باشد، آنگاه اندازه‌های تداخل بین نقاط یک ناحیه و نقاط واقع در نواحی غیرهمسایه آن بزرگتر از بزرگترین اندازه تداخل نقطه q و همسایگانش خواهد بود و در نتیجه جزء اندازه‌های تداخل بی‌ارزش خواهند بود. شکل ۵ این موضوع را نشان می‌دهد. ■

بنابراین در حل این مساله فرض می‌کنیم که چنین نقطه‌ای وجود دارد. با توجه به اینکه پردازش موازی زمانی بکار گرفته می‌شود که تعداد نقاط زیاد باشد، احتمال عدم وجود چنین نقطه‌ای بسیار کم است.

پیچیدگی محاسبه اندازه‌های تداخل توسط هر پردازنده برابر است با $O((2n/(p+1)) * \log(2n/(p+1)))$

۳-۲-۳- مرتب‌سازی اندازه‌های تداخل بصورت موازی

پس از محاسبه اندازه‌های تداخل توسط هر پردازنده مرحله مرتب‌سازی اندازه‌های تداخل آغاز می‌شود. مجددا از الگوریتم مرتب‌سازی ادغامی که در ۳-۲-۱ شرح داده شد استفاده می‌شود. هر پردازنده ابتدا اندازه‌های تداخلی که محاسبه نموده را مرتب نموده و سپس پردازنده‌های زوج فهرست اندازه‌های تداخل مرتب شده را به پردازنده‌های فرد می‌دهند و ادامه کار، درست مانند مرتب‌سازی نقاط که در ۳-۲-۱ شرح داده شد می‌باشد. پس از $\log p$ مرحله پردازنده P_1 فهرست کلیه اندازه‌های تداخل را خواهد داشت. درنتیجه پیچیدگی الگوریتم مرتب‌سازی اندازه‌های تداخل برابر می‌شود با: $2n/(p+1) * (\log 2n/(p+1) + 2^{(\log p+1)} - 2)$

۳-۳- انجام جستجوی موازی برای پیدا کردن اندازه برچسب بهینه

در الگوریتم غیرموازی برچسب‌گذاری نقشه‌ها توضیح دادیم که برای بدست‌آوردن اندازه برچسب بهینه روی فهرست اندازه‌های تداخل به جستجوی دودویی می‌پردازیم و برای هر اندازه تداخل بررسی می‌کنیم آیا برچسب‌گذاری معتبری برای اندازه برچسب مورد بررسی وجود دارد یا خیر. با توجه به وجود p پردازنده می‌توان بجای جستجوی دودویی از جستجوی pتایی استفاده نموده. بدین روش که در هر مرحله پردازنده مبصر از فهرست مورد جستجو p اندازه را انتخاب کرده و به هر یک از پردازنده یکی از اندازه‌ها را ارسال کند(پردازنده مبصر نیز به بررسی وجود پاسخ برای یکی از این p اندازه انتخاب شده می‌پردازند)، سایر پردازنده‌ها پس از بررسی وجود پاسخ

نتیجه را به پردازنده مبصر اعلام کنند(پردازنده مبصر خود نیز به بررسی وجود پاسخ برای یکی از اندازه‌های تداخل می‌پردازد). سپس پردازنده مبصر با توجه به نتایج اعلام شده محدوده جدیدی از فهرست اندازه‌های تداخل را که جواب ممکن است در آن باشد را در نظر گرفته و مجددا p اندازه را انتخاب و به سایر پردازنده‌ها ارسال می‌کند. اگر تعداد اندازه‌ها موجود درفهرست مورد بررسی m تا باشد آنگاه عناصر انتخاب شده دارای شماره‌های زیر خواهند بود.

$$\frac{m}{p+1},\frac{2m}{p+1},\frac{3m}{p+1},\ldots,\frac{i*m}{p+1},\ldots,\frac{(p-1)*m}{p+1},\frac{p*m}{p+1}$$

زمان انجام جستجو به این روش بهمراه زمانی که پردازنده‌ها صرف محاسبه وجود پاسخ برای اندازه مربوطه می‌کنند برابر $n\log_p^n$ خواهد شد.

۳-۳- تحلیل پیچیدگی الگوریتم موازی

مرتب سازی نقاط بصورت موازی و محاسبه اندازه‌های تداخل و همچنین مرتب‌سازی اندازه‌های تداخل بصورت موازی دارای پیچیدگی $O(n/p(\log n/p+2^{\log p})$ است. جستجوی اندازه‌تداخل بهینه نیز دارای پیچیدگی $n\log_p^n$ است. بنابراین پیچیدگی کل الگوریتم از درجه $O(n\log_p^n)$ خواهد. همچنین افزایش سرعت و کارایی االگوریتم موازی نسبت به الگوریتم غیر موازی بترتیب زیر است.

$$Speed-up(P)=\frac{T(1)}{T(p)}=\frac{\log_2^n}{\log_p^n}=\log_2^p$$

$$Efficiency(P)=\frac{T(1)}{pT(p)}=\frac{\log_2^n}{p\log_p^n}=\frac{\log_2^p}{p}$$

۴- نتیجه‌گیری

در این مقاله یک الگوریتم موازی برای مساله برچسب‌گذاری نقشه‌ها معرفی نمودیم. نقشه می‌تواند یک نقشه معمولی(نقشه یک کشور)، نمودار، گراف یا هر شکل دیگری که نیاز به برچسب‌گذاری دارد، باشد. با توجه به NP-hard بودن مساله، استفاده از این الگوریتم موازی، موجب حل سریعتر مساله می‌شود.

این الگوریتم، اولین الگوریتم موازی برای این مساله است و شاید بتوان آنرا بهتر نمود. گام بعدی در این مساله می‌تواند بهتر نمودن محاسبه اندازه‌های تداخل باشد. همچنین می‌توان برروی موازی‌سازی الگوریتم از طریق تقسیم محاسبه وجود اندازه برچسب بهینه کار نمود.

مراجع

[1] M. Formann, F. Wagner, “A Packing Problem with Applications to Lettering of Maps”. *Proceedings of the 7th Annual ACM Symposium on Computational Geometry*. pp. 281-288, 1991.

[2] H. Aonuma, H. Imai, Y.Kambayashi, “A visual system of placing characters appropriatly in multimedia map databases”. *Proceedings of the IFIP TC 2/WG 2.6 Working Conference on Visual Database Systems*, North Holland. pp. 525-546, 1989.

[3] K. Imai, T. Asano, “Efficient Algorithms for Geometric Graph Search Problems”. *SIAM J. Comput. 15*. pp. 478-494, 1986.

[4] F. Wagner, A. Wolff, “Map Labeling Heuristics: Provably Good and Practically Useful”. *Proceedings of the 11th Annual ACM Symposium on Computational Geometry*. pp. 109-118, 1995.

[5] M. Formann, “*Algorithms for Geometric Packing and Scaling Problems*”. Dissertation, Fachbereich Mathematik und Informatik, Freie Universität, Berlin, 1992.