یازدهمین کنفرانس بین المللی کامپیوتر انجمن کامپیوتر ایران، پژوهشگاه دانشهای بنیادی، پژوهشکده علوم کامپیوتر، ۴ تا ۶ بهمن ۱۳۸۴، تهران، ایران.
11th International CSI Computer Conference (CSICC'2006), School of Computer Science, IPM, Jan. 24-26, 2006, Tehran, Iran.

الگوریتمی موازی برای برچسب‌گذاری نقشه‌ها به‌صورت تقریبی

میثم توسلی
دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف
tavassoli@ce.sharif.edu

فرشاد رستم‌آبادی
دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف
پژوهشکده علوم‌کامپیوتر، پژوهشگاه دانش‌های بنیادی
rostamabadi@imp.ir

محمد قدسی
دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف
پژوهشکده علوم‌کامپیوتر، پژوهشگاه دانش‌های بنیادی
ghodsi@sharif.edu

چکیده: در این مقاله ابتدا به بررسی الگوریتم‌های تقریبی برای برچسب‌گذاری نقشه‌ها می‌پردازیم و سپس الگوریتم‌های موازی برای آن بیان می‌کنیم. برچسب‌گذاری نقشه‌ها صورت‌های مختلفی دارد که در اکثر موارد این مسأله جزء مسائل NP-Hard قرار می‌گیرد. در این مقاله پس از معرفی بعضی حالتهای مختلف این مسأله، ابتدا یک الگوریتم ترتیبی را که برای حالتی که برچسب‌ها ارتفاعی واحد دارند با زمان اجرای $O(n\log n)$ ضریب تقریبی برابر ۲ دارد، بیان می‌کنیم. پس از آن حالت توسعه یافته همین الگوریتم که با زمان اجرای $O(n\log n+n^{2k-1})$ ضریب تقریب $(1+1/k)$ را نتیجه می‌دهد بیان خواهیم کرد [1]. در ادامه دو الگوریتم موازی مختلف برای برچسب‌گذاری یکی در مدل CREW PRAM و دیگری برای مدل توری ارائه می‌دهیم. الگوریتم موازی ارائه شده برای برچسب‌گذاری در مدل PRAM با p پردازنده افزایش سرعتی برابر با $O(p)$ داشته که از لحاظ زمان اجرا و کار انجام شده و تعداد پردازنده نسبت به الگوریتم‌های ترتیبی شناخته شده برای این مسأله بهینه است. الگوریتم موازی روی توری نیز با $p=n$ پردازنده افزایش سرعتی به میزان $\Omega(\sqrt{n}\log n)$ دارد. در آخر روش پیاده‌سازی الگوریتم موازی برچسب‌گذاری را در محیط LAM/MPI شرح داده و نتایج عملی حاصل را بررسی می‌کنیم.

واژه‌های کلیدی: برچسب‌گذاری، هندسه محاسباتی، الگوریتم‌های موازی، الگوریتم‌های تقریبی.

۱ مقدمه

برچسب‌گذاری[1] نقشه‌ها به‌طور خودکار یکی از مهمترین مسائل در سیستم‌های اطلاعات جغرافیایی[2] و گرافیک کامپیوتری است، که توجه زیادی در سال‌های اخیر به آن شده است.

برچسب‌گذاری نقشه عبارت است از قرار دادن متن یا اطلاعات گرافیکی که به طورکلی با عنوان برچسب شناخته می‌شوند در کنار ویژگی‌های مشخصی از نقشه. این عوارض قالباً به صورت نقطه، خط یا یک محیط-بسته می‌باشند و برچسب‌ها با اشکال ساده هندسی از جمله مربع، دایره، مستطیل و ... تخمین زده می‌شوند.

نیاز اصلی در برچسب‌گذاری جدا بودن برچسب‌ها از یکدیگر است به طوری که با یکدیگر اشتراکی نداشته باشند. از این رو اکثر پژوهش‌های به عمل آمده در این زمینه مربوط به نحوه برچسب‌گذاری به صورتی است که بیشترین تعداد ویژگی‌ها در نقشه برچسب‌گذاری شده و اندازه برچسب‌ها نیز تا حد امکان به اندازه اصلی آنها نزدیک باشد. متاسفانه حتی در حالت‌های ساده‌ی این مسئله نشان داده شده که این مسئله جزء مسائل *NP-Hard* بوده[2] و معمولاً راه‌حل‌های تقریبی برای آنها ارائه شده‌است.

در عمل برای برچسب‌ها محدودیت‌های بیشتری در نظر گرفته می‌شود تا الگوریتم‌های برچسب‌گذاری کاراتر و راحت‌تر شوند. محدود کردن شکل برچسب‌ها به صورت مربع‌هایی با اندازه یکسان یک محدودیت می‌تواند باشد، چراکه در بسیاری از نقشه‌های تکنیکی تمام برچسب‌ها اندازه یکسانی دارند. مورد قابل توجه دیگر برچسب‌های با ارتفاع یکسان اما عرض متفاوت هستند. این مورد به ویژه زمانی که می‌خواهیم عوارض روی نقشه را نام‌گذاری کرده و این نام‌ها از فونت و اندازه یکسانی برخوردارند، ظاهر می‌شود.

در این مقاله فرض می‌کنیم برچسب‌ها به صورت مستطیل‌هایی با ارتفاع ثابت و طول‌های متفاوت هستند و می‌خواهیم حداکثر تعداد ممکن از این برچسب‌ها را بر روی نقشه قرار دهیم به طوری که با یکدیگر اشتراکی نداشته باشند. از آنجا که این مسأله NP-Hard است، دو راه حل تقریبی، یکی با ضریب تقریب ۲ و زمان اجرای $O(n\log n)$ و دیگری با ضریب تقریب $(1+1/k)$ و زمان اجرای $O(n\log n+n^{2k-1})$ که توسط پنکاج و سایرین معرفی شده است را بیان می‌کنیم. سپس دو الگوریتم موازی یکی برای موازی‌سازی این الگوریتم در مدل CREW PRAM و دیگری برای مدل توری ارائه می‌دهیم. الگوریتم اول بر روی CREW PRAM با $p=n^{1-x}$ پردازنده زمان اجرایی برابر $O(n^x\log n)$ خواهد داشت، که نشان می‌دهیم کارآیی آن $\Omega(1)$ بوده و از لحاظ زمان اجرا و کار انجام شده و تعداد پردازنده نسبت به الگوریتم‌های ترتیبی شناخته شده برای این

1 Map Labeling
2 GIS

مسأله بهینه است. الگوریتم دوم با $p = n$ پردازنده به شکل توری $\sqrt{n} \times \sqrt{n}$ زمان اجرایی برابر $O(\sqrt{n})$ خواهد داشت.

ابتدا در بخش ۲ به بررسی کارهای انجام شده در این زمینه می‌پردازیم. سپس در بخش ۳ مسأله را به طور دقیق تعریف کرده و در بخش ۴ دو الگوریتم ترتیبی برای برچسب‌گذاری معرفی می‌کنیم، که از آن‌ها برای طراحی الگوریتم موازی استفاده خواهیم کرد. در بخش ۵ الگوریتم موازی برچسب‌گذاری در مدل PRAM و در بخش ۶ الگوریتم موازی در مدل توری ارائه شده است. در آخر نحوه‌ی پیاده‌سازی موازی این الگوریتم در محیط LAM/MPI گفته شده و نتایج حاصل از این پیاده‌سازی را بررسی می‌کنیم.

۲ پژوهش‌های گذشته

در انجمن نقشه‌کشی فعالیت‌های زیادی در زمینه برچسب‌گذاری نقشه‌ها انجام شده است [2]. همچنین پژوهشگران در زمینه الگوریتم نیز پژوهش‌های زیادی در این زمینه انجام داده‌اند. *فرمن* و *واگنر*[3] در [3] برچسب‌گذاری نقاط را با برچسب‌های مربع‌شکل به این صورت بیان کرده‌اند که هدف اصلی برچسب‌گذاری کلیه نقاط با بیشترین سایر برچسب‌ها است. یکی از چهار گوشه‌ی این برچسب‌ها باید بر روی نقطه مورد نظر قرار گرفته باشد. این برچسب‌های مربع شکل در حقیقت نمایان نوشته یا اندازه‌ایست که باید برای هر نقطه در نظر گرفته شود. و هدف از اندازه بیشینه برای برچسب‌ها، یافتن بیشترین اندازه فونت برای برچسب‌گذاری است.

در صورتی که در مسئله هر چهار مکان مختلف برای قرار گرفتن برچسب اجازه داده شده باشد مسئله جزء مسائل *NP-Hard* خواهد بود. فرمن و واگنر در مقاله خود [3] یک الگوریتم با زمان اجرای $O(n \log n)$ ارائه داده‌اند که اندازه برچسب‌ها حداقل نصف اندازه برچسب‌ها در برچسب‌گذاری بهینه است. آنها همچنین ثابت کرده‌اند که برچسب‌گذاری با ظریب تقریب بهتر از 2 برای این مسئله موجود نمی‌باشد مگر آنکه $P=NP$ باشد. روشی که آنها به کار برده‌اند بدین صورت می‌باشد که برای هر چهار مکان مجاز برچسب‌گذاری هر نقطه چهار برچسب در نظر گرفته و اندازه کلیه برچسب‌های نقاط موجود در نقشه را هم‌زمان افزایش می‌دهند و برچسب‌هایی که تداخل دارند را حذف می‌کنند. برای کنترل وجود برچسب‌گذاری معتبر در میان برچسب‌های باقی مانده از روش $2-SAT$ استفاده می‌کنند. *Kucera* در مقاله خود [4] یک الگوریتم *Super-polynomial* برای همین مسئله ارائه کرده که برای مجموعه‌های با سایز تقریبی کمتر از 100 مناسب می‌باشد.

واگنر و *ولف*[4] در [5] نشان دادند که روش فرمن و واگنر به ندرت نتیجه‌ای بهتر از نصف پاسخ بهینه می‌دهد، لذا پیاده‌سازی‌های مختلفی را ارائه کردند تا در عمل نتایج بهتری در مورد اندازه برچسب‌ها بگیرند.

دانش‌پژوه و قدسی نیز روشی برای اجرای موازی این الگوریتم بیان کردند که با p پردازنده افزایش سرعتی برابر با $\log_2^p$ دارد [6].

Doddi در [7] برچسب‌گذاری را به‌صورت عمومی‌تر از جمله به‌صورت دایره، بیضی و چندضلعی در نظر گرفت و راه‌حل‌های تقریبی برای تقریب اندازه برچسب‌ها ارائه کرد.

پنکاج[5] و دیگران در [1] روشی را برای برچسب‌گذاری نقاط با ارتفاع ثابت با تقریب $(1+1/k)$ و زمان اجرای $O(n \log n + n^{2k-1})$ ارائه کردند. الگوریتم تقریبی ارائه شده توسط آنها در بخش‌های بعد شرح داده شده است، و الگوریتم‌های موازی ارائه شده در این مقاله نیز بر اساس همین الگوریتم طراحی شده‌اند.

پوون[6] و دیگران نیز در مقاله خود [8] برای برچسب‌گذاری با ارتفاع واحد مجموعاً ۹ حالت بر اساس نحوه قرار گرفتن برچسب بر روی نقطه در نظر گرفته و برای هر حالت یک راه‌حل تقریبی معرفی کرده‌اند.

رستم‌آبادی و قدسی در مقاله خود [9] حالت کلی‌تری از این مسأله را در نظر گرفتند که برای هر عارضه درون نقشه (نقطه) یکسری مکان کاندید را تعریف کرده که برای هر مکان یکسری برچسب می‌تواند قرار گیرد. در این حالت آن‌ها راه‌حل ساده‌ای با زمان اجرای $O(n \log n)$ معرفی کردند که ضریب تقریبی برابر ۳ دارد.

۳ تعریف مسأله

در این مقاله فرض می‌کنیم برچسب‌ها به صورت مستطیل‌هایی با ارتفاع ثابت و طول‌های متفاوت هستند و می‌خواهیم حداکثر تعداد ممکن از این برچسب‌ها را بر روی نقشه قرار دهیم به طوری که با یکدیگر اشتراکی نداشته باشند.

تعریف ۱: هر برچسب r_i را با چهار مقدار x_i^r، x_i^l، y_i^u و y_i^d که به ترتیب عبارتند از مختصات x راست و چپ و مختصات y بالا و پایین، مشخص می‌کنیم. $\bar{y}_i = \lceil y_i^d \rceil$ را به عنوان سقف مختصات y یال پایین مستطیل تعریف می‌کنیم. هدف از این‌کار دسته‌بندی برچسب‌ها بر اساس مقدار $\bar{y}_i$ است.

به عبارت دقیق‌تر $P = \{p_1, p_2, ..., p_n\}$ مجموعه‌ای از n نقطه درون نقشه می‌باشد، که به ازای هر نقطه p_i یک برچسب r_i و یک مجموعه π_i از نقاط علامت‌گذاری شده بر روی محیط r_i داریم. به عنوان نمونه انتخاب‌های مختلف r_i می‌تواند دو سر یال سمت چپ یا چهار رأس این برچسب یا همان چهار رأس به اضافه‌ی نقاط وسط هر یال برچسب باشد. به بیان ساده‌تر تنها محدودیت موجود برای برچسب‌ها ارتفاع یکسان کلیه برچسب‌های موجود در نقشه است، هر نقطه در نقشه می‌تواند چندین برچسب کاندید داشته باشد اما تمامی برچسب‌های کاندید مربوط به یک نقطه باید حداقل در همان نقطه با یکدیگر اشتراک داشته باشند. (این محدودیت از آنجا ناشی شده که برای یک

3 Formann and Wagner
4 Wagner and Wolff

5 Pankaj
6 Poon

نقطه درون نقشه نتوان دو برچسب از میان برچسب‌های کاندیدش انتخاب کرد).

مجموعه‌ی S_i را به عنوان مجموعه‌ی مکان‌های کاندید برای قرارگرفتن برچسب r_i و $S=\bigcup_{i=1}^{n} S_i$ را به عنوان اجتماع S_i ها تعریف می‌کنیم. مسأله‌ی برچسب‌گذاری در حقیقت همان مسأله‌ی پیدا کردن بزرگترین زیر مجموعه‌ی S از برچسب‌های دو به دو متمایز است. از آنجایی که کلیه مستطیل‌های مجموعه‌ی S_i در p_i باهم تلاقی دارند، هر نقطه‌ی p_i از نقشه حداکثر یک برچسب از مجموعه‌ی S_i می‌گیرد. محاسبه‌ی یک مجموعه‌ی مستقل از مستطیل‌ها به عنوان یک مسأله‌ی NP-Hard شناخته شده است [10, 11]. بنابراین الگوریتم دقیق با زمان اجرای چندجمله‌ای برای آن وجود ندارد مگر $P=NP$ باشد. به همین دلیل معمولاً از الگوریتم‌های تقریبی برای حل آن استفاده می‌شود. یک الگوریتم $\varepsilon-approximation$ برای محاسبه‌ی مجموعه‌ی مستقل از مستطیل‌ها، الگوریتمی است که برای ($\varepsilon>1$) مجموعه‌ی مستقلی با اندازه‌ی حداقل γ/ε برگرداند. که γ سایز ماکسیمم مجموعه‌ی مستقل در S است.

۴ الگوریتم‌های ترتیبی

در این بخش دو الگوریتم ترتیبی که توسط پنکاج و دیگران [1] برای برچسب‌گذاری نقشه‌ها ارائه شده را معرفی می‌کنیم. در قسمت اول الگوریتمی با پیچیدگی زمانی $O(n\log n)$ و ضریب تقریب ۲ و در قسمت دوم الگوریتمی با زمان اجرای $O(n\log n+n^{2k-1})$ و ضریب تقریب $(1+1/k)$ آورده شده است.

۴-۱ الگوریتم تقریبی با ضریب تقریب ۲

برای یافتن یک مجموعه‌ی مستقل از مستطیل‌ها در R در حالتی که ارتفاع این مستطیل‌ها (برچسب‌ها) ثابت است پنکاج و دیگران در [1] یک الگوریتم تقریبی با ضریب تقریب ۲ و زمان اجرای $O(n\log n)$ به این ترتیب معرفی کرده‌اند.

مجموعه‌ی R را که شامل n مستطیل با ارتفاع واحد است در نظر بگیرید. m خط افقی $(m\le n)$ $l_1,l_2,...,l_m$ را به این ترتیب رسم می‌کنیم:

- فاصله‌ی بین دو خط اکیداً بیشتر از یک است.
- هر خط حداقل با یک مستطیل تقاطع دارد.
- هر مستطیل با یک خط تقاطع دارد.

توجه کنید که شرط اول سبب می‌شود هیچ مستطیلی با دو خط تقاطع نداشته باشد. این خطوط مجموعه‌ی R را به زیر مجموعه‌های $R_1,R_2,...,R_m$ افراز می‌کنند، به این ترتیب که R_i شامل مستطیل‌هایی است که با خط l_i تقاطع دارند.

در مرحله‌ی بعد به ازای هر مجموعه‌ی R_i بزرگترین مجموعه‌ی مستقل M_i را محاسبه می‌کنیم. به این ترتیب که ابتدا مستطیل‌های مجموعه‌ی R_i را بر اساس x^r آنها مرتب کرده، سپس با یک الگوریتم حریصانه کافیست مستطیل با کمترین x^r را انتخاب و مستطیل‌هایی که با آن تقاطع دارند را حذف کنیم و این کار را تا آخرین مستطیل ادامه دهیم. این مرحله به ازای هر مجموعه‌ی R_i زمانی برابر $O(|R_i|\log|R_i|)$ لازم دارد. نکته قابل توجه اینست که مستطیل‌های مجموعه‌ی R_i فقط با مجموعه‌های R_{i-1} و R_{i+1} ممکن است تداخل داشته باشند. بنابراین با محاسبه‌ی دو مجموعه‌ی $M^o=\{M_1\cup M_3\cup...\cup M_{2\lceil m/2\rceil-1}\}$ و $M^e=\{M_2\cup M_4\cup...\cup M_{2\lceil m/2\rceil}\}$ دو مجموعه از مستطیل‌های مستقل داریم. بنابراین یکی از این مجموعه‌ها حتماً سایزی بیشتر از $\gamma/2$ دارد و با انتخاب آن الگوریتمی با ضریب تقریب ۲ خواهیم داشت. زمان اجرای این الگوریتم $O(n\log n)$ خواهد بود.

۴-۲ الگوریتم تقریبی با ضریب تقریب $(1+\frac{1}{k})$

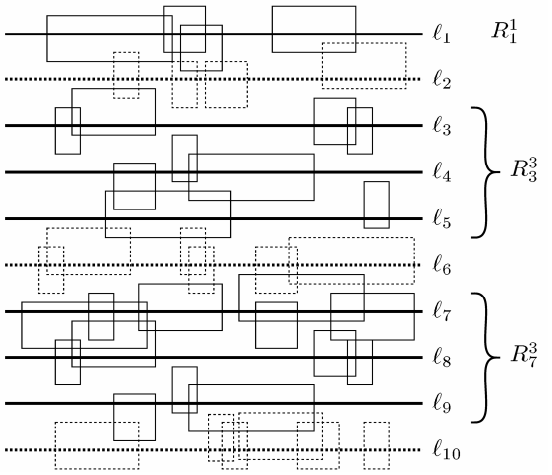
حال در این قسمت الگوریتمی شبیه آنچه در قسمت قبل بیان شد ارائه می‌کنیم، با این تفاوت که ضریب تقریب آن بهتر شده و به $(1+1/k)$ می‌رسد. البته این کاهش ضریب تقریب با افزایش زمان اجرا به میزان $O(n\log n+n^{2k-1})$ خواهد بود. ایده‌ی اصلی همان رسم خطوط $l_1,l_2,...,l_m$ با همان مشخصات قبلی است. اما این‌بار از یک الگوریتم پویا برای پیدا کردن مجموعه‌ی مستقل برای مستطیل‌های متقاطع با k خط پشت سرهم استفاده می‌کنیم. مجموعه‌ی R_i^k را به صورت $R_i^k=R_i\cup R_{i+1}...\cup R_{i+k-1}$ (R_i^k مجموعه‌ی مستطیل‌هایی است که با یکی از خطوط $\{l_i,l_{i+1},...,l_{i+k-1}\}$ تقاطع دارند) تعریف می‌کنیم. از این به بعد به R_i^k زیر گروه می‌گوییم. حال $k+1$ گروه $G_1,...,G_{k+1}$ (شکل ۱) را به این صورت تعریف می‌کنیم:

$$G_j=R_1^{j-1}\cup\bigcup_{i\ge 0}R^k_{i(k+1)+j}=R\setminus\bigcup_{i\ge 0}R_{i(k+1)+j}$$

به عبارت دیگر گروه G_j از روی R با حذف مستطیل‌هایی که با شروع از خط l_j با $k+1$ امین خطوط تقاطع دارند، ساخته می‌شود.

نکته‌ی قابل توجه این است که مستطیل‌های درون هر دو زیرگروه پشت سرهم از هر گروه با یکدیگر تقاطع ندارند، چرا که برای مثال خط l_{k+1} زیر گروه‌های R_1^k و R_{k+2}^k را در گروه G_1 از هم جدا می‌کند. بنابراین در هر گروه G_j مستطیل‌های هر زیرگروه از کلیه مستطیل‌های زیرگروه‌های دیگر این گروه مستقل است. از این رو با محاسبه‌ی مجموعه‌های مستقل برای هر زیرگروه و اجتماع آنها یک مجموعه‌ی مستقل برای کل گروه خواهیم داشت. در آخر توجه کنید که هر گروه با حذف حداکثر $\lceil m/(k+1)\rceil$ خط و مستطیل‌های متقاطع با آنها بوجود می‌آید. و این مستطیل‌ها در هر دو گروه با یکدیگر متفاوت هستند. لذا با محاسبه‌ی مجموعه‌های مستقل برای هر گروه و انتخاب

بزرگترین آنها با توجه با اصل لانه کبوتری حداکثر $\gamma/(k+1)$ مستطیل را از دست داده‌ایم، و الگوریتمی با ضریب تقریب $(1+1/k)$ خواهیم داشت.



شکل ۱: گروه G_2 برای k=3

محاسبه‌ی بزرگترین مجموعه‌ی مستقل برای هر زیرگروه با یک الگوریتم پویا در زمان $O(n_i^{2k-1})$ که n_i تعداد مستطیل‌های درون زیرگروه i ام است، امکان‌پذیر است. در نتیجه کل الگوریتم زمان اجرایی برابر $O(n\log n + n^{2k-1})$ دارد.

۵ مدل CREW PRAM

در این بخش روشی برای موازی‌سازی الگوریتم‌های ترتیبی معرفی شده در بخش قبل با استفاده از مدل پردازشی CREW PRAM ارائه خواهیم کرد. تعداد پردازنده‌های استفاده شده در این مدل $p<\sqrt{n}$ است و هدف افزایش سرعت به میزان p است (تسریع خطی).

۵-۱ خواص برچسب‌گذاری در محیط PRAM

اولین قدم در الگوریتم‌های موازی معمولاً تقسیم داده‌ها میان پردازنده‌های مختلف است. در اینجا نیز قصد داریم هر پردازنده برچسب‌گذاری بخشی از نقشه را انجام دهد. به طوری که تعداد برچسب‌های اختصاص یافته به تمام پردازنده‌ها تقریباً برابر باشند. در الگوریتم موازی هم از همان خطوط $l_1,l_2,...,l_m$ و مجموعه‌های $R_1,R_2,...,R_m$ استفاده کرده و هر چند خط متوالی به یک پردازنده اختصاص می‌یابد. به همین منظور نیاز داریم برچسب‌های هر مجموعه پشت سر هم، و به همان ترتیب مجموعه‌ها، روی حافظه‌ی مشترک PRAM قرارگرفته باشند. بدین معنی که اگر $i<j$ آنگاه برچسب‌های مجموعه‌ی R_i قبل از R_j قرار بگیرند. همچنین برچسب‌های هر مجموعه‌ی R_i بر اساس شان روی PRAM مرتب شده‌باشند.

لم ۱: *بر روی CREW PRAM با تعداد $p=n^{1-x}$ پردازنده در زمان $O(n^x\log n)$ می‌توان برچسب‌ها را به گونه‌ای مرتب کرد که به ازای هر دو برچسب i و j اگر $(\bar{y}_i<\bar{y}_j)\,or\,(\bar{y}_i=\bar{y}_j\ and\ x_i\le x_j)$ باشد آنگاه برچسب i قبل از برچسب j قرار گرفته باشد.*

اثبات: برای اینکار از یک الگوریتم موازی مرتب سازی بر روی PRAM استفاده می‌کنیم. یکی از الگوریتم‌های ساده و کارا برای مرتب سازی بر روی PRAM الگوریتم تصادفی است که در [12] آمده.

۱) پردازنده‌ی j، $0\le j<p$، تعداد $|S|/p^2$ نمونه‌ی تصادفی از $|S|/p$ عنصر مربوط به خود را انتخاب و در مکان مربوطه در لیست T با اندازه‌ی $|S|/p$ ذخیره می‌کند.
۲) پردازنده‌ی 0 لیست T را مرتب می‌کند.
۳) پردازنده‌ی j ، $0\le j<p$، عناصر مربوط به خود را که بین m_i و m_{i+1} هستند را در زیر لیست $T^{(i)}$ ذخیره می‌کند.
۴) پردازنده‌ی j ، $0\le j<p$، زیر لیست $T^{(i)}$ را مرتب می‌کند.

کافی است در هنگام مقایسه برچسب‌ها با یکدیگر شرط $(\bar{y}_i<\bar{y}_j)\,or\,(\bar{y}_i=\bar{y}_j\ and\ x_i\le x_j)$ را برای یافتن ترتیب برچسب‌ها اعمال کنیم. زمان اجرای الگوریتم مرتب سازی به این ترتیب است: زمان اجرای مرحله‌ی دوم $O((n/p)\log(n/p))$ است. در مرحله‌ی سوم نیز n/p تا ***diminished prefix sum*** که هر کدام $O(\log(n/p))$ زمان لازم دارد انجام می‌شود. در مرحله‌ی آخر هم به طور میانگین اگر اندازه‌ی مجموعه‌های $T^{(i)}$ تقریباً یکسان باشند $O((n/p)\log(n/p))$ خواهد بود. □

تعریف: به ازای هر پردازنده‌ی i، $(0\le i<p)$ دو مقدار $M_i^o=\sum_{j=0}^{m_i/2}M_{2j+1}$ و $M_i^e=\sum_{j=0}^{m_i/2}M_{2j}$ را به عنوان مجموع بزرگترین زیرمجموعه‌های مستقل R_j (به ترتیب زمانی که j فرد یا زوج باشد) که توسط پردازنده‌ی i ام پردازش می‌شوند تعریف می‌کنیم (m_i تعداد مجموعه‌های است که توسط پردازنده‌ی i ام پردازش می‌شود).

لم ۲: *محاسبه‌ی دو مقدار M_i^o و M_i^e توسط پردازنده‌ی i ام به طور میانگین با زمان اجرای $O(n/p)$ قابل انجام است.*

اثبات: هر پردازنده باید دو مقدار M_i^o و M_i^e مربوط به n/p برچسب متعلق به خود را محاسبه نماید. برای این کار پردازنده i ام از برچسب $(n/p)\times i$ ام شروع به پیمایش لیست می‌نماید. برای یافتن هر مجموعه R_j هر دو بر چسب متوالی که $\bar{y}$ مختلفی دارند ابتدا یا انتهای یک مجموعه را تشکیل می‌دهند. این بدین دلیل است که اندازه برچسب‌ها واحد بوده در نتیجه اگر کلیه خطوط l_i را به ازای $\bar{y}$ های مختلف رسم کنیم کافی‌است اندازه برچسب‌ها را $1-\varepsilon$ فرض کرده، یعنی دو برچسب اگر از روی یال‌های افقی بر هم مماس شدند را مجزا در نظر بگیریم، تا سه شرط ذکر شده در بخش ۴-۱ برای خطوط l_i برقرار شوند.

از آنجا که یافتن M_j ها به صورت ترتیبی انجام می‌شود هر مجموعه R_j باید فقط توسط یک پردازنده پردازش شود. به همین خاطر هر پردازنده پس از شروع از برچسب $(n/p)\times i$ام تا رسیدن به ابتدای اولین مجموعه‌ی R_j کاری انجام نداده و در انتها هم پس از رسیدن به برچسب $(n/p)\times(i+1)-1$ام تا رسیدن به انتهای مجموعه‌ی R_j فعلی به پردازش خود ادامه می‌دهد. از آنجایی که تعداد برچسب‌های پردازش شده توسط هر پردازنده به طور میانگین n/p می‌باشد، زمان اجرای لازم برای محاسبه‌ی دو مقدار M_i^o و M_i^e برای هر پردازنده $O(n/p)$ می‌باشد. □

لم ۳: *اگر مجموعه‌ی R_i شامل n_i مستطیل با ارتفاع یکسان باشد که همگی با یک خط افقی تقاطع داشته باشند، در صورتی که مستطیل‌ها بر اساس مختصات x سمت راستشان مرتب شده باشند در زمان $O(n_i)$ می‌توان بزرگترین زیرمجموعه‌ی R_i که هیچ دو مستطیلی با یکدیگر اشتراک نداشته باشند را پیدا کرد* [13]. □

لم ۴: *محاسبه‌ی دو مجموعه‌ی M^o و M^e در PRAM از روی مجموعه‌های M_i^o و M_i^e، $(0\le i<p)$ در زمان $O(\log p)$ امکان‌پذیر است.*

اثبات: برای محاسبه‌ی سایز دو مجموعه‌ی M^o و M^e پردازنده‌ها پس از اتمام محاسبه‌ی M_i^o و M_i^e مربوط به خود یک *diminished prefix sum* روی تعداد مجموعه‌هایی که پردازش کرده‌اند انجام می‌دهند. به این ترتیب هر پردازنده‌ی i فرد یا زوج بودن مجموعه‌های M_i^o و M_i^e محاسبه کرده‌ی خود را در میان کل مجموعه‌های R_j موجود در نقشه تشخیص می‌دهد. و در صورت لزوم این دو مقدار را با یکدیگر تعویض می‌کند. پس از آن با یک بار انجام prefix sum روی دو مقدار بدست آمده توسط هر پردازنده سایز دو مجموعه‌ی M^o و M^e در کل بدست می‌آید. که با مقایسه‌ی آن توسط پردازنده‌ها هر پردازنده می‌داند که کدامیک از مجموعه‌های M_i^o و M_i^e خود را به عنوان خروجی نمایش دهد. زمان اجرای این کار $O(\log p)$ که همان زمان لازم برای انجام prefix sum است، می‌باشد. □

۲-۵ الگوریتم PRAM

الگوریتم برچسب‌گذاری در مدل CREW PRAM به ترتیب زیر خواهد بود:

۱) ابتدا برچسب‌ها را بر اساس مقادیر x^r و $\bar{y}$ آنها به گونه‌ای مرتب می‌کنیم که به ازای هر دو برچسب i و jاگر $(\bar{y}_i<\bar{y}_j)\ or\ (\bar{y}_i=\bar{y}_j\ and\ x_i\le x_j)$ آنگاه برچسب i قبل از برچسب j قرار گرفته باشد. (لم ۱)

۲) برچسب‌ها را به تعداد مساوی بین پردازنده‌ها تقسیم می‌کنیم. پردازنده‌ی i ام از میان برچسب‌های اختصاص یافته به آن دو مجموعه‌ی M_i^o و M_i^e را محاسبه می‌نماید. (لم ۲)

۳) با یک عمل *diminished prefix sum* روی تعداد مجموعه‌ی پردازش شده توسط هر پردازنده، پردازنده‌ها ترتیب زوج و فرد بودن مجموعه‌های پردازش کرده‌ی خود را در میان کل مجموعه‌ها تشخیص می‌دهند. (لم ۴)

۴) با یک عمل *prefix sum* روی سایز مجموعه‌های M_i^o و M_i^e $(0\le i<p)$ کلیه پردازنده‌ها تشخیص می‌دهند که مجموعه‌های فرد به عنوان جواب تقریبی انتخاب شده یا مجموعه‌های زوج. (لم ۴)

قضیه ۱: *مجموعه‌ی R شامل n برچسب با ارتفاع واحد می‌باشد. با تعداد $p=n^{1-x}$ پردازنده $(0<x<0.5)$ در زمان $O(n^x\log n)$ می‌توان برچسب‌گذاری صحیحی با ضریب تقریب ۲ برای R پیدا کرد.*

این نکته نیز قابل ذکر است که شرط $x<0.5$ به خاطر الگوریتم مرتب‌سازی تصادفی استفاده شده در مرحله‌ی اول الگوریتم بوجود آمده. چراکه حداکثر تعداد پردازنده برای این الگوریتم مرتب‌سازی $\sqrt{n}$ می‌باشد. در آخر میزان تسریع، کارایی و مقدار کار انجام شده در الگوریتم موازی را با فرمول‌های زیر محاسبه می‌نماییم:

$$SpeedUp(n,p)=\frac{\Omega(n\log n)}{O(n^x\log n)}=\Omega(n^{1-x})=\Omega(p)$$

$$Efficiency=SpeedUp/p=\Omega(1)$$

$$Work(n,P)=pT(n,p)=\Theta(n^{1-x})O(n^x\log n)=O(n\log n)$$

همانطور که مشاهده می‌شود تسریع الگوریتم $\Omega(p)$ و کارایی آن $\Omega(1)$ است، که نسبت به بهترین الگوریتم‌های ترتیبی شناخته شده برای این مسأله، بهینه است.

حال می‌خواهیم نشان دهیم چگونه در مدل PRAM به ضریب تقریب $(1+1/k)$ دست یابیم. این امر نیز به سادگی امکان‌پذیر است. کافی‌ست هر پردازنده به جای محاسبه‌ی دو مجموعه‌ی M_i^o و M_i^e، $k+1$ مجموعه‌ی $M_i^0,M_i^1,...,M_i^k$ را با همان الگوریتم پویا که در روش ترتیبی به کار رفت محاسبه نماید. این بار هر پردازنده در پایان کارش، کلیه زیر گروه‌هایی که آخرین مجموعه‌ای که مسئول پردازش آن است در آن قرار می‌گیرد را باید محاسبه نماید. در آخر با همان عمل *diminished prefix sum* ترتیب واقعی مجموعه‌های M_i^j را که محاسبه کرده در کل نقشه بدست آورده و در آخر با یک عمل *prefix sum* سایز واقعی هر گروه را بدست می‌آورد.

۶ الگوریتم موازی برچسب‌گذاری با استفاده از ساختار توری

روش دیگری که برای موازی سازی الگوریتم تقریبی برچسب‌گذاری در اینجا معرفی می‌شود استفاده از ساختار توری با پردازنده‌های کوچک است. تعداد پردازنده‌های به کار رفته در این الگوریتم $\sqrt{n}\times\sqrt{n}$ خواهد بود، که البته می‌توان این تعداد را کاهش داده اما در عوض باید قدرت پردازشی و حافظه‌ی هر پردازنده را افزایش دهیم. در این بخش ابتدا به پاره‌ای از جزئیات لازم برای اجرای الگوریتم روی مدل توری اشاره می‌کنیم و پس از آن الگوریتم مربوطه را ارائه می‌دهیم.

۶-۱ خواص برچسب‌گذاری در مدل توری

در این روش مانند مدل PRAM از الگوریتمی شبیه آنچه در بخش قبل ارائه شد، استفاده می‌کنیم. اما در مدل توری به جای آنکه برچسب‌ها روی حافظه‌ای مشترک مرتب شوند، هر برچسب در اختیار یک پردازنده قرار می‌گیرد. در نتیجه تعداد پردازنده‌های مورد نیاز برابر تعداد برچسب‌های موجود در نقشه خواهد بود.

لم ۵: *بر روی توری با تعداد $p=n$ پردازنده در زمان $O(\sqrt{n})$ می‌توان برچسب‌ها را به گونه‌ای مرتب کرد که به ازای هر دو برچسب i و j اگر $(\bar{y}_i<\bar{y}_j)\,or\,(\bar{y}_i=\bar{y}_j\ and\ x_i\le x_j)$ آنگاه برچسب i قبل از برچسب j قرار گرفته باشد (ترتیب بر روی توری به صورت مارپیچی خواهد بود).*

اثبات: برای مرتب سازی برچسب‌ها بر روی توری از یکی از الگوریتم‌های مرتب‌سازی روی توری از جمله *shearsort* با زمان اجرای $O(\sqrt{n}\log n)$ [12] یا روش بازگشتی با زمان اجرای $O(\sqrt{n})$ [12] استفاده می‌کنیم. تنها نکته باقی‌مانده اینست که در هنگام مقایسه دو برچسب پردازنده‌ها دو مقدار $\bar{y}$ و x^r برچسب مربوط به خود را برای یکدیگر ارسال می‌کنند. در صورتی که $(\bar{y}_i<\bar{y}_j)\,or\,(\bar{y}_i=\bar{y}_j\ and\ x_i\le x_j)$ آنگاه برچسب i قبل از برچسب j خواهد بود و گرنه j قبل از i قرار می‌گیرد. □

لم ۶: *بر روی توری با تعداد $p=n$ پردازنده، در صورت مرتب بودن برچسب‌ها بر اساس $\bar{y}$ و x^r آنها، مجموعه‌های R_i را می‌توان در $3\sqrt{p}-2$ مرحله به گونه‌ای مشخص کرد که کلیه برچسب‌های مجموعه‌ی R_i اندیسی برابر i دریافت کنند.*

اثبات: برای یافتن مجموعه‌های R_i پس از آنکه برچسب‌ها روی توری مرتب شدند، کافی‌است اولین پردازنده و آخرین پردازنده‌ای که برچسب‌های مجموعه‌ی R_i بین این دو پردازنده قرار می‌گیرند را مشخص نماییم. بدین منظور پردازنده‌ی j ام $(0\le j<n)$ مقدار $\bar{y}_j$ را برای پردازنده‌ی $j-1$ ام و $j+1$ ام ارسال می‌کند. اگر $\bar{y}_{j-1}\ne\bar{y}_j$ آنگاه j به عنوان اولین پردازنده و اگر $\bar{y}_{j+1}\ne\bar{y}_j$ آنگاه j به عنوان آخرین پردازنده‌ی یک لیست انتخاب می‌شود. حال به کلیه پردازنده‌های ابتدای هر لیست اندیس ۱ و به سایر پردازنده‌ها اندیس صفر نسبت می‌دهیم. پس از آن با یک عمل *prefix sum* روی اندیس‌های نسبت داده شده، هر پردازنده اندیس واقعی مجموعه‌ی R_i را که به آن تعلق دارد، دریافت می‌کند. عمل *prefix sum* هم با یکبار انجام آن روی سطرها و سپس محاسبه‌ی *diminished prefix sum* روی ستون آخر و اعلام نتیجه‌ی آن برای کلیه عناصر هر سطر، برای انجام عمل جمع با مقدار *prefix sum* بدست آمده در مرحله‌ی قبل، با $3\sqrt{P}-3$ مرحله قابل اجراست. اگر یافتن عناصر ابتدای هر مجموعه هم در یک مرحله انجام شود در کل $3\sqrt{p}-2$ مرحله کار انجام داده‌ایم. □

لم ۷: *اگر مجموعه‌ی R_i شامل $|R_i|$ مستطیل با ارتفاع یکسان باشد که همگی با یک خط افقی تقاطع داشته باشند، در صورتی که مستطیل‌ها بر اساس مختصات x سمت راستشان مرتب شده بر روی آرایه‌ای از پردازنده‌ها باشند، در زمان $O(|R_i|)$ می‌توان بزرگترین زیرمجموعه‌ی R_i که هیچ دو مستطیلی با یکدیگر اشتراک نداشته باشند را پیدا کرد.*

اثبات: همانند لم ۳ با توجه به اینکه هر پردازنده با پردازنده‌ی بعد از خود رابطه دارد، به سادگی پردازنده‌ی اول از لیست R_i شروع به پردازش کرده و دو مقدار x^r مربوط به برچسب خود و $|M_i|$ که در ابتدا صفر است را برای پردازنده‌ی بعد از خود ارسال می‌کند. هر پردازنده با مقایسه‌ی x^l خود با x^r دریافتی از پردازنده‌ی ماقبل خود اگر $x^r_{j-1}<x^l_j$ باشد برچسب خود را به مجموعه‌ی M_i اضافه و مقدار $|M_i|$ را یک واحد افزایش می‌دهد و مقادیر x^r خود و $|M_i|$ را برای پردازنده‌ی بعد از خود ارسال می‌کند. در غیر اینصورت همان مقادیر قبلی را ارسال خواهد کرد. این پروسه تا رسیدن به آخرین پردازنده‌ی هر لیست ادامه پیدا می‌کند. □

لم ۸: *محاسبه‌ی دو مقدار $|M^o|$ و $|M^e|$ در توری زمانی که هر پردازنده مقدار i و $|M_i|$ را دارد در $2\sqrt{p}-2$ مرحله قابل انجام است.*

اثبات: در صورتی که هر پردازنده دو مقدار $|M^o|$ و $|M^e|$ را در ابتدا برای خود صفر قرار دهد، و فقط پردازنده‌های انتهای هر لیست R_i بسته به فرد بودن یا زوج بودن اندیس i ، به ترتیب مقدار $|M^o|$ یا $|M^e|$ خود را برابر $|M_i|$ قرار داده و دیگری را صفر بگذارند، با یک عمل *semigroup* می‌توان جمع مقادیر $|M^o|$ و $|M^e|$ را بدست آورد. عمل *semigroup* هم در توری به سادگی با انجام یکبار چرخش عناصر روی هر سطر و یکبار چرخش روی هر ستون و انجام عمل جمع در هر نود، در $2\sqrt{p}-2$ مرحله قابل انجام است.

۲-۶ الگوریتم موازی روی مدل توری

الگوریتم برچسب‌گذاری بر روی توری به صورت زیر خواهد بود:

۱) ابتدا برچسب‌ها را بر اساس $\bar{y}$ آنها مرتب می‌کنیم (در حالت مساوی ملاک مرتب‌سازی x^r برچسب‌ها خواهد بود). (لم ۵)

۲) عناصر ابتدا و انتهای هر مجموعه‌ی R_i را مشخص و برای کلیه برچسب‌های مجموعه‌ی R_i اندیس i را محاسبه می‌نماییم. (لم ۶)

۳) مقدار $|M_i|$ را برای هر مجموعه‌ی R_i محاسبه می‌کنیم. (لم ۷)

۴) سایز دو مجموعه‌ی M^o و M^e را کلیه پردازنده‌ها با الگوریتم *semigroup* محاسبه کرده، و کلیه پردازنده‌ها با مقایسه‌ی این دو مقدار می‌فهمند که مجموعه‌های فرد به عنوان پاسخ نهایی انتخاب شده یا مجموعه‌های زوج. پردازنده‌هایی که اندیس مجموعه‌ای که متعلق به آن هستند از نظر زوج و فردی با مجموعه‌ی بزرگتر برابر است، در صورتی که برچسب‌شان به عنوان یکی از برچسب‌های مجموعه‌ی M_i مربوطه انتخاب شده باشد، پاسخ نهایی را تشکیل می‌دهند.

قضیه ۲: *اگر مجموعه‌ی R شامل n برچسب با ارتفاع واحد باشد، با تعداد $p=n$ پردازنده به شکل توری $\sqrt{P}\times\sqrt{P}$ در زمان $O(\sqrt{n})$ می‌توان برچسب‌گذاری صحیحی با ضریب تقریب ۲ برای R پیدا کرد.*

در آخر میزان تسریع و کارایی الگوریتم برچسب‌گذاری موازی بر روی توری را با فرمول‌های زیر محاسبه می‌نماییم:

$$SpeedUp(n,p)=\Omega(n\log n)/\mathrm{O}(\sqrt{n})=\Omega(\sqrt{n}\log n)$$

$$Efficiency=SpeedUp/\,p=\Omega(\sqrt{n}\log n)/n=\frac{\log n}{\sqrt{n}}$$

۷ پیاده‌سازی الگوریتم موازی در محیط LAM/MPI

برای پیاده‌سازی الگوریتم‌های ترتیبی و موازی برچسب‌گذاری در محیط LAM/MPI از همان الگوریتم دوم با ضریب تقریب $(1+1/k)$ استفاده می‌کنیم.

برای اینکار ابتدا باید شیوه‌ی بارگذاری داده‌ها بر روی پردازنده‌ها در حالت اولیه مشخص شود. به همین منظور ۲ روش مختلف را پیشنهاد می‌کنیم. در روش اول فرض می‌کنیم داده‌ها به صورت متمرکز در اختیار پردازنده اصلی قرار دارند. پردازنده اصلی ابتدا داده‌ها را خوانده و آنها را میان سایر پردازنده‌ها تقسیم می‌کند. در صورتی که برای این تقسیم از الگوریتم‌های مرتب سازی استفاده کنیم و عمل مرتب‌سازی را همان پردازنده‌ی اول انجام دهد با توجه به زمان زیادی که صرف مرتب‌سازی می‌شود برای k های کم مثلاً حالت $k=1$ زمان مرتب-سازی تقریباً با زمان یافتن برچسب‌گذاری نهایی یکسان شده و تسریع بسیار کم خواهد بود. به همین دلیل یا باید برای مرتب‌سازی هم، از الگوریتم‌های مرتب‌سازی موازی در محیط LAM/MPI استفاده کرد و یا روشی که ما پیشنهاد می‌کنیم استفاده از یک شیوه‌ی درهم سازی[۷] است. این کار نیز با دانستن مختصات حدودی محیط نقشه به راحتی قابل انجام است. کافی است تابع درهم‌سازی تعریف کنیم که بر اساس مختصات y آنها هر مستطیل را به یک پردازنده بدهد. به عبارت دیگر نقشه به بخشهای افقی تقسیم می‌شود و هر پردازنده مسئول یافتن برچسب‌گذاری بهینه در یک یا چند بخش می‌شود. تعداد این تقسیم بندی‌ها می‌تواند یا برابر با تعداد پردازنده‌ها باشد که در صورتی که تعداد برچسب‌ها در مناطقی از نقشه بسیار کم و در مناطق دیگر بسیار زیاد باشد احتمال توزیع نامتوازن برچسب‌ها میان پردازنده‌ها وجود دارد، یا از تعداد پردازنده‌ها بیشتر باشد و چند بخش در اختیار هر پردازنده قرار گیرد، که در این حالت احتمال توزیع نامتوازن برچسب‌ها کمتر می‌شود.

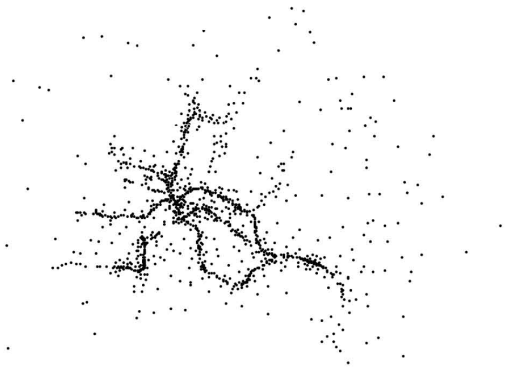
روش بعدی برای بارگذاری داده‌های برچسب‌ها میان پردازنده‌ها استفاده از شیوه‌ی نامتمرکز است. در این روش فرض می‌شود هر پردازنده اطلاعات یکسری از برچسب‌ها را در اختیار داشته و اطلاعات هر برچسب فقط در اختیار یک پردازنده است. حال می‌خواهیم برای کل برچسب‌ها یک برچسب‌گذاری صحیح پیدا کنیم. در این حالت نیز مانند حالت قبل از همان شیوه‌ی تقسیم بندی افقی و استفاده از یک تابع درهم ساز مناسب استفاده می‌کنیم. با این تفاوت که کلیه پردازنده‌ها باید از تابع درهم‌ساز یکسانی استفاده کرده و هر پردازنده در آخر ممکن است برای تعدادی پردازنده اطلاعات برچسب‌های مربوط به بخش آنها را ارسال کند.

بعد از مرحله‌ی بارگذاری اولیه نوبت به اجرای الگوریتم تقریبی می‌رسد. در اینجا نیز همانند آنچه در مدل PRAM گفته شد، هر پردازنده قسمت‌های مربوط به خود را پردازش کرده و در فاز آخر حاصل ماکسیمم مجموعه‌های مستقلی که برای هر گروه بدست آورده و تعداد مجموعه‌های موجود در هر قسمت را برای پردازنده مرکزی ارسال و این پردازنده با جمع بندی روی این مقادیر تصمیم می‌گیرد که بزرگترین اندازه برچسب‌گذاری مربوط به کدام مجموعه می‌شود.

برای تست این الگوریتم نیاز به داده‌هایی هست که از نظر نحوه‌ی تراکم برچسب‌ها بیشترین شباهت را به نقشه‌های واقعی داشته باشند. به عنوان مثال در شکل ۲ نقشه‌ای که عارضه‌های آن با یکسری نقاط نشان داده‌شده مشاهده می‌شود. همانطور که در این شکل پیداست با تولید نقاط کاملاً تصادفی نمی‌توان نقشه‌هایی که در عمل با آن مواجه هستیم را تولید کرد. لذا سعی شده در تولید داده‌ها تصادفی از شیوه‌هایی استفاده کنیم که داده‌ها به داده‌های موجود در دنیای واقعی نزدیک‌تر باشند. به عنوان مثال مکان‌هایی از نقشه را در ابتدا به طور تصادفی به عنوان مکان‌های پرتراکم انتخاب و برچسب‌ها را با احتمال بیشتر نزدیک این مکان‌ها قرار دهیم. و هر چه فاصله از این مکان‌ها بیشتر می‌شود احتمال قرار گرفتن برچسب در آن نواحی را کم می‌کنیم. همچنین در

[7] hashing

تولید اندازه برچسب‌ها نیز الگوریتمی استفاده کردیم که اکثر برچسب‌ها طولی بین ۱ تا ۳ داشته باشند، و تعداد برچسب‌های با طول بیشتر یا کمتر از این مقدار، کم باشد. به عنوان مثال برچسب‌هایی که طولی بیشتر از ۵ دارند بسیار کم هستند.



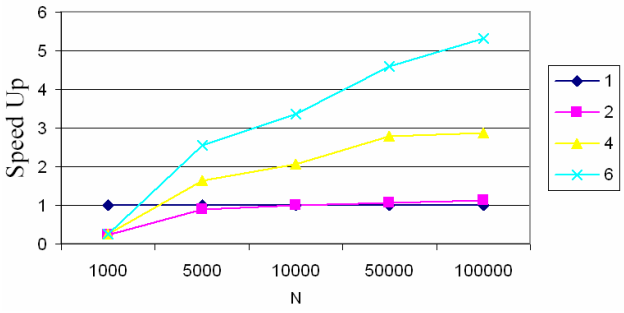
شکل ۲: نمونه‌ای از داده‌های واقعی

به این ترتیب با تولید داده‌های تصادفی مختلف و اجراهای متعدد الگوریتم روی این داده‌ها، با تعداد پردازنده‌ی مختلف سعی کردیم در عمل هم به افزایش سرعت خی دست‌یابیم. نتایج حاصل از پیاده‌سازی را در جدول ۱ و شکل ۳ می‌توان مشاهده کرد. این نتایج بر اساس میانگین چندین بار اجرای مختلف الگوریتم بر روی داده‌های شبه واقعی با تعداد پردازنده‌های متفاوت بدست آمده است.

جدول ۱: نتایج حاصل از اجرای موازی الگوریتم

افزایش سرعت		تعداد پردازنده ها			
		1	2	4	6
سایز ورودی	1000	1	0.233	0.257	0.262
	5000	1	0.894	1.647	2.543
	10000	1	0.997	2.054	3.354
	50000	1	1.057	2.789	4.586
	100000	1	1.13	2.876	5.314

همانطور که در نتایج پیاده‌سازی قابل مشاهده است، زمانی که تعداد برچسب‌ها بسیار کم است، سرعت اجرای الگوریتم حتی از حالت تک-پردازنده‌ای نیز کمتر است. اما با افزایش تعداد برچسب‌ها (سایز ورودی) و تعداد پردازنده‌ها کم کم به افزایش سرعتی نزدیک به حالت خطی می‌توان دست یافت.



شکل ۳: نمودار نتایج اجرای ترتیبی و موازی الگوریتم برچسب‌گذاری

۸ نتیجه‌گیری

در این مقاله الگوریتمهایی موازی برای مسأله‌ی برچسب‌گذاری در مدل‌های مختلف از جمله PRAM و توری ارائه کرده و نتایج عملی حاصل از پیاده‌سازی آن در محیط LAM/MPI را بررسی کردیم. با توجه به NP-Hard بودن مسأله‌ی برچسب‌گذاری در حالت‌هایی که بررسی شد الگوریتم‌های موازی در سرعت بخشیدن به اجرای مسأله به صورت موازی می‌توانند مؤثر واقع شوند.

تا به حال در مورد الگوریتم‌های موازی برای این مسأله تحقیقات زیادی انجام نشده لذا می‌توان حالت‌های مختلف دیگر از این مسأله را نیز با الگوریتم‌های متفاوت موازی کرد، تا زمان‌های اجرای بهتری بدست آوریم.

مراجع

[1] Pankaj K. Agarwal, Marc van Kreveld, and Subhash Suri. *Label placement by maximum independent set in rectangles,* In Proceedings of the 9th Canadian Conference on Computational Geometry (CCCG'97), pages 233-238, Kingston, Canada, 11-14 August 1997.

[2] J. Christensen, J. Marks and S. Shieber. *An empirical study of algorithms for point feature label placement,* ACM Trans, Graph., 1995, 14:202-232.

[3] M. Formann and F. Wagner, *A packing problem with applications to lettering of maps,* In Proc. 7th Annu. ACM Sympos. Comput. Geom. 1991, 281-288.

[4] L. Kucera, K. Mehlhorn, B. Preis, and E. Schwarzenecker, *Exact algorithms for a geometric packing problem,* In Proc, 10th Sympos, Theoret, Aspects Comput, Sci., volume 665 of Lecture Notes Comput, Sci., Springer-Verlag 1993, 317-322.

[5] F. Wagner and A. Wolff., *An efficient and effective approximation algorithm for the map labeling problem,* In Proc, 3rd Annu, European Sympos. Algorithms, volume 979 of Lecture Notes Comput. Sci., Springer-Verlag, 1995, 420-433.

[6] Sh. Daneshpajouh, M. Ghodsi, *Parallel Algorithm for Map Labeling Problem*, (In persian) in 2nd Conference on Information and Knowledge Technology (IKT 2005), Amir Kabir University of Technology, June 2005.

[7] S. Doddi, M. Marathe, A. Mirzaian, B. Moret, and B. Zhu, *Map labeling and its generalization,* In Proceedings of the 8th ACM-SIAM Sympos on Discrete Algorithms, 1997, 148-157.

[8] Sheung-Hung Poon, Chan-Su Shin, Tycho Strijk, and Alexander Wol. *Labeling points with weights,* In Proc. 17th European Workshop on Computational Geometry (CG'01), Berlin, 26-28 March 2001, 97-100.

[9] F. Rostamabadi and M. Ghodsi, *Unit Height k-Position Map Labeling,* 19th European Workshop on Computer Geometry, March 24-26, 2003.

[10] H. Imai and Ta. Asano, *Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane,* J. Algorithms, 1983, 4:310-323.

[11] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto, *Optimal packing and covering in the plane are NP-complete,* Inform. Process. Lett., 1981, 12(3):133-137.

[12] Behrooz Parhami, *Introduction to Parallel Processing*, Plenum Press 1999.

[13] TH Cormen, CE Leiserson, RL Rivest, C Stein, *Introduction to algorithms,* MIT Press, 2001.