مطالعات علوم کاربردی در مهندسی
دوره ۹، شماره ۲، تابستان ۱۴۰۲
صفحات ۳۰-۱۵

Online ISSN: ۲۴۷۶-۴۵۰۷
Print ISSN: ۲۵۳۸-۲۰۵۵
www.irijournals.com

ارائه روش نوین موازی‌سازی الگوریتم یافتن مجموعه غالب متصل در گراف با استفاده از کتابخانه OpenMp

ملیحه هاشمی[۱]، امید اخلاصی[۲]

استاد حق التدریس دانشگاه آزاد اسلامی واحد کهنوج
کارشناسی ارشد مهندسی کامپیوتر گرایش نرم افزار دانشگاه آزاد اسلامی واحد کرمان

چکیده

موازی کردن الگوریتم های مختلف به طور مداوم در حال تحول می باشد که محققان به عنوان یک تکنولوژی جدید با آن مواجه هستند. در دهه ی گذشته، مدل های جدیدی از الگوریتم ها، سخت افزارهای جدید برای اجرای موازی و چالش های جدید در حل مسائل پیچیده، باعث پیشرفت در این زمینه شده است. استفاده از روش های موازی سازی برای مقابله با برخی از موضوعات در حال رشد می باشد. مجموعه غالب متصل حداقلی یک مجموعه غالب متصل با حداقل گره‌های ممکن در شبکه است. یافتن مجموعه غالب متصل حداقلی برای گراف، یک مسئله NP-Hard است. بنابراین تاکنون الگوریتم بهینه‌ای برای یافتن این مجموعه غالب پیدا نشده است. این مسئله کاربردهای بسیاری در زمینه مسیریابی و رایانش‌های توزیع‌شده یا خوشه‌ای دارد. در این پژوهش یک نسخه موازی جدید از یافتن مجموعه غالب متصل در گراف ارائه خواهد شد که در بخش محاسبه زیر درخت‌های گراف دارای قابلیت اجرای موازی است و این بخش بر اساس کتابخانه موازی OpenMP اجرا می‌گردد که در آن هر یک از بخش های موازی تحت عنوان یک نخ تعریف شده و از قدرت اجرای موازی سیستم‌های چندپردازنده بهره می‌برد. این تحقیق با مدنظر قرار دادن کتابخانه موازی سازی OpenMp، در پی ارائه روشی جدید بوده که سرعت و دقت الگوریتم یافتن مجموعه غالب متصل در گراف را برای مسایل بهینه‌سازی با استفاده از این فنّاوری افزایش دهد. ایده اصلی در اینجا استفاده از نخ‌های چندگانه در شکل یک نخ واحد بوده که احتمال همگرایی برای به دست آوردن یک راه‌حل بهینه را افزایش می‌دهد. روش پیشنهادی شبیه سازی شده و بر روی چهار مجموعه دادهگراف مختلف آزمایش گردیده است. نتایج روش پیشنهادی به خوبی مشخص است با افزایش تعداد هسته‌های پردازنده مورد استفاده، کارایی افزایش می‌یابد و با افزایش تعداد واحدهای اجرایی موازی در کتابخانه OpenMP نیز شاهد افزایش در میزان تسریع به حالت یافتن مجموعه غالب متصل گراف در حالت سریال می‌باشیم. این امر کارایی روش پیشنهادی را نشان می‌دهد.

کلمات کلیدی: پردازش گراف، یافتن مجموعه غالب متصل، موازی سازی، کاهش زمان اجرا، کتابخانه OpenMP

مقدمه

موازی کردن الگوریتم‌های مختلف و بخصوص الگوریتم‌های گراف بدلیل پیچیدگی آن در صورت بزرگ شدن اندازه گراف، به طور مداوم در حال تحول می‌باشد که محققان به عنوان یک تکنولوژی جدید با آن مواجه هستند. در دهه‌ی گذشته، مدل‌های جدیدی از الگوریتم‌ها، سخت‌افزارهای جدید برای اجرای موازی و چالش‌های جدید در حل مسائل پیچیده، باعث پیشرفت در این زمینه شده است. این موضوعات شامل استفاده از مدل‌های کلاسیک موازی در پلتفرم‌های اخیر (مثل گرید، معماری‌های کلود) می‌باشد. با این حال، انتقال الگوریتم‌های موجود جدید به سخت‌افزار به عنوان یک هدف دنبال نمی‌شود بلکه محققان به دنبال موازی‌سازی بهینه الگوریتم‌های موجود هستند[۱].

گراف بدون جهت $G = (V, E)$ داده شده است. V مجموعه‌ای از رأس‌ها را نشان می‌دهد و E مجموعه‌ای از یال‌ها را مشخص می‌کند. مجموعه غالب[1] (DS) از گراف G یک زیرمجموعه از گره‌هاست به طوری که هر گره در گراف یا در زیرمجموعه یا مجاور به حداقل یک گره در زیرمجموعه قرار دارد[۲]. اگر زیرگروه القا شده توسط گره‌ها در یک DS متصل شود، مجموعه غالب را یک مجموعه غالب متصل گویند. مجموعه **غالب** برای گراف $G = (V, E)$، زیرمجموعه‌ای از گره‌هاست به گونه‌ای که هر گره یا درون این مجموعه است یا همسایه‌ای در این مجموعه دارد. مجموعه غالب متصل حداقلی یک مجموعه غالب متصل با حداقل گره‌های ممکن در شبکه است. تعداد گره‌های مجموعه غالب متصل حداقلی را عدد غلبه گراف می‌نامیم. یافتن مجموعه غالب متصل حداقلی برای گراف G و با عدد غلبه $Y(G)$ کوچک‌تر از K، یک مسئله NP-Hard است. بنابراین تاکنون الگوریتم بهینه‌ای برای یافتن این مجموعه غالب پیدا نشده است. این مسئله کاربردهای بسیاری در زمینه مسیریابی و رایانش‌های توزیع‌شده یا خوشه‌ای دارد. برای نمونه در یک شبکه، فرستادنِ پلکانی پیام از خوشه‌ای به خوشه دیگری می‌تواند از راه سرخوشه‌ها انجام شود. سرخوشه گره‌ای است که مسیریابی درون و میان خوشه‌ای را انجام می‌دهد. پرسش اینجاست که برای چنین شبکه‌ای کدام گره‌ها باید برای سرخوشه شدن برگزیده شوند تا بتوان از هر خوشه‌ای به خوشه‌ای دیگر پیام فرستاد و هم‌چنین، کم‌ترین سرخوشه‌ها را داشته باشیم. یافتن مجموعه غالب کمینه برای چنین شبکه‌ای هم‌ارز با یافتن کم‌ترین تعداد سرخوشه‌ها است.

فرضیات تحقیق

فرضیات این تحقیق به شرح زیر می‌باشد:

- اجرای موازی الگوریتم یافتن مجموعه غالب متصل در گراف می‌تواند کارایی این الگوریتم را افزایش دهد و به خوبی بتواند در مسائل بهینه‌سازی بکار رود.
- اجرای موازی الگوریتم یافتن مجموعه غالب متصل در گراف مبتنی بر کتابخانه OpenMp به خوبی از قدرت محاسباتی موازی در سیستم چند پردازنده‌ای بهره برده و با کارایی مناسب و پیچیدگی زمانی مناسب و با توجه به‌سرعت بالا در همگرایی به جواب بهینه، سربار اضافی را به عملیات بهینه‌سازی تحمیل نمی‌کند.

UDG و DGB

UDG: فرض کنید $G = (V, E)$ یک گراف است که در آن V مجموعه ای از راس ها در شبکه و E مجموعه ای از یال هاست که تمام لینک ها در شبکه نشان می دهد. اگر تمام گره های شبکه دارای محدوده انتقال مشابه باشند، گراف G به عنوان یک UDG شناخته می شود[۹].

DGB: در عمل، محدوده انتقال تمام گره ها الزاما برابر نیست. در این مورد، یک شبکه می تواند با استفاده از یک گراف جهت دار $G = (V, E)$ مدل سازی شود. گره های V در یک صفحه اقلیدسی قرار گرفته و هر راس $v_i \in V$ دارای یک محدوده انتقال $r_i \in [r_{min}, r_{max}]$ است. یک یال جهت دار $(v_i, v_j) \in E$، اگر و تنها اگر $d(v_i, v_j) \le r_i$ و $d(v_i, v_j)$ نشان دهنده فاصله اقلیدسی بین v_i و v_j است. چنین نمودارهایی گراف‌های دیسک نامیده می شوند. یک یال (v_i, v_j) دو طرفه است اگر هر دو

[1] Dominating Set (DS)

(v_i , v_j) و (v_j , v_i) در E باشند، یعنی $d(vi, vj) \leq \min\{ri, rj\}$. نمودار دیسک که در آن تمام یال ها در شبکه دو طرفه هستند، DGB نامیده می شود. در این مورد، G جهت دار نیست[۹].

مجموعه غالب

مجموعه غالب: گراف بدون جهت $G = (V, E)$ داده شده است. V مجموعه ای از راس ها را نشان می‌دهد و E مجموعه ای از یال ها را مشخص می کند. مجموعه غالب[۲] (DS) از گراف G یک زیر مجموعه از گره هاست به طوری که هر گره در گراف یا در زیر مجموعه یا مجاور به حداقل یک گره در زیر مجموعه قرار دارد[۹].

اتصال DS: اگر زیرگروه القا شده توسط گره ها در یک DS متصل شود، مجموعه غالب را یک مجموعه غالب متصل (CDS) گویند. مجموعه غالب متصل نمودار توپولوژی شبکه می تواند به عنوان یک ستون مجازی استفاده شود تا به هر گره کمک کند تا پیام خود به سینک را ارسال کند. با کمک مجموعه غالب متصل، بار پیام های شبکه گرافی می تواند کاهش یابد، بنابراین مسیریابی بسیار ساده تر می شود و می تواند به سرعت به تغییرات توپولوژی شبکه منجر شود. از آنجا که تنها گره های مجموعه غالب متصل مسئول ارسال پیام های شبکه هستند، بنابراین گره های غیر از مجموعه غالب متصل می توانند ماژول ارتباطی خود را برای صرفه جویی در انرژی صرفه جویی کنند، در حالی که هیچ داده ای برای انتقال ندارند[۹].

مجموعه غالب متصل حداقلی: مجموعه غالب متصل حداقلی (MCDS) یک مجموعه غالب متصل با حداقل گره های ممکن در شبکه است. نشان داده شده است که مجموعه غالب متصل یک مسئله NP-Hard است. فرض کنید $G = (V, E)$ یک گراف متصل و بدون جهت باشد. تعداد رأس های مجاور به رأس vi (یا تعداد یال هایی که در رأس vi اتفاق می افتد) به عنوان درجه این رأس تعریف می شود که آن را با Di نشان می دهیم. یک مجموعه غالب متصل با درجه محدود (DC-CDS) گراف G یک مجموعه غالب متصل از G با توجه به $Di \leq D_{cons}$ است، برای همه $vi \in V$، که D_{cons} یک عدد صحیح مثبت است که نشان دهنده محدودیت درجه است[۹].

حداقل سلسله مراتب متصل با محدودیت درجه: DC-CDS با حداقل گره های ممکن در شبکه MDCS (DC-) با محدودیت درجه نامیده می شود. هر گره ستون فقرات در شبکه دارای وزن است. وزن را می توان از نظر انرژی، زمان، عرض باند و غیره تعریف کرد. بدیهی است، با کاهش تعداد گره های ستون فقرات، وزن ستون فقرات نیز کاهش می یابد. بنابراین، در یک سناریو واقع بینانه که هر گره وزن متفاوت دارد، اگر مجموعه غالب متصل دارای حداقل وزن بجای حداقل کاردینالیتی، مقیاس MCDS (یا DC-MCDS) مقرون به صرفه است[۹].

مجموعه غالب متصل با حداقل وزن: با توجه به گراف بدون جهت و گره وزن دار $G =(V, E, W)$، Vنشان دهنده مجموعه ای از گره ها، E نشان دهنده مجموعه ای از یال ها و W نشان دهنده مجموعه ای از وزن های مرتبط با گره گراف است. مجموعه غالب متصل با حداقل وزن (MWCDS) گراف G یک مجموعه غالب متصل از G دارای حداقل وزن است و نشان داده شده است که MWCDS مساله NP-hard است[۱۰].

MWCDS با محدودیت درجه: فرض کنید $G =(V, E, W)$، یک گراف متصل، بدون جهت و گره وزن دار است. مجموعه غالب متصل با حداقل وزن بهینه (DC-MWCDS) گراف G یک مجموعه غالب متصل با حداقل وزن تحت محدودیت درجه Dcons است. از طرف دیگر ODC- MWCDS یک مساله بهینه سازی است که به دنبال مجموعه غالب متصل با کمترین وزن با درجه بهینه می گردد[۱۰].

MWCDS محدود به درجه و انرژی: فرض کنید $G =(V, E, W)$، یک گراف متصل، بدون جهت و گره وزن دار است. مجموعه غالب متصل با حداقل وزن محدود به درجه (OEDC-MWCDS) از گراف G یک مجموعه غالب متصل با حداقل وزن و درجه محدود و انرژی بهینه است. از سوی دیگر، OEDC-MWCDS یک مشکل بهینه سازی است که سعی دارد مجموعه غالب متصل با کمترین وزن و درجه محدود و انرژی بهینه داشته باشد. در بخش بعد، ما برای ساختن ستون فقرات انرژی در گراف دیسک با لینک های دو طرفه از مساله OEDC-MWCDS استفاده می کنیم[۱۱].

[2] Dominating Set (DS)

مجموعه مستقل (IS): با توجه به گراف بدون جهت G = (V, E) یک مجموعه مستقل از گراف G یک زیر مجموعه از رأس هایی است که بین هیچ دو گره در زیر مجموعه یال وجود ندارد[۱۱].

مجموعه مستقل حداکثر: مجموعه مستقل حداکثر (MIS) مجموعه ای مستقل است که نمی تواند هیچ گره بیشتری را در V پذیرفته باشد. بنابراین MIS یک DS از یک گراف است. توجه داشته باشید که این DS (به عنوان MIS به دست آمده) ممکن است همبند نباشد[۱۱].

پردازش گراف‌های بزرگ

پردازش مبتنی بر گراف، شاخه‌ی دیگری از پردازش بر روی داده‌های حجیم هست که بر روی داده‌های حجیم به‌صورت گراف کار می‌کند. یکی از مثال‌های معروف در این زمینه شبکه‌های اجتماعی است. برای مثال برای بررسی ارتباطات بین افراد و یا کشورهای مختلف باید گراف دوستی بین این‌ها مورد پردازش قرار بگیرد[۱۲]. در گذشته برای پردازش گراف‌ها نیز از الگوریتم‌هایی نظیر نگاشت کاهش استفاده می‌شد اما گوگل با سیستم جدیدی به نام Pregel به خوبی گراف‌ها را مورد پردازش قرار می‌داد. پردازش گراف‌های بزرگ مشکلات زیادی دارد. برای مثال معمولاً محلی بودن دسترسی به حافظه در آن‌ها بسیار پایین است[۱۳]. علاوه بر این پردازش‌هایی که در رئوس هر گراف انجام می‌شود خیلی کم است و از طرفی گراف بزرگ است و دیگر مشکلی که وجود دارد این است که درجه موازی‌سازی رئوس در طول اجرا متغیر است. و درنتیجه برنامه‌هایی مانند نگاشت کاهش به خوبی نمی‌تواند گراف‌ها را پردازش کند[۱۳].

موازی‌سازی الگوریتم یافتن مجموعه غالب متصل در گراف

قدمت الگوریتم های موازی یافتن مجموعه غالب متصل به حدود ۳ دهه پیش بر می گردد. بیشتر اجراهای موازی شکل گرفته برای یافتن مجموعه غالب متصل از ساختار کلی این الگوریتم همگام سطح پیروی میکنند. علاوه بر حفظ پیچیدگی موازی، سه روند بهینه سازی کلیدی عبارت اند از [۳۵]:

- اطمینان از موازی سازی لبه
- متعادل‌سازی بار
- کاهش هزینه های همگام سازی به دلیل بروز رسانی های اتمی و همگام سازی مانع در پایان هر سطح

بهبود موقعیت مرجع حافظه با تغییر طرح گراف و یا ساختارهای داده یافتن مجموعه غالب متصل.

الگوریتم موازی برای یافتن مجموعه غالب متصل گراف

در این بخش فرض بر این است که G = (V,E) و نمودار اختیاری است که در $p \geq 2$ پردازشگرهای ارسال پیام چند پردازنده ای در یک حالت تقریبی، تقسیم شده است. ما یک نسخه موازی ساده را برای تقسیم بندی رئوس G در مجموعه‌های تقریبا برابر ارائه داده ایم. هر مجموعه بر روی پردازشگر خودش قرار گرفته است. هدف ایجاد بخشی از گراف با چندین لبه متصل به رئوس در مجموعه‌های مختلف است. از آن جایی که ما فقط به گراف‌های بسیار پراکنده توجه داریم، فرض می‌کنیم که G به عنوان مجموعه ای از لیست‌های مجاور ذخیره شده است. پردازشگر زمانی متغییر $\upsilon \in V$ را تعیین می‌کند که دارای لیستی از رئوس مجاور با V ذخیره شده در حافظه جانبی می‌باشد. درنهایت، از آنجا که علاقه مند به پیدا کردن جداکننده لبه با حداقل تعداد لبه‌ها هستیم، فرض می‌کنیم که تمام لبه‌ها دارای یک هزینه هستند.

الگوریتم ما با تقسیم هسته‌های پردازشگر کتابخانه OpenMP که با مجموعه P نشان داده می‌شود، به دو زیر مجموعه P_1 , P_2 با اندازه‌های مختلف متفاوت بیش تر از ۱ آغاز می‌شود. مجموعه‌های P_1 , P_2 تقسیم رأس‌ها را به وجود می‌آورند. هدف اولیه ما کاهش تعداد لبه‌های متصل به رئوس در P_1 , P_2 است. اگر $P_1 = \phi, P_2 = \phi$ باشد، ما کاری برای انجام دادن نداریم، پس پروسه را متوقف می‌سازیم. درغیر این صورت روش زیر را ادامه می‌دهیم. ابتدا برای هر بخش یک پردازشگر انتخاب

می‌کنیم. می‌گوییم $L_1 \in P_1$ و $L_2 \in P_2$ می‌باشد و نخ اجرایی بخش محسوب می‌شود. اگر $S \in P_i$ باشد، می‌توانیم بگوییم که نخ اجرایی S، L_i است.

هر پردازشگر در $P_1 \cup P_2$، مقادیر D تمام رئوس را محاسبه می‌کند. مقدار D بیانگر تفاضل جمع هزینه‌یال های خروجی یک گره و جمع هزینه‌یال های ورودی یک گره است. پردازشگر سپس آن‌ها را برای نخ اجرایی گزارش می‌دهد. بعد از آن هر نخ اجرایی یک حلقه نا مشخص با بیش ترین مقدار D، انتخاب می‌کند و یکدیگر را از انتخاب خود مطلع می‌سازند، سپس دو راس را علامتگذاری می‌کنند و آن‌ها را در لیستی به همراه نتایج خود ذخیره می‌کنند. سپس مقادیر D، رئوس نامشخص را به روز رسانی می‌کنند.

از این معادلات دریافتیم که آن‌ها به لیست‌های مجاور هر دو راس نیاز دارند. نخ‌های اجرایی، این اطلاعات را از پردازشگری تقاضا می‌کنند که برای رئوس انتخاب شده تعیین شده اند. پس از دریافت اطلاعات، مقادیر نسبی D را به روز رسانی می‌کنند. آن‌ها این روند علامت گذاری رئوس و به روز رسانی مقادیر D تا زمانی که تمام رأس‌ها برای پردازشگرهای مشخص شده در P_1 , P_2 تعیین شوند، تکرار می‌کنند. سپس نخ‌های اجرایی با استفاده از روند مشابه مثل الگوریتم سریال تصمیم می‌گیرند که چه راس هایی را جایگزین کنند. آن‌ها پردازشگرهای تصمیماتشان را آگاه می‌سازند و پردازشگرها، لیست‌های مجاور انتخاب شده را جایگزین می‌کنند.

بعد از جا به جایی رئوس، هر پردازشگر،دارای تعداد یکسانی رئوس می‌باشند. پردازشگر این الگوریتم کلی را تکرار می‌کند تا تعداد لبه‌های خارجی بین P_1 , P_2 نتوانند کاهش یابد. سپس مجموعه‌های P_1 , P_2 را به‌صورت موازی برای این الگوریتم به کار می‌گیرند. پروسه کلی در الگوریتم ۱ مطرح شده است.

الگوریتم ۱: الگوریتم موازی برای یافتن مجموعه غالب متصل در گراف

۱. پردازشگرها خودشان را به ۲ گروه P_1 , P_2 با اندازه‌های مختلف تقسیم می‌کنند. اگر هر گروه خالی باشد، آن‌ها کار خودشان را متوقف می‌سازند. درغیر این صورت، آن‌ها برای هر گروه یک پردازشگر انتخاب می‌کنند و می‌گویند : $L_1 \in P_1, L_2 \in P_2$

۲. هر پردازشگر در $P_1 \cup P_2$ ،مقادیر D رئوس را محاسبه می‌کنند.

۳. هر پردازشگر، مقادیر D را به نخ اجرایی گزارش می‌دهد. نخ اجرایی، تمام رئوس که در نیمی از بخش قرار دارد را بی نشان می‌کند.

۴. هر نخ اجرایی L_i، یک راس بدون علامت V_i با بیش ترین مقدار D را انتخاب می‌کند و سپس آن را علامت گذاری می‌کند و نخ اجرایی دیگر را از انتخاب خود آگاه می‌سازد.

۵. نخ‌های اجرایی، لیست‌های مجاور V_1 و V_2 را از پردازشگرهای تعیین شده تقاضا می‌دهند و مقادیر D رئوس مشخص نشده را به روز رسانی می‌کنند.

۶. اگر حداقل یک راس، در نیمی از بخش بدون علامت گذاری باقی بماند، پردازشگرها باید پروسه را از مرحله ۴ تکرار کنند.

۷. رهبران تصمیم می‌گیرند که رئوس را با استفاده از لیست جفت رئوس جابه‌جا کنند و سپس آن را به پردازشگرهای گروهشان بگویند.

۸. پردازشگرها، جا به جایی رئوس را انجام می‌دهند.

۹. پردازشگرها پروسه را از مرحله ۲آغاز می‌کنند و آن را تکرار می‌کنند تا پیشرفت بیشتری حاصل نشود.

۱۰. هرکدام از مجموعه‌های P_1 , P_2 الگوریتم را به‌صورت موازی از مرحله ۱ به کار می‌گیرند.

برای کاهش تعداد پیام‌های رد و بدل شده بین P_1 , P_2 ، رئوس a و b را برای جا به جایی انتخاب می‌کنیم تا $D_a + D_b$ را به حداکثر برساند. یعنی لبه احتمالی بین a و b را نادیده می‌گیریم. بنابراین می‌توانیم رئوسی را انتخاب کنیم که دستاورد واقعی آن کمتر از حداکثر باشد. همانطور که گفته شد، الگوریتم به تعداد زیادی پیام در حال عبور نیاز دارد. هرپردازشگر، به‌صورت مداوم، تمام لیست‌های مجاور را به نخ اجرایی ارسال می‌کند. از آنجایی که می‌خواهیم مسائل بسیار بزرگ را برای پردازشگر حل کنیم، برخی از این پیام‌های انتقالی به‌صورت اجتناب ناپذیر می‌باشند.

به هرحال می‌توانیم ارسال پیام را از طریق ۲ نخ اجرایی کاهش دهیم و هنگامی که به نظر می‌رسد اصلاحات بیشتر در حال انجام شدن می‌باشند، علامت گذاری رئوس متوقف می‌شود. در این پیاده سازی، نخ اجرایی علامتگذاری رئوس را از زمانی متوقف می‌سازند که مجموع تمام نتایج محاسبه شده کمتر از $-d$ باشد و یا آن‌ها با نتایج منفی متوالی بیش تر از d رو به رو شده باشند. درحالی که d، با حداکثر اندازه هر راس در نمودار برابر می‌باشد، از آنجایی که ما در ابتدا به نمودارهای پراکنده توجه داشته ایم، به یکباره مجموع تمام نتایج‌های محاسبه شده منفی شدند و به‌صورت منفی باقی ماندند.

علاوه براین، با توجه به انتقال اولیه رئوس برای پردازشگرها، نخ‌های اجرایی تنها فقط یکبار چندین نتایج منفی متوالی را مشاهده می‌کنند و استفاده از این روش، امکان بهبود بیشتر وجود ندارد. این اصلاحات زمان اجرای الگوریتم را بدون تاثیر چشمگیر بر اندازه جدا کننده‌های لبه، بهبود می‌بخشد. در بخش‌های ۷ و ۸، بیشتر در مورد انتقال اولیه رئوس به پردازشگرها بحث شده است. اغلب پردازشگرها با اجرای الگوریتم توسط نخ‌های اجرایی، بیکار مانده اند. اگرچه در ابتدای این الگوریتم، یک الگوریتم موازی وجود دارد. بیشتر پردازشگرها در کارهایی مثل اقدامات الگوریتم همکاری دارند. به عنوان مثال، بیش تر پردازشگرها به نخ اجرایی تبدیل شده اند.

با توزیع قسمت‌های مستقل الگوریتم یافتن مجموعه غالب متصل در گراف مبتنی بر موازی سازی از طریق رویکرد تقسیم و غلبه بر روی چندین المان پردازشی که به طور موازی کار می‌کنند، قادر خواهیم بود تا تسریع محاسباتی را به دست آوریم. اولین انتخاب در موازی‌سازی می‌تواند از استقلال بین افراد حداقل در گام ارزیابی استفاده کند. به طور سنتی مدل‌های موازی براساس اداره کردن جمعیت‌ها طبقه‌بندی می‌شوند. انتخاب بین جمعیت مجتمع یا توزیع شده معمولاً به عنوان گزینه اصلی در پیدایش زیر جمعیت‌ها و میزان ارتباط بین آن‌ها است.

مدل Master-Slave از مهم ترین مدل های پردازش موازی است که نخ اصلی Master بوده و از طریق ارسال پیام نخ های Slave یا اجرایی را مدیریت می کند. از جمعیت مجتمع استفاده می‌کند و ارزیابی افراد را به شکل موازی انجام می‌دهد. مدل‌های موازی مجتمع هیچ فرضی را از معماری کامپیوتر زیرین در نظر نمی‌گیرد و می‌تواند به‌صورت مؤثر روی حافظه اشتراکی پیاده‌سازی شود. روی یک حافظه اشتراکی با چندین پردازنده، جمعیت می‌تواند در حافظه اشتراکی نگهداری شود و هر پردازنده می‌تواند جمعیت تخصصی داده شده را خوانده و نتایج ارزیابی را بدون هیچ تداخلی بازنویسی کند.

برای اکثر سیستم‌ها، توانایی برای استفاده مجدد از یک معماری سخت‌افزاری سیستم برای عملیات‌های زیاد یا اصلاحات طراحی، یک ویژگی جذاب خواهد بود. این انعطاف‌پذیری می‌تواند از طریق استفاده از سخت‌افزار و هسته‌های پردازشی با قابلیت اجرای موازی با کتابخانه موازی‌سازی OpenMp به دست آید. استفاده از چنین رویکردهای مربوط به اجرای موازی محاسبات می‌تواند اندازه فیزیکی و هزینه سیستم جاگذاری شده را کاهش داده و انعطاف‌پذیری سیستم برای اصلاحات را افزایش دهد.

ازآنجایی‌که هر عامل خادم به‌عنوان یک تابع تعریف‌شده، پس هر یک از این عامل‌ها می‌توانند چندین بار در طراحی بکار گرفته شوند. در این طراحی هر تابع ادغام یک‌بار در کتابخانه موازی‌سازی OpenMp فراخوانی شده تا فرآیند مخصوص به خود را اجرا کرده و همچنین در فرآیند اجرای موازی نیز فراخوانی می‌گردد. بر این اساس اجرای موازی می‌تواند با منابع موجود اجرا گردیده که این باعث استفاده کارآمد از منابع موجود گردیده که به طراحی کمک می‌کند تا عاملیت بهتری را از خود نشان دهد.

پس از مطالعه الگوریتم یافتن مجموعه غالب متصل در گراف، به منظور ارائه پیاده‌سازی موازی این الگوریتم، مطلوب است که بین داده‌ها وابستگی وجود نداشته باشد. به همین ترتیب، داده‌ها را می‌توان به بخش‌های کوچک تقسیم کرد، هر نخ می‌تواند یک بخش را محاسبه کند. در نتیجه این روش همبستگی داده، سرعت پردازش الگوریتم بخش‌بندی گراف را افزایش می‌دهد. در سطح نخ، گراف اصلی به چند زیرگراف تقسیم می‌شود، عملیات یافتن مجموعه غالب متصل بصورت بازگشتی برای هر بخش انجام می‌شود، سپس الگوریتم یافتن مجموعه غالب متصل کلی گراف را می‌توان با چندین نخ در کتابخانه موازی‌سازی openMp انجام داد و هر نخ فقط باید عناصری را که به آن اختصاص داده شده است به دست آورد و همان تابع الگوریتم

یافتن مجموعه غالب متصل در گراف را برای این عناصر اجرا کند. به عبارت دیگر، هر موضوع به طور مستقل می‌تواند پیمانه‌ای نمایی را انجام دهد.

در ساختار مدیریت هم‌زمانی نخ با استفاده از کتابخانه OpenMp، برای پردازنده چندهسته‌ای، زمانی که یک هسته فراخوانده می‌شود، روی یک شبکه اجرا می‌شود. تعداد بلوک و نخ در یک شبکه می‌تواند ساخته شود. نخ‌ها ممکن است مکان‌های مختلف حافظه را وارد کند. هر نخ یک حافظه خصوصی دارد. هر بلوک دارای یک حافظه مشترک است که قابل دسترسی برای هر نخ در همان بلوک است. تمام نخ‌های بلوک‌های مختلف به حافظه سراسری دسترسی دارند. با این حال، یک هسته را می‌توان با چندین بلوک از نخ‌ها با استفاده از کتابخانه OpenMp اجرا کرد، بنابراین تعداد کل نخ‌ها به معنی تعداد نخ‌ها در هر بلوک ضرب در تعداد بلوک است. شکل (۱)، شبه کد الگوریتم موازی یافتن مجموعه غالب متصل در گراف بر اساس کتابخانه OpenMp را نشان می‌دهد.

```
1 function MCDS(G(V,E)):
 2      determine a balanced initial partition of the nodes
into sets A and B for finding Maximal Connected Dominant Set
(MCDS) in Graph
 3
 4   #pragma omp parallel section
 5   {
 6      do
 7         compute D values for all a in A and b in B
 8         let gv, av, and bv be empty lists
 9         #pragma omp parallel for shared(D, A, B)
 10        for (n= 1 to |V|/2)
 11        find a, b from A, B, where g = D[a]+ D[b]-2*c(a, b)
is maximal
 12            remove a and b from further consideration in
this pass
 13           add g to gv, a to av, and b to bv
 14           update D values for the elements of A = A \ a
and B = B
 15        end for
 16        find k which maximizes g_max, the sum of gv[1],...,
gv[k]
 17        if (g_max > 0) then
 18          Exchange av[1],av[2],...,av[k] with
bv[1],bv[2],..,bv[k]
 19     until (g_max <= 0)
 20   }
 21  return G(V, E)
```

شکل ۱: شبه کد الگوریتم موازی یافتن مجموعه غالب متصل در گراف بر اساس کتابخانه OpenMp

در برنامه‌نویسی موازی به جای استفاده از حلقه‌های معمولی، بخش‌های مختلف گراف را بین همه پردازنده‌ها تقسیم نموده و کار به طور موازی انجام می‌گردد تا از همه ظرفیت سیستم استفاده شود و زود تر به زمان اجرای نهایی برسیم. با استفاده از رهنمون Parallel Section که برای اجرای موازی یک بخش است و همچنین رهنمون Parallel For که برای اجرای موازی یک حلقه است، این کار به راحتی هر چه تمام تر انجام می‌گیرد. این دستورات به طور خودکار با توجه به تعداد

پردازنده‌ها و هسته‌های موجود، تعداد Threadهای مناسبی را به کار خواهند گرفت و کار را ما بین Threadها تقسیم کرده و موازی‌سازی را انجام می‌دهند که در این عملیات هر پردازنده عملیات را بر روی یک زیرگراف انجام می‌دهد و عملیات یافتن مجموعه غالب متصل را در زیر گراف مربوط به خود انجام می‌دهد. کار به این ترتیب است که متد For ابتدا تعداد Threadهای مناسبی را (با توجه به سخت‌افزار موجود) در اختیار می‌گیرد. سپس هر Thread مسئولیت اجرای یک بار چرخش حلقه را به عهده می‌گیرد. در هر بار چرخش هر Thread متدی که مشخص کرده ایم را اجرا می‌کند. پس از اتمام اجرای متد، چرخش دیگری به Thread محول می‌شود و الی آخر.... در این روش، برخلاف حلقه‌های معمولی، طبیعتا نباید انتظار داشته باشیم که حلقه به‌صورت سریالی اجرا شود و هیچ ترتیب مشخصی در چرخش‌ها وجود نخواهد داشت.

پیاده‌سازی ساختار موازی بر اساس کتابخانه OpenMp

برای پیاده‌سازی الگوریتم‌ها به‌صورت موازی کتابخانه‌های متعددی همراه با رابط‌های مناسب ارتباطی در زبان‌های برنامه‌نویسی مختلف توسعه‌یافته‌اند. کتابخانه‌یOpenMPیک کتابخانه‌ی C/C++ است که توسط میکروسافت و به منظور تسهیل برنامه‌نویسی برای پردازنده‌های چندهسته‌ای تهیه شده است. ازآن جمله می‌توان به کتابخانه‌هایی مبتنی بر رابط ارسال پیام زبان‌هایی چونC++، Cو Fortran نظیرMPI، MPICH2،PICH، SCor و...اشاره نمود. برخی از این کتابخانه‌ها تحت لیسانسGNU توسعه یافته و به سادگی تحت سیستم عامل‌هایی چون لینوکس و...به‌صورت رایگان قابل استفاده می‌باشند. کتابخانهMPI به دلیل کارایی بال در صنعت و همچنین درکارهای تحقیقاتی به شدت مورد استفاده می‌باشد. البته باید متذکر شد که همه نسخه‌های موجودMPI نمی‌توانند از حافظه مشترک بین پردازه‌ها نهایت استفاده را برند، چرا کهMPI تنها امکان ارتباط بین پردازنده‌ها با حافظه توزیع شده را فراهم می‌آورد. بدیهی است که استفاده از حافظه مشترک سریع‌ترین راه را نسبت به هر وسیله ارتباطی دیگر جهت انتقال داده بین پردازنده‌ها فراهم خواهد آورد .جهت استفاده از حافظه مشترک می‌توان از کتابخانهOpenMP استفاده نمود. مشخص است که استفاده هم‌زمان از این کتابخانه‌ها می‌تواند به لحاظ سرعت اجرای برنامه منجر به بهترین نتیجه گردد.

باید توجه داشت، OpenMP تضمین نمی‌کند که از حافظه اشتراکی استفاده بهینه خواهد کرد. همچنین مواردی مانند وابستگی داده‌ها، شرایط مسابقه یا بن‌بست‌ها[3] باید توسط خود برنامه‌نویس در کد برنامه کنترل شود وOpenMP عموماً نمی‌تواند کاری درباره آن‌ها انجام دهد. هم‌زمان‌سازی ورودی و خروجی هنگام دسترسی موازی و چک کردن ترتیب اجرای کد برنامه نیز ازجمله وظایف برنامه‌نویس است و از عهده OpenMP خارج است. به‌این ترتیب، برنامه‌نویس باید ساختار کد و الگوریتم خود را کاملاً کنترل کرده و اطمینان حاصل کند که موارد ذکر شده در اجرای برنامه رخ نخواهد داد.

روش پیشنهادی از مدل Join-Fork برای اجرای موازی استفاده می‌نماید. همه ی برنامه‌های OPenMP به عنوان یک پروسس تنها شروع می‌شود یعنی نخ اصلی اجرا می‌شود تا زمانی که با اولین مولفه ناحیه موازی مواجه می‌شوند. Fork بیانگر زمانی است که نخ اصلی سپس تیمی از نخ های موازی ایجاد می‌کند. قسمتی از برنامه که در ناحیه موازی قرار داده شده است به‌صورت موازی بین تیم نخ های مختلف قرار داده می‌شود. Join بیانگر زمانی است که قسمت موازی برنامه کامل گردید، نخ‌ها به طور هم‌زمان خاتمه می‌یابند و فقط نخ اصلی باقی می‌ماند.

یک برنامه ساخته شده با این مدل برنامه نویسی موازی می‌تواند بر روی یک سیستم کامپیوتری اجرا شود، به طوری که OpenMP برای موازی بودن در گره (چند هسته ای) استفاده می‌شود. OpenMP یک روش موازی سازی با پیاده سازی چندنخی است، به این ترتیب که یک نخ به عنوان نخ اصلی در نظر گرفته شده که در آن یک سری دستورالعملها به‌صورت پیوسته اجرا می‌شود و یک تعداد مشخصی از نخ‌های فرعی را ایجاد می‌کند و سیستم یک کار را در میان آن‌ها تقسیم می‌کند. پس از آن موضوعات از طریق اختصاص دادن نخ‌ها به پردازنده‌های مختلف اجرا می‌شوند. قسمتی از کد که به‌صورت موازی اجرا می‌شود، براساس دستوراتی و بر اساس دستورالعمل رهنمون کامپایلر مشخص می‌شود که باعث می‌شود تا با رسیدن به

[3] Deadlock

این بخش، عملیات تولید نخ‌های جدید و موازی سازی عملیات انجام شود. هر نخ دارای شناسه متصل به آن است که می‌تواند با استفاده از یک تابع به نام omp_get_thread_num بدست آید. شناسه رشته یک عدد صحیح است و نخ اصلی دارای شماره صفر است. پس از اجرای کدهای موازی و اتمام کار تمامی نخ‌ها، رشته‌ها به نخ master متصل می‌شوند که به سمت انتهای برنامه ادامه می‌یابد. به طور پیش فرض، هر نخ به طور مستقل یک بخش موازی از کد را اجرا می‌کند. ساختارهای به اشتراک گذاشتن کار می‌تواند مورد استفاده قرار گیرد تا تقسیم یک وظیفه در میان موضوعات به طوری که هر موضوع بخش اختصاصی آن را اجرا می‌کند. در این روش همپوشانی و همپوشانی داده‌ها با استفاده از OpenMP به دست می‌آید.

نحوه ایجاد نخ

عناصر اصلی OpenMP ساختارهایی برای ایجاد نخ‌ها، توزیع بار کاری (به اشتراک گذاری کار)، مدیریت داده‌های محیطی، هماهنگ سازی نخ‌ها، رویه‌های زمان اجرا در سطح کاربر و متغیرهای محیطی هستند. در OpenMP از #pragmas استفاده می‌کند. در محاسبات موازی با OpenMp، مدل fork-join راهی برای راه اندازی و اجرای برنامه‌های موازی است، به طوری که اجرایی کردن واحدهای موازی در نقاط مشخص شده در برنامه، برای پیوستن در یک نقطه ی بعدی و اجرای ترتیبی انجام می‌شود. بخش‌های موازی ممکن است به‌صورت بازگشتی تا زمانی که یک جزئیات دقیق کار به دست می‌آید، ادامه پیدا کند. تعیین تعداد نخ‌ها و واحدهای موازی برای اجرای هر بخش از طریق دستور Omp_set_num_threads(x) می‌باشد که در اینجا x تعداد واحدهای موازی یا همان تعداد نخ‌های مجازی می‌باشد.

موازی سازی حلقه در OpenMP

تقریبا تمام برنامه‌های مفید دارای نوعی حلقه در کد هستند که شامل حلقه for یا While است. این امر مخصوصا برای همه برنامه هایی است که زمان زیادی را برای اجرای آن می‌گذرانند، بیشتر وجود دارد اغلب زمان ها، تکرارهای مختلف این حلقه‌ها هیچ ارتباطی با یکدیگر ندارند، بنابراین این حلقه‌ها یک هدف اولیه برای حل مساله هستند. OpenMP به طور مؤثر از این ویژگی‌های برنامه رایج استفاده می‌کند و بسیار آسان است که برنامه OpenMP را قادر به استفاده از چند پردازنده به سادگی با اضافه کردن چندین دستورالعمل کامپایلر به کد منبع خود کنید. همزمان سازی حلقه‌ها با OpenMP بسیار ساده است. موازی کردن یک حلقه، به سادگی و از طریق نوشتن چندین پارامتر مختلف مشخص می‌گردد و OpenMP از بقیه امور مراقبت می‌کند.

با استفاده از کلاس Parallel و متدهای For این کار به راحتی هر چه تمام تر انجام می‌گیرد. این دستورات به طور خودکار با توجه به تعداد پردازنده‌ها و هسته‌های موجود، تعداد Threadهای مناسبی را به کار خواهند گرفت و کار را ما بین Threadها تقسیم کرده و موازی‌سازی را انجام می‌دهند. در این تحقیق بر اساس اصل بازگشتی و تیوری مدل برنامه‌نویسی تقسیم وغلبه، بخش‌های متفاوت الگوریتم بخش‌بندی گراف از هم مستقل بوده و با توجه به این شرایط، می‌توان از فرآیند موازی سازی بهره برد.

چارچوب کلی ارزیابی

از آنجا که عملیات یافتن مجموعه غالب متصل مجزای گراف توسط هر پردازنده با دید سراسری پردازنده یک پردازنده به داده ها متفاوت بوده و نتایج با نتایج یافتن مجموعه غالب متصل گراف اصلی متفاوت خواهد بود. بنابراین نیاز خواهد بود تا روشی معرفی گردد که این روش بتواند به خوبی ارتباطات بین پردازنده های مختلف را برای آگاهی از اشتراکات موجود در زیر بخش های مختلف تقسیم شده برقرار سازد و بعد از یافتن مجموعه غالب متصل گراف های مختلف عملیات همپوشانی اصلی گراف را دنبال کنند. به دلیل نیاز به ارتباطات بسیار زیاد بین پردازنده های مختلف در عملیات یافتن مجموعه غالب متصل گراف استفاده از کتابخانه موازی سازی OpenMP بسیار به کارآمدی این روش کمک خواهد نمود. دلیل این امر این است که روش موازی سازی چند نخی بر اساس امکانات کتابخانه OpenMP بر روی سیستم چند هسته ای از مکانیزم ارتباط بر اساس

حافظه اشتراکی استفاده می‌کند و این استفاده از حافظه اشتراکی کلید موفقیت و کارآیی روش هایی است که نیاز به ارتباطات بسیار زیادی بین پردازنده های در حال اجرا دارد. سیستم حافظه اشتراکی به این معنی نیست که هر پردازنده از حافظه جداگانه برای خود اقتصاد استفاده نمی‌کند و هر تغییری که توسط یک پردازنده در بخشی از گراف انجام شود، به دلیل وجود حافظه اشتراکی و قابل استفاده توسط تمامی پردازنده های دیگر این در این تغییر در تمامی پردازنده های دیگر نیز قابل مشاهده بوده و عملیات همروندی بین پردازنده مستقل است. ارتباطات ما بین سیستم‌های اجرایی سریع خواهد بود که این امر کار را با این روش ها به شدت بهبود می‌بخشد و برای کاربردهایی چون کاربرد پیش روی این مسئله یعنی یافتن مجموعه غالب متصل موازی گراف بسیار کارا می باشد.

برای اجرای روش پیشنهادی باید روش بخش پیشنهادی به نحوی که در بخش قبل بیان گردید، بر روی داده‌های گرافی مختلف اعمال گردیده و در نهایت یافتن مجموعه غالب متصل بر اساس زمان ارای کل تحلیل گردد. در اینجا برای ارزیابی کارایی روش پیشنهادی از مقایسه زمان اجرای حالت سریال با حالت موازی شده با کتابخانه OpenMp استفاده می‌گردد.

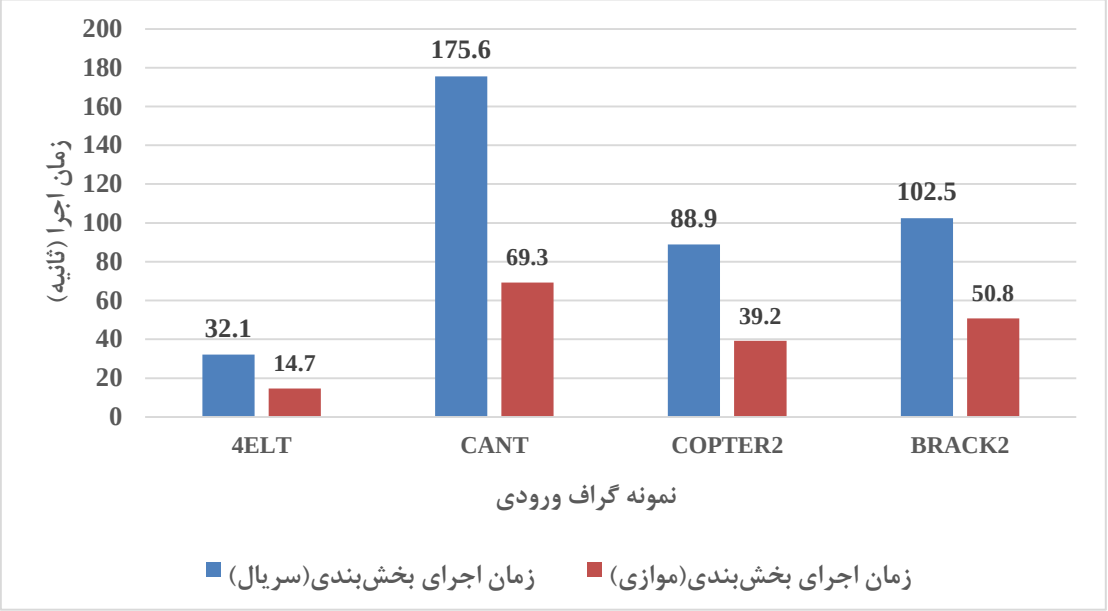
ارزیابی روش یافتن مجموعه غالب متصل موازی گراف با حالت سریال

با توجه به این‌که حل بهینه و دقیق مساله یافتن مجموعه غالب متصل گراف، فقط در مسائل کوچک امکان‌پذیر است و حل مساله در ابعاد بزرگ تقریباً غیرممکن است، در این پژوهش از الگوریتم‌های موازی سازی کمک گرفته می‌شود و در زمان معقول، جواب‌های بهینه و یا نزدیک به بهینه را به دست می‌آورد. در این مقایسه تعداد پردازنده ها برابر با ۴ در نظر گرفته شده است. برای این کار از مقدار به دست آمده از زمان اجراهای مختلف برای مقایسه استفاده می‌گردد. این مقایسه بر حسب گراف های ورودی متفاوت نشان داده شده است.

جدول ۱: مقایسه زمان اجرا بین یافتن مجموعه غالب متصل سریال و موازی برای داده‌های مختلف (P=4)

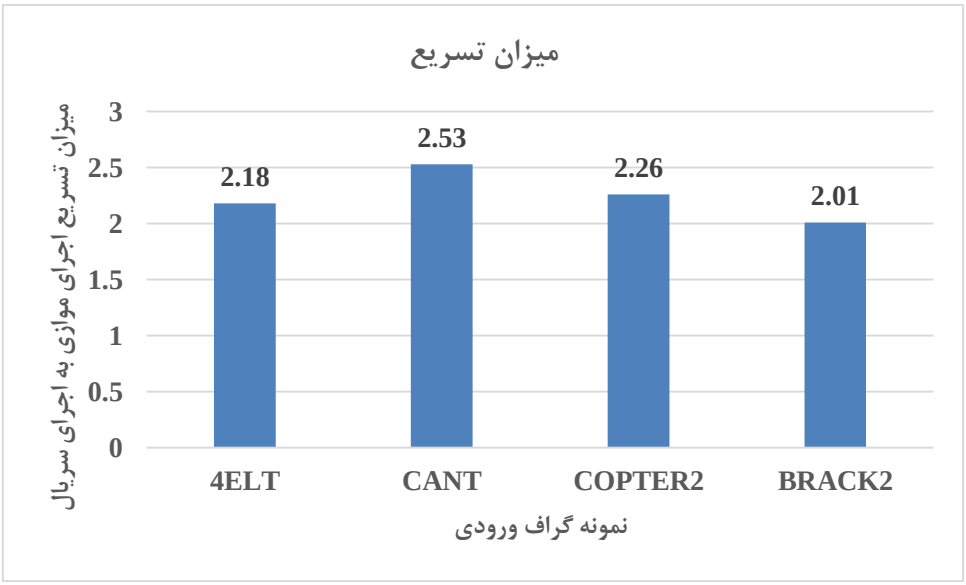
گراف ورودی	زمان اجرای یافتن مجموعه غالب متصل (سریال)	زمان اجرای یافتن مجموعه غالب متصل (موازی)	میزان تسریع
4ELT	۳۲,۱	۱۴,۷	۲,۱۸
CANT	۱۷۵,۶	۶۹,۳	۲,۵۳
COPTER2	۸۸,۹	۳۹,۲	۲,۲۶
BRACK2	۱۰۲,۵	۵۰,۸	۲,۰۱

ذکر این مطلب مناسب است که عملکرد به دست آمده با استفاده از یافتن مجموعه غالب متصل گراف حاصل در حالت استفاده از کتابخانه OpenMp و اجرای موازی عملیات بیان شده ، در مقایسه با روش یافتن مجموعه غالب متصل سریال دارای برتری می‌باشد. همان‌طور که نتایج نشان می‌دهد، روش پیشنهادی بر حسب تمامی مقادیر مختلف برای بخش های ایجاد شده توسط الگوریتم پیشنهادی برای تمامی نمونه ها دارای برتری قابل توجهی به حالت سریال دارد. همچنین باید به این نکته توجه نمود که با افزایش تعداد یال و بزرگ شدن گراف ها حجم عملیات های افزایش می‌یابد که اجرای موازی آن در حالت سریال با افزایش تعداد بخش، تسریع بیشتری را نشان می دهد. به عنوان نمونه گراف ورودی CANT که دارای تعداد یال بسیار بیشتری نسبت به نمونه های دیگر بوده عملیات پردازش را در زمان بسیار بیشتری انجام داده و در عین حال میزان تسریع نسبت به حالت سریال برای این نمونه ورودی بیشتر بوده است. نمودار شکل (۲) این مقایسه را به خوبی نشان می‌دهد.



شکل ۲: نمودار مقایسه زمان اجرا بین یافتن مجموعه غالب متصل سریال و موازی گراف

همان طور که نتایج نشان می‌دهد روش پیشنهادی با استفاده از ابزار موازی OpenMp به خوبی توانسته است زمان اجرای الگوریتم یافتن مجموعه غالب متصل را بهبود بخشد و در همه حالات تسریع بیشتر از ۲ را فراهم سازد. باید به این نکته توجه داشت که اگرچه روش پیشنهادی بر روی یک پردازنده با ۴ نخ مجزا بر روی ۴ هسته با ابزار OpenMp موازی شده است، اما در عمل هیچگاه نمی‌توان به تسریع ۵ برابر دست یافت. این به این دلیل است که بخشی از الگوریتم و روش پیشنهادی بصورت سریال اجرا می گردد و همچنین سربار ارتباطی بین پردازنده ها و سربار توزیع آن ها مانع از دست یابی به تسریع بیشتر می گردد. این نکته حائز اهمیت است که میزان تسریع روش پیشنهادی با افزایش حجم بخش و افزایش حجم محاسبات در بخش موازی، تسریع روش پیشنهادی افزایش یافته است. برای مشاهده این امر مقایساتی بین تسریع روش پیشنهادی نسبت به حالت سریال صورت گرفته که نمودار ارائه شده در شکل ۲ این مقایسه را نشان می‌دهد.



شکل ۳: نمودار مقایسه تسریع بر اساس نمونه‌های مختلف ورودی

ارزیابی روش یافتن مجموعه غالب متصل موازی با حالت سریال بر اساس تعداد واحد اجرایی موازی

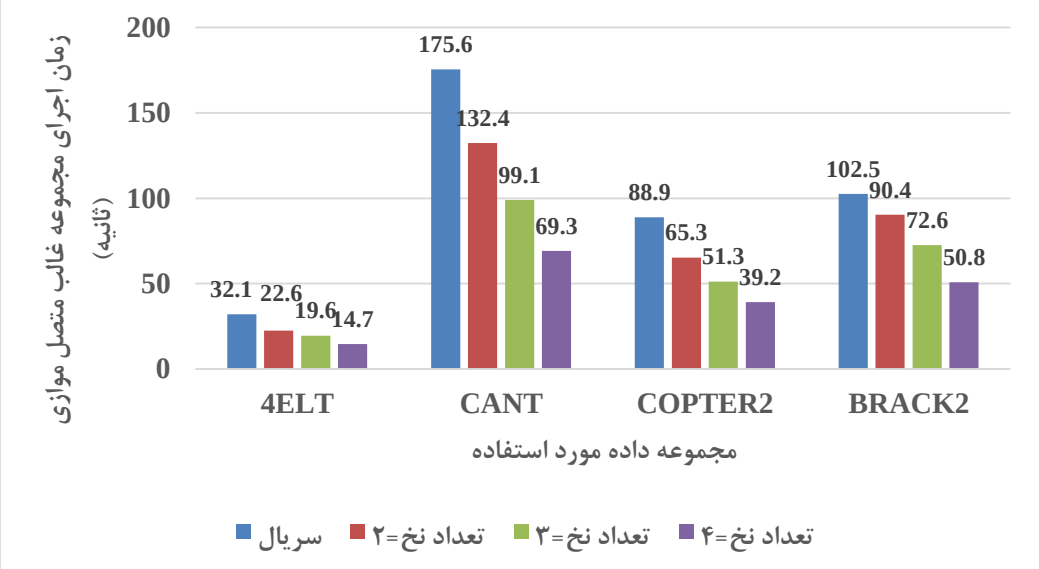
روش پیشنهادی از نسخه موازی شده الگوریتم یافتن مجموعه غالب متصل گراف بر روی پردازنده چندهسته‌ای بهره می برد و تعداد هسته های مورد استفاده تاثیری مستقیم بر سرعت اجرای برنامه خواهد داشت. همچنان‌که کامپیوترهای شخصی

متداول تر شده اندو بیشتر برنامه‌های کاربردی برای آنها طراحی شده‌است. کاربر نهایی نیاز به سریعتر شدن و نگه داری بهتر از سیستم را فهمیده‌است. تسریع به وسیله افزایش سرعت کلاک و اخیرا اضافه کردن هسته‌های پردازش چندگانه برای تراشه‌های مشابه به دست آمده‌است اگرچه سرعت تراشه در طی سالها به طور نمایی افزایش یافته‌است. این زمان در حال خاتمه یافتن است و تولیدکنندگان به سمت پردازش چندهسته‌ای تمایل دارند اگرچه به وسیله افزایش تعدادی هسته روی یک تراشه تنها چالش‌هایی در رابطه با حافظه و انسجام حافظه نهان و همچنین ارتباط بین هسته‌ها ناشی می‌شود. پروتکل‌های منسجم و شبکه‌های به هم متصل تعدادی از موضوعات را حل کرده‌اند اما تا زمانی که برنامه نویسان نوشتن برنامه‌های کاربردی موازی را یاد می‌گیرند مزیت کامل و کارایی پردازنده‌های چند هسته‌ای ها دست نخواهند یافت. برای ارزیابی روش پیشنهادی در این بخش مقایساتی بر حسب تغییر در زمان اجرای روش پیشنهادی بر حسب افزایش در تعداد هسته های اجرایی انجام شده است تا مقیاس پذیری روش پیشنهادی مورد ارزیابی قرار بگیرد. نتایج ارائه شده در جدول زیر این مقایسات را نشان می‌دهد.

جدول ۲: جدول مقایسه تسریع بین یافتن مجموعه غالب متصل موازی بر اساس تعداد هسته متفاوت

فایل گراف/ تعداد هسته	سریال	۲	۳	۴
4ELT	۳۲,۱	۲۲,۶	۱۹,۶	۱۴,۷
CANT	۱۷۵,۶	۱۳۲,۴	۹۹,۱	۶۹,۳
COPTER2	۸۸,۹	۶۵,۳	۵۱,۳	۳۹,۲
BRACK2	۱۰۲,۵	۹۰,۴	۷۲,۶	۵۰,۸

همان‌طور که از نتایج روش پیشنهادی به خوبی مشخص است با افزایش تعداد هسته‌های پردازنده مورد استفاده، کارایی افزایش می‌یابد و با افزایش تعداد واحدهای اجرایی موازی در کتابخانه OpenMP نیز شاهد افزایش در میزان تسریع به حالت یافتن مجموعه غالب متصل گراف در حالت سریال می‌باشیم. این امر کارایی روش پیشنهادی را نشان می‌دهد. نمودار زیر مقایسه این حالات را به خوبی نشان می‌دهد.



شکل ۴: نمودار مقایسه تسریع یافتن مجموعه غالب متصل موازی بر اساس تعداد هسته متفاوت

شایان ذکر است که در سیستم‌های موازی با رشد تعداد پردازنده ها انتظار می رود که در زمان اجرای عملیات نیز کاهش یابد. البته باید توجه داشت که این کاهش زمان الزاماً به صورت خطی نمی باشد و تسریع عملیات موازی سازی به روش

خطی به معنی این است که اگر شما ۴ واحد اجرایی موازی استفاده می کنید، در شما می بایست ۴ برابر تسریع داشته باشید و زمان اجرا می بایست بر چهار تقسیم گردد. کاربرد های واقعی امکان دستیابی به P برابر تسریع و کاهش زمان اجرا در حالت استفاده از P واحد اجرایی بر روی پردازنده های مختلف امکان پذیر نمی باشد. این امر به این دلیل است که تمامی برنامه های نوشته شده قابلیت اجرای موازی را ندارند و بخش عمده‌ای از زمان اجرای الگوریتم به صورت سریال صرف می‌گردد و تنها بخشی از برنامه به صورت موازی اجرا می شود. باید توجه داشت حتی در واقعیت، بخش موازی از اجرای برنامه نیز P برابر نخواهد شد، زیرا زمانی نیز برای ارتباطات مابین پردازنده می‌شود و این امر منجر به خطی ننودن تسریع بخش موازی کد موجود در کتابخانه OpenMp می گردد. البته باید توجه داشت که حتی اگر بخشی از برنامه به صورت موازی اجرا گردد و این بخش حداقل نیمی از کل زمان اجرائی را ثبت کند، باز هم با بهبود قابل توجهی در زمان اجرا مواجه خواهیم شد و این در کاربردهای مختلف سیستم های زمان واقعی یا سیستم های بهینه سازی که با حجم زیادی از داده ها سر و کار دارند بسیار حائز اهمیت می باشد. به عنوان مثال در گراف شبکه‌های اجتماعی که ما با گره‌های و یال‌های بسیار زیادی سر و کار داریم و گراف های موجود بسیار بزرگ می باشند استفاده از روش های موازی سازی برای تسویه عملیات پردازش گراف در کارایی این الگوریتم ها بسیار تاثیرگذار می‌باشد.

بحث و نتیجه‌گیری

استفاده از یافتن مجموعه غالب متصل، به دلیل کاربردهای آن چون یافتن گروه‌ها در شبکه‌های اجتماعی، شبکه‌های زیستی و شبکه‌های بیماری شناختی افزایش یافته‌است. هدف دومین شرط این است که با حداقل تبادل اطلاعات بین پردازشگرهای مختلف عملیات انجام شود. روشهای تقسیم بندی گراف‌ها می‌توانند در این عملیات مفید واقع شوند و به این ترتیب که ابتدا شبکه المان محدود را به‌صورت گراف نشان دهیم و سپس گراف را به قسمتهای مساوی تقسیم کنیم. از آنجایی که یافتن مجموعه غالب متصل گراف جزو مسائل سخت است، راه‌حل‌های عملی آن از روش‌های مکاشفه‌ای به دست آمده‌اند. دو گروه گسترده از این روش‌ها وجود دارد: محلی و سراسری.

این تحقیق با مدنظر قرار دادن امکانات موازی سازی چندنخی در کتابخانه OpenMp، در پی ارائه روشی جدید بوده که سرعت و دقت الگوریتم بخش‌بندی گراف را با استفاده از این فنّاوری افزایش می‌دهد. ما برخی از عملگرهای الگوریتم یافتن مجموعه غالب متصل گراف را با معماری جدید برای دستیابی به عملکرد بهتر و بهینه موازی سازی کرده‌ایم. گوریتم یافتن مجموعه غالب متصل چندمرحله‌ای در یک یا چند مرحله انجام می‌شود. هر مرحله اندازه گراف را با از بین بردن رأس‌ها و یال‌ها کاهش می‌دهد، گراف‌های کوچک‌تر را یافتن مجموعه غالب متصل می‌کند، نهایتاً به عقب برمی‌گردد و بخش‌های گراف اصلی را تصحیح و بازسازی می‌کند. طیف گسترده‌ای از روش‌های تصحیح و بخش‌بندی را می‌توان در شمای کلی روش چند مرحله‌ای قرار داد. در بسیاری از موارد، این روش می‌تواند سرعت اجرای بالا و نتایج با کیفیت بالا را هم‌زمان به دست دهد. در این تحقیق یک نسخه موازی جدید از بخش‌بندی گراف ارائه خواهد شد که در بخش محاسبه زیر درخت‌های گراف دارای قابلیت اجرای موازی است و این بخش بر اساس کتابخانه موازی OpenMP اجرا می‌گردد که در آن هر یک از بخش‌های موازی تحت عنوان یک نخ تعریف شده و از قدرت اجرای موازی سیستم‌های چندپردازنده بهره می‌برد.

با توجه به اینکه بعد از هر مرحله یافتن مجموعه غالب متصل، گراف اولیه به دو زیر گراف کاملا مجزا تقسیم می‌گردد، می‌توان عملیات موازی سازی را به خوبی بر روی الگوریتم یافتن مجموعه غالب متصل گراف اعمال نموده و هر بخش را به پردازنده مستقلی سپرد تا بنابر اصل تقسیم و غلبه بتوان اجرای بخش‌های مستقل را موازی اجرا نمود و از امکانات چندین پردازنده بصورت توام استفاده نمود. برای این کار در این تحقیق از کتابخانه موازی سازی OpenMp با قابلیت استفاده از امکانات سیستم‌های چندپردازنده‌ای بهره می‌بریم تا کارایی الگوریتم‌های یافتن مجموعه غالب متصل گراف را بهبود بخشیم.

الگوریتم ما با تقسیم پردازشگرهای P به دو مجموعه P_1 , P_2 با اندازه‌های مختلف متفاوت بیش تر از ۱ آغاز می‌شود. مجموعه‌های P_1 , P_2تقسیم رأس‌ها را به وجود می‌آورند. هدف اولیه ما کاهش تعداد لبه‌های متصل به رئوس در P_1 , P_2 است. اگر $P_1 = \phi, P_2 = \phi$ باشد، ما کاری برای انجام دادن نداریم، پس پروسه را متوقف می‌سازیم. درغیر این صورت روش

زیر را ادامه می‌دهیم. ابتدا برای هر بخش یک پردازشگر انتخاب می‌کنیم. می‌گوییم $L_1 \in P_1$ و $L_2 \in P_2$ می‌باشد و نخ اجرایی بخش محسوب می‌شود. اگر $S \in P_i$ باشد، می‌توانیم بگوییم که نخ اجراییS، L_i است.

هر پردازشگر در $P_1 \cup P_2$ ،مقادیر D تمام رئوس را محاسبه می‌کند و سپس آن‌ها را برای نخ اجرایی گزارش می‌دهد. بعد از آن هر نخ اجرایی یک حلقه نا مشخص با بیش ترین مقدار D، انتخاب می‌کند و یکدیگر را از انتخاب خود مطلع می‌سازند، سپس دو راس را علامتگذاری می‌کنند و آن‌ها را در لیستی به همراه نتایج خود ذخیره می‌کنند. سپس مقادیر D، رئوس نامشخص را با استفاده از معادلات مربوطه به روز رسانی می‌کنند.

از این معادلات دریافتیم که آن‌ها به لیست‌های مجاور هر دو راس نیاز دارند. نخ‌های اجرایی، این اطلاعات را از پردازشگری تقاضا می‌کنند که برای رئوس انتخاب شده تعیین شده اند. پس از دریافت اطلاعات، مقادیر نسبی D را به روز رسانی می‌کنند. آن‌ها این روند علامت گذاری رئوس و به روز رسانی مقادیر D تا زمانی که تمام رأس‌ها برای پردازشگرهای مشخص شده در P_1 , P_2 تعیین شوند، تکرار می‌کنند. سپس نخ‌های اجرایی با استفاده از روند مشابه مثل الگوریتم سریال تصمیم می‌گیرند که چه راس هایی را جایگزین کنند. آن‌ها پردازشگرهای تصمیماتشان را آگاه می‌سازند و پردازشگرها، لیست‌های مجاور انتخاب شده را جایگزین می‌کنند. بعد از جا به جایی رئوس، هر پردازشگر،دارای تعداد یکسانی رئوس می‌باشند. پردازشگر این الگوریتم کلی را تکرار می‌کند تا تعداد لبه‌های خارجی بین P_1 , P_2 نتوانند کاهش یابد. سپس مجموعه‌های P_1 , P_2 را به‌صورت موازی برای این الگوریتم به کار می‌گیرند.

برای ارزیابی روش پیشنهادی از مقایسات ای بر حسب تغییر در تعداد واحدهای اجرائی بر روی سیستم های چند پردازنده ای با استفاده از کتابخانه OpenMP استفاده می گردد. ما عملیات یافتن مجموعه غالب متصل را بر روی گراف های مختلفی اجرا می‌کنیم که هر یک دارای منحصر به فردی می باشد و هر یک مربوط به یک کاربرد واقعی میباشد که این داده‌ها به بنچمارک هایی برای مقایسه و ارزیابی الگوریتم های مبتنی بر گراف در بسیاری از مطالعات تبدیل شدند و در این مطالعه نیز به صورت مشاهده مورد استفاده قرار گرفته است. عملکرد به دست آمده با استفاده از یافتن مجموعه غالب متصل گراف حاصل در حالت استفاده از کتابخانه OpenMp و اجرای موازی عملیات بیان شده، در مقایسه با روش یافتن مجموعه غالب متصل دارای برتری می‌باشد. نتایج نشان می‌دهد که روش پیشنهادی بر حسب تمامی مقادیر مختلف برای بخش‌های ایجاد شده توسط الگوریتم پیشنهادی برای تمامی نمونه ها دارای برتری قابل توجهی به حالت سریال دارد. همچنین باید به این نکته توجه نمود که با افزایش تعداد یال و بزرگ شدن گراف ها حجم عملیات های افزایش می‌یابد که اجرای موازی آن در حالت سریال با افزایش تعداد بخش، تسریع بیشتری را نشان می دهد. به عنوان نمونه گراف ورودی CANT که دارای تعداد یال بسیار بیشتری نسبت به نمونه های دیگر بوده عملیات پردازش را در زمان بسیار بیشتری انجام داده و در عین حال میزان تسریع نسبت به حالت سریال برای این نمونه ورودی بیشتر بوده است. نتایج شبیه‌سازی بر روی داده‌های استاندارد، اعتبار روش پیشنهادی را تأیید می‌کند. نتایج نشان می‌دهد که روش پیشنهادی به‌صورت کلی بسیار کارآمد و دارای تسریع مناسبی نسبت به حالت سریال می‌باشد.در مطالعات آتی از سایر روش‌های موازی سازی برای اجرای موازی‌سازی یافتن مجموعه غالب متصل گراف استفاده می‌گردد و سعی می‌گردد تا با استفاده از کتابخانه MPI از قدرت اجرای موازی بر روی چند سیستم توزیع شده بهره برد.

منابع

1. Yang, Z., Shi, M. & Wang, W. Greedy approximation for the minimum connected dominating set with labeling. *Optim Lett* (2020).

2. Nguyen, M.H., Hà, M.H., Nguyen, D.N. *et al.* Solving the *k*-dominating set problem on very large-scale networks. *Comput Soc Netw* **7,** 4 (2020).

3. Nguyen MH, Hà MH, Hoang DT, Nguyen DN, Dutkiewicz E, Tran T. An efficient algorithm for the k-dominating set problem on very large-scale networks (extended abstract), 8th international conference, CSoNet 2019, Ho Chi Minh City, Vietnam, November 18–20, 2019, proceedings. Lecture notes in computer science, vol. 11917. p. 74–6.

4. Wang Y, Cai S, Chen J, Yin M. A fast local search algorithm for minimum weight dominating set problem on massive graphs. In: Twenty-seventh international joint conference on artificial intelligence (IJCAI). 2018. p. 1514–22.
5. Ugurlu O, Tanir D. A hybrid genetic algorithm for minimum weight dominating set problem. In: Zadeh L, Yager R, Shahbazova S, Reformat M, Kreinovich V, editors. Recent developments and the new direction in soft-computing foundations and applications, vol. 361., Studies in fuzziness and soft computingBerlin: Springer; 2018. p. 137–48.
6. Campan A, Truta TM, Beckerich M. Approximation algorithms for dd-hop dominating set problem. In: 12th international conference on data mining. 2016. p. 86–91.
7. Senouci, O., Aliouat, Z. & Harous, S. DCA-DS: A Distributed Clustering Algorithm Based on Dominating Set for Internet of Vehicles. *Wireless Pers Commun* (2020).
8. de Berg, M., Kisfaludi-Bak, S., Woeginger, G.: The complexity of dominating set in geometric intersection graphs. Theor. Comput. Sci. **769**, 18–31 (2019)
9. de Berg M., Kisfaludi-Bak S. (2020) Lower Bounds for Dominating Set in Ball Graphs and for Weighted Dominating Set in Unit-Ball Graphs. In: Fomin F., Kratsch S., van Leeuwen E. (eds) Treewidth, Kernels, and Algorithms. Lecture Notes in Computer Science, vol 12160.
10. Wu, J., & Li, H. (2001). A dominating-set-based routing scheme in Ad Hoc wireless networks. *Telecommunication Systems*, *18*(1), 13–36.
11. 13. Erey, A. Uniform Length Dominating Sequence Graphs. *Graphs and Combinatorics* (2020).
12. Shi, Y., Xu, X., Lu, C., & Chen, S. (2016). Distributed and weighted clustering based on d-Hop dominating set for vehicular networks. *KSII Transactions on Internet and Information Systems*, *10*, 1661–1678.
13. Chinnasamy, A., Sivakumar, B., Selvakumari, P., & Suresh, A. (2019). Minimum connected dominating set based RSU allocation for smartCloud vehicles in VANET. *Cluster Computing*, *22*, 12795–12804.
14. Gologranc, T., Jakovac, M., Kos, T., Marc, T.: On graphs with equal total domination and Grundy total domination number (2019).
15. Henning, M.A., Klavžar, S., Rall, D.F.: Total version of the domination game. Graphs Combin. **31**, 1453–1462 (2015).
16. Iršič, V.: Effect of predomination and vertex removal on the game total domination number of a graph. Discret. Appl. Math. **257**, 216–225 (2019)
17. Jiang, Y., Mei, L.: Game total domination for cyclic bipartite graphs. Discret. Appl. Math. **265**, 120–127 (2019).
18. Košmrlj, G.: Realizations of the game domination number. J. Comb. Optim. **28**, 447–461 (2014).
19. Lin, J.C.-H.: Zero forcing number, Grundy domination number, and their variants. Linear Algebra Appl. **563**, 240–254 (2019).
20. Mohanty, J.P., Mandal, C., Reade, C.: Distributed construction of minimum connected dominating set in wireless sensor network using two-hop information. Comput. Netw. **123**, 137–152 (2017)
21. Mohanty, J.P., Mandal, C., Reade, C., Das, A.: Construction of minimum connected dominating set in wireless sensor networks using pseudo dominating set. Ad Hoc Netw. **42**, 61–73 (2016)
22. Yu, F., Xia, X., Li, W., Tao, J., Ma, L., Cai, Z.-Q.: Critical node identification for complex network based on a novel minimum connected dominating set. Soft Comput. **21**, 1–9 (2016).

23. Fahong, Y., Xia, X., Li, W., Tao, J., Ma, L., Cai, Z.: Critical node identification for complex network based on a novel minimum connected dominating set. Soft Comput. **21**(19), 5621–5629 (2017)
24. Hedar, A.-R., Ismail, R.: Hybrid genetic algorithm for minimum dominating set problem. In: Computational Science and Its Applications–ICCSA 2010, pp. 457–467. Springer, Berlin (2010)
25. Li, R., Hu, S., Gao, J., Zhou, Y., Wang, Y., Yin, M.: Grasp for connected dominating set problems. Neural Comput. Appl. **28**, 1–9 (2016)
26. Yadav, A.K., Yadav, R.S., Singh, R., Singh, A.K.: Connected dominating set for wireless ad hoc networks: a survey. Int. J. Eng. Syst. Modell. Simul. **7**(1), 22–34 (2014)
27. Coelho, R.S., Moura, P.F.S., Wakabayashi, Y.: The k-hop connected dominating set problem: hardness and polyhedra. Electron. Notes Discrete Math. **50**, 59–64 (2015).
28. Mohanty, J.P., Mandal, C.: A distributed greedy algorithm for construction of minimum connected dominating set in wireless sensor network. In: Applications and Innovations in Mobile Computing (AIMoC), 2014, pp. 104–110. IEEE (2014)
29. Kui, X., Wang, J., Zhang, S.: A data gathering algorithm based on energy-balanced connected dominating sets in wireless sensor networks. In: 2013 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1139–1144. IEEE (2013)
30. Kamei, S., Kakugawa, H.: A self-stabilizing 6-approximation for the minimum connected dominating set with safe convergence in unit disk graphs. Theor. Comput. Sci. **428**, 80–90 (2012)
31. Dagdeviren, Z.A., Aydin, D., Cinsdikici, M.: Two population-based optimization algorithms for minimum weight connected dominating set problem. Appl. Soft Comput. **59**, 644–658 (2017).