گزارش الگوریتم تکاملی مبتنی بر متقاطع یکپارچه برای رنگ‌آمیزی نمودارهای دارای وزن راس

محمد خورشیدی روزبهانی ۴۰۲۱۵۷۴۱۰۰۲۰۱۳

شارا شاهوردیان ۴۰۲۱۵۷۴۱۰۰۲۰۳۲

چکیده

رنگ‌آمیزی نمودار یکی از مسائل اصلی بهینه‌سازی است که به طور گسترده در ادبیات مورد مطالعه قرار گرفته است. در این مطالعه، یک الگوریتم تکاملی نوآورانه به نام الگوریتم تکاملی مبتنی بر متقاطع یکپارچه با عملگر متقاطع منحصر به فرد خود و تکنیک جستجوی محلی برای رنگ‌آمیزی نمودارهای دارای وزن راس پیشنهاد می‌شود. عملگر متقاطع یکپارچه به منظور استفاده از اطلاعات خاص حوزه در افراد و تکنیک جستجوی محلی به منظور کاوش در حلقه‌های مجاور با استفاده از عملیات‌های تعویض وزن هدفمند می‌باشد. عملکرد کار پیشنهادی بر روی مجموعه داده‌های مصنوعی و نمونه‌های DIMACS با مقایسه آن با الگوریتم‌های تکاملی پیشرو از ادبیات ارزیابی می‌شود. مطالعه تجربی نشان می‌دهد که الگوریتم ما در ٪۷۱ موارد آزمایشی بهتر از کار مرتبط عمل کرده است و در ٪۱۷ موارد آزمایشی مصنوعی به نتیجه مشابهی می‌رسد. آزمایش‌های انجام شده بر روی نمونه‌های DIMACS نشان می‌دهد که الگوریتم ما تعداد بهترین رنگ‌ها را در ۷۰ از ۷۳ نمودار پیدا می‌کند، بنابراین کار پیشنهادی در زمان معقول موفق به رنگ‌آمیزی نمودارهای دارای وزن راس می‌شود.

۱ مقدمه

نظریه رنگ‌آمیزی نمودار به تقسیم یک مجموعه از راس‌ها به کلاس‌های رنگی جداگانه تحت شرطی می‌پردازد که هیچ راسی که یال مشترکی دارد نمی‌تواند به یک کلاس تخصیص یابد. هدف مسئله کلاسیک رنگ‌آمیزی نمودار، تعیین کوچکترین مقدار رنگ k برای به‌دست آوردن یک راه‌حل قانونی است. مسئله رنگ‌آمیزی-k نمودار (k-GCP) سعی دارد یک رنگ‌آمیزی ممکن برای یک نمودار با یک مقدار داده شده k پیدا کند. اگر راه‌حل بدون تضاد باشد، آنگاه یک رنگ‌آمیزی-k قانونی به‌دست می‌آید. مسئله رنگ‌آمیزی نمودار می‌تواند با حل یک سری مسائل رنگ‌آمیزی-k حل شود. با شروع با یک مقدار k کافی بزرگ، مقدار k می‌تواند هر بار که یک رنگ‌آمیزی قانونی پیدا شود، کاهش یابد. این فرایند تکرار می‌شود تا زمانی که یک راه‌حل غیرقانونی به‌دست آید. هدف اصلی k-GCP کمینه‌سازی تعداد یال‌های تضادی برای یک مقدار ثابت k است.

مسائل رنگ‌آمیزی نمودار با اهداف مختلفی وجود دارند. مسئله رنگ‌آمیزی متقارن [۱] شامل یک رنگ‌آمیزی قانونی با اختصاص رئوس به k کلاس رنگی مستقل است که تعداد رئوس در این کلاس‌ها حداکثر می‌تواند یکی اختلاف داشته باشد، در حالی که هدف مسئله کمینه‌سازی جمع رنگ‌های اختصاص یافته به رئوس است. همچنین، وزن‌ها می‌توانند به رئوس در مسائل رنگ‌آمیزی نمودار اضافه شوند [۲]. هدف مسئله رنگ‌آمیزی رأسی وزن‌دار، به‌دست آوردن یک رنگ‌آمیزی قانونی رنگی-k با هدف کمینه‌سازی جمع هزینه‌های کلاس‌های رنگ آن است. هزینه یک کلاس رنگی توسط رأسی که وزن بیشتری در کلاس دارد، تعیین می‌شود.

مسئله رنگ‌آمیزی نمودار به طور معمول برای مدل‌سازی مسائل واقعی مانند برنامه‌ریزی زمان، تخصیص منابع و تخصیص ثبت‌ها [۳]–[۵] استفاده می‌شود. اکثر این مسائل دارای تعداد محدودی منابع هستند. از آنجا که رنگ‌آمیزی-k مقدار رنگ ثابت k را در نظر می‌گیرد، این مقدار می‌تواند به تعداد منابع موجود در سیستم اشاره کند، بنابراین رنگ‌آمیزی-k می‌تواند برای حل این مسائل استفاده شود. در اکثر موارد، تعداد رنگ‌ها، ،k ممکن است کافی نباشد تا یک رنگ‌آمیزی قانونی به‌دست آید، بنابراین برخی از رئوس بدون رنگ خواهند ماند. اهمیت رئوس ممکن است برابر نباشد، بنابراین از یک مقدار وزن برای نشان دادن اهمیت آن‌ها استفاده می‌شود. کار پیشنهادی ما مسئله رنگ‌آمیزی-k (k-GCP) را با استفاده از یک گراف دارای وزن رأسی با هدف کمینه‌سازی جمع وزن کل رئوس بدون رنگ برای یک مقدار داده شده k در نظر می‌گیرد. مسئله رنگ‌آمیزی-k رأسی با وزن از یک گراف بدون جهت و دارای وزن G=(V،E،w) استفاده می‌کند، که در آن V مجموعه رئوس را نشان می‌دهد، E مجموعه یال‌ها را نشان می‌دهد و w مجموعه مقادیر وزن رئوس در V است که تاکیدی بر اهمیت آن‌ها دارند. هدف k-GCP انجام رنگ‌آمیزی

رئوس در V با استفاده از یک تعداد پیش‌تعریف شده از رنگ‌ها است. اگر تعداد داده شده از رنگ‌ها نتواند همه رئوس را رنگ‌آمیزی کند، برخی از رئوس بدون رنگ خواهند ماند. رئوس بدون رنگ به عنوان رئوس تضادی تعریف می‌شوند. تابع تناسب f (k) برابر با جمع وزن کل رئوس بدون رنگ هنگام استفاده از یک تعداد پیش‌تعریف شده از k رنگ است. هدف k-GCP کمینه‌سازی مقدار تابع تناسب f (k) است.

مسئله رنگ‌آمیزی نمودار به عنوان یک مسئله NPComplete اثبات شده است [۶] و بسیاری از روش‌های هیوریستیکی برای مسئله رنگ‌آمیزی نمودار [۷]–[۱۱]، مسئله رنگ‌آمیزی متقارن [۱۲]، [۱۳]، مسئله رنگ‌آمیزی-k [۱۴]، [۱۵]، مسئله کمینه‌سازی جمع رنگ‌ها [۱۶] و مسئله رنگ‌آمیزی رأسی وزن‌دار [۱۷] در ادبیات پیشنهاد شده‌اند. در این مطالعه، یک الگوریتم تکاملی ترکیبی [۱۸] برای مسئله رنگ‌آمیزی-k رأسی وزن‌دار پیشنهاد می‌شود. الگوریتم ما به نام الگوریتم تکاملی مبتنی بر متقاطع یکپارچه (InCEA) با عملگر متقاطع جدید خود و تکنیک جستجوی محلی است. عملگر متقاطع یکپارچه گروه بیشینه‌ای از رئوس بدون تضاد را به کلاس‌های رنگی فرزندان با استفاده موفقیت‌آمیز از اطلاعات خاص مسئله در والدین تقسیم می‌کند. دو کلاس رنگی انتخاب شده به‌طور تصادفی از والدین به‌طور تدریجی ترکیب می‌شوند تا هر کلاس رنگی از فرزندان شکل گیرد. رئوس تضادی به استخر انداخته می‌شوند و هر بار که یک کلاس رنگی جدید از فرزندان ایجاد می‌شود، رئوس در استخر امتحان خود را برای پیدا کردن یک کلاس رنگی بدون تضاد می‌دهند. اگر در پایان عملگر متقاطع، رئوسی در استخر موجود باشند، تکنیک جستجوی محلی سعی می‌کند این رئوس را به یکی از کلاس‌های رنگی فرزندان با استفاده از یک عملیات تعویض وزن‌دار قرار دهد تا مقدار تابع تناسب کمینه شود.

در مطالعه تجربی ما، الگوریتم InCEA را با کارهای مرتبط از ادبیات مقایسه کردیم. نتایج به دست آمده از مجموعه داده‌های مصنوعی تولید شده و نمونه‌های DIMACS نشان می‌دهد که InCEA در بیشتر موارد آزمایشی از نظر مقادیر تناسب و زمان محاسبات عملکرد کارهای مرتبط را برتری می‌بخشد. عملکرد الگوریتم‌ها بر روی ۷۳ نمونه DIMACS که گراف‌های چالش برای مسئله رنگ‌آمیزی نمودار هستند، نیز در این مطالعه ارائه شده است. از آنجا که تعداد کمینه رنگ‌های مورد نیاز برای رنگ‌آمیزی گراف‌ها برای این بنچمارک‌ها در ادبیات پیدا نشده است، این مقاله همچنین کمینه تعداد رنگ‌های استفاده شده برای رنگ‌آمیزی این نمونه‌ها را گزارش می‌دهد. همانطور که در مطالعه تجربی اشاره شده، الگوریتم تکاملی ما می‌تواند در بیشتر موارد آزمایشی ارائه شده به سرعت بسیار خوبی نتایج بهتری را نسبت به الگوریتم‌های موجود در ادبیات به دست آورد.

مهم‌ترین مشارکت‌های این کار می‌تواند به شرح زیر باشد:

- عملگر متقاطع یکپارچه به منظور استفاده از اطلاعات خاص مسئله در افراد با کمک یک استخر و یک عملیات جستجوی معکوس هدفمند استفاده می‌شود.
- عملیات تعویض وزن‌دار در تکنیک جستجوی محلی به منظور کاوش در حلقه‌های مجاور و افزایش فرصت برای رسیدن به بیشینه جهانی هدفمند است.
- هیچ محاسبات اضافی برای تکنیک جستجوی محلی یا محاسبه تناسب لازم نیست زیرا استخر از قبل رأس(های) بدون رنگ را نگه می‌دارد، بنابراین الگوریتم از بار جستجوهای جامع خلاص می‌شود.

در بقیه مقاله، ابتدا مطالعات مرتبط و تعریف مسئله مسئله رنگ‌آمیزی-k به ترتیب در بخش‌های ۲ و ۳ ارائه می‌شوند. در بخش ۴، جزئیات اجزای کار پیشنهادی ما را توضیح می‌دهیم. سپس، عملکرد الگوریتم ما را با الگوریتم‌های موجود در ادبیات بر روی نمونه‌های آزمایشی مختلف در بخش ۵ مقایسه و بررسی می‌کنیم. در نهایت، خلاصه کار پیشنهادی خود را ارائه می‌دهیم و پیشنهاداتی برای جهت‌های آینده ممکن را در بخش ۶ ارائه می‌دهیم.

۲ کار مرتبط

بسیاری از مسائل واقعی مانند برنامه‌ریزی زمان، تخصیص ثبت‌ها و تخصیص منابع می‌توانند با مسئله رنگ‌آمیزی نمودار نمایش داده شوند، جایی که رئوس معرفی‌کننده اشیاء و یال‌ها محدودیت‌ها را نشان می‌دهند. در تمام این مسائل، تعداد محدودی از منابع (رنگ‌ها) وجود دارد، بنابراین برخی از رئوس بدون رنگ خواهند ماند. وزن‌ها می‌توانند به رئوس اضافه شوند تا اهمیت آن‌ها را نشان دهند، و یک گراف دارای وزن رأسی می‌تواند برای مدل‌سازی این مسائل استفاده شود.

در مسئله تخصیص ثبت‌ها [۱۹]، هدف این است که حداکثر تعداد متغیرها را به حداقل تعداد ثبت‌ها اختصاص داد تا متغیرها به سرعت بالایی توسط واحد پردازش مرکزی (CPU) دسترسی داشته باشند. مسئله می‌تواند با استفاده از یک گراف بدون جهت نمایش داده شود، جایی که متغیرها توسط رئوس نشان داده می‌شوند و مداخلات بین متغیرها توسط یک یال نشان داده می‌شود. هدف این مسئله کمینه کردن تعداد

ثبت‌های استفاده شده (رنگ‌ها) و انتخاب متغیرهایی است که کمتر دسترسی دارند، که می‌تواند به طور مستقیم برای انتخاب رئوس بدون رنگ و محاسبه تابع تناسب در k-GCP [۲۰] اعمال شود.

مسئله تخصیص منابع نیز می‌تواند با استفاده از k-GCP حل شود. یکی از مهم‌ترین مسائل تخصیص منابع در شبکه‌های کامپیوتری [۲۱]–[۲۵]، تخصیص پهنای باند در شبکه‌های بی‌سیم است. در این مسئله، شبکه با سنسورهای خود می‌تواند به عنوان یک گراف بدون جهت و رئوس نمایش داده شود. اگر فاصله بین دو سنسور بیشتر از یک حد پیش‌تعیین شده باشد، این سنسورها نمی‌توانند از همان باند استفاده کنند، بنابراین رئوس نماینده آنها توسط یک یال متصل می‌شوند. هدف این مسئله کمینه کردن تعداد باندها (رنگ‌ها) برای تمامی سنسورها است. همچنین وزن رئوس کیفیت سنسورها را نشان می‌دهد.

برنامه‌ریزی یک مسئله چالش‌برانگیز برای آموزش [۲۶]، محاسبات [۲۷]، تولید [۲۸] و ارتباطات [۲۹]، [۳۰] است. به طور کلی، مسائل بر روی یک عنصر مانند یک دانش‌آموز، یک تولید، یک کارمند یا یک کار با ویژگی‌های مختلف خود تمرکز دارند. عناصر به عنوان رئوس نمایش داده می‌شوند و تضادها بین این عناصر با یال‌ها در گراف نمایش داده می‌شوند. اهمیت عناصر با وزن رئوس نشان داده می‌شود. زمان، ماشین‌ها، منابع، وسایل نقلیه به عنوان رنگ‌ها استفاده می‌شوند و تابع هدف می‌تواند بیشینه‌سازی کارایی زمانی، بهره‌وری منابع و غیره باشد. بنابراین، مسئله رنگ‌آمیزی نمودار یک انتخاب مناسب برای مسائل برنامه‌ریزی است [۳۱].

هنگامی که مسائل واقعی با استفاده از مسئله رنگ‌آمیزی نمودار مدل‌سازی می‌شوند، بسیاری از روش‌های دقیق [۳۲]–[۳۴] و هیوریستیک [۳۵]–[۳۸] به دلیل پیچیدگی NP-Hard آن [۶]، [۳۹]، [۴۰] قابل استفاده هستند. اگرچه الگوریتم‌های دقیق پیشنهادی برای حل مسئله رنگ‌آمیزی نمودار [۴۱] می‌توانند بهترین راه‌حل‌ها را برای نمونه‌های کوچک پیدا کنند، اما برای نمونه‌های بزرگ هزینه زیادی از نظر مصرف حافظه و زمان دارند [۴۲].

الگوریتم تکاملی یکی از روش‌های هیوریستیک برای مسئله رنگ‌آمیزی نمودار است [۲]. الگوریتم‌های تکاملی فرایند طبیعی را با عملگرهای متقاطع و جهش شبیه‌سازی می‌کنند. این الگوریتم‌ها با یک جمعیت سر و کار دارند که تعداد پیش‌تعریف شده‌ای از راه‌حل‌های کاندید که به عنوان افراد برای یک گراف دارای وزن رأسی داده شده، نگه داشته می‌شوند [۴۳].

در ادبیات، الگوریتم‌های تکاملی وجود دارند که مسائل واقعی را که به k-GCP رمزگذاری شده‌اند را حل می‌کنند، مانند الگوریتم تکاملی ترکیبی (HEA) [۴۴] و الگوریتم ممتاز به منظور هزینه (COMA) [۴۵].

HEA و COMA جمعیت اولیه خود را با استفاده از سه معیاری که در [۴۴] پیشنهاد شده‌اند، ایجاد می‌کنند. جزئیات این معیارها در بخش IV-A ذکر شده است. هنگامی که جمعیت‌های اولیه ایجاد شدند، هر دو HEA و COMA اپراتورهای متقاطع خود را برای بهبود افراد خود اعمال می‌کنند. الگوریتم‌ها دو والدین را از جمعیت خود انتخاب کرده و کلاس‌های رنگی این والدین را ترکیب می‌کنند تا فرزندی ایجاد کنند. انتخاب کلاس‌های رنگ در این الگوریتم‌ها متفاوت است. HEA از عملگر تقسیم بدون تضاد (CFPX) استفاده می‌کند که کلاس‌های رنگ را با بیشترین زیرمجموعه رأس بدون تضاد انتخاب می‌کند. از سوی دیگر، COMA کلاس رنگ را با بیشترین وزن زیرمجموعه بدون تضاد انتخاب می‌کند و عملگر متقاطع آن به نام عملگر تقاضای گرایی هزینه (COPX) است.

فرزندانی که توسط CFPX یا COPX ایجاد می‌شوند، تضمین نمی‌کنند که یک راه‌حل بدون تضاد تولید کنند. HEA و COMA اپراتور جستجوی محلی را که در [۴۴] پیشنهاد شده است، اعمال می‌کنند. آنها یکی از رأس‌های تضادی را از یک کلاس رنگی از فرزند با استفاده از مقادیر درجه و وزن رأس‌ها انتخاب می‌کنند. هر دو الگوریتم تمامی کلاس‌های رنگ را برای قرار دادن این رأس تضادی بازدید می‌کنند و آن را به کلاس رنگی اختصاص می‌دهند که باعث بیشینه کاهش در مقدار تابع تناسب شود. تعداد ایتریشن‌ها در این مرحله جستجو به ۱۰٪ از کل تعداد تضادها در HEA محدود شده است، در حالی که COMA چنین محدودیتی ندارد و برای هر رأس تضادی در فرزند تمامی کلاس‌های رنگ را بازدید می‌کند. تکنیک‌های مبتنی بر تضاد و مبتنی بر هزینه در HEA معرفی شده‌اند، و کمترین مقدار تابع تناسب به دست آمده از این دو رویکرد به عنوان مقدار تابع تناسب فرزند انتخاب می‌شود. در حالی که COMA از تکنیک‌های مبتنی بر تضاد، مبتنی بر هزینه و مبتنی بر معیار استفاده می‌کند تا مقدار تابع تناسب فرزند را محاسبه کند.

۳ بیان مسئله

گراف وزن‌دار بدون جهت $G(V, E, w)$ را در نظر بگیرید، جایی که V و E مجموعه‌های رئوس و یال‌ها هستند، و w مجموعه‌ی مقادیر وزن رئوس شامل V است. اگر V دارای n راس باشد، اندازه مجموعه وزن، $|w|$، همچنین n است و E حداکثر می‌تواند $\frac{n\times(n-1)}{2}$ یال داشته باشد، که $0 \leq |E| \leq \frac{n\times(n-1)}{2}$

هر رأس $v \in V$ به یک کلاس رنگ C_i تعلق دارد، که یکی از مجموعه‌های مستقل جدایی شده از $C = \{C_1, C_2, C_3, \ldots, C_k\}$ است، به طوری

که $1 \leq k \leq n$.

اگر $u \in V$ مجاور $v \in V$ باشد، در این صورت یک یال $\{u, v\} \in E$ وجود دارد و u و v نمی‌توانند در یک رنگ باشند (۱). این به عنوان رنگ‌آمیزی قانونی یا ممکن-k شناخته می‌شود.

$$\forall v, \quad u \in C_i, \{u, v\} \notin E, \quad i = 1, 2, 3, \ldots, k \tag{۱}$$

برای یک مقدار داده شده از تعداد کلاس‌های رنگ k، هدف مسئله رنگ‌آمیزی-k ارائه یک کلاس رنگ برای هر رأس v به گونه‌ای است که شرط (۱) را رعایت کرده و یک رنگ‌آمیزی ممکن ارائه دهد. اگر v نتواند به یک کلاس رنگ اختصاص یابد، آنگاه v به عنوان یک رأس تضادی تعریف شده و غیررنگ‌آمیزی می‌شود. در این صورت، راه‌حل غیرقابل اجرا است و هدف مسئله رنگ‌آمیزی-k کمینه کردن مجموع وزن‌های رأس‌های تضادی با استفاده از $f(k)$ است (۲).

$$minimize \;\; f(k) = \sum w(v), \quad v \notin C \tag{۲}$$

۴ الگوریتم تکاملی مبتنی بر متقاطع یکپارچه

روش عمومی رویکرد ما بر مبنای الگوریتم ژنتیک به صورت کلی در الگوریتم ۱ آورده شده است. در هر نسل از الگوریتم، ترکیب یکپارچه (InCX) انجام می‌شود که در الگوریتم ۲ ارائه شده است. به عنوان یک قسمت از اپراتور ترکیب، عملیات جستجو به عقب که در الگوریتم ۳ آمده است، انجام می‌شود. بر اساس خروجی اپراتور ترکیب، که یک فرزند تکی است، کار ما به بهبود فرزند با استفاده از تکنیک جستجوی محلی که در الگوریتم ۴ ارائه شده است، هدفمند است.

الگوریتم ۱

الگوریتم ۲

الگوریتم ۳

الگوریتم ۴

۱.۴ نمایندگی انفرادی و نسل اولیه جمعیت

در الگوریتم پیشنهادی، هر فرد S_i در جمعیت با روش تقسیم [۴۶] نمایش داده می‌شود که $S_i = \{C_1, C_2, C_3, \ldots, C_k\}$ است و هر کلاس رنگ C_j شامل یک گروه از رئوس بدون تضاد است با در نظر گرفتن مجموع k کلاس رنگ.

جمعیت اولیه شامل تعداد پیش‌تعیین شده‌ای از راه‌حل‌های کاندیدایی است که با استفاده از معیار درجه لکه [۴۴] ایجاد شده‌اند، که رئوس را به سه روش مختلف مرتب می‌کند. معیار درجه لکه از وزن رئوس، $w(v_i)$، و درجه رئوس که با $d(v_i)$ نشان داده می‌شود، استفاده می‌کند. برای تنوع در جمعیت، درجات لکه برای ۴۰٪، ۴۰٪ و ۲۰٪ از کل رئوس با استفاده از (۳الف، ۳ب و ۳ج)، به ترتیب، تنظیم می‌شود.

$$S_1(v_i) = w(v_i) \times d(v_i) \tag{۳}$$

$$S_2(v_i) = w(v_i) \times d(v_i)^2 \tag{۴}$$

$$S_3(v_i) = w(v_i) \tag{۵}$$

رئوس بر اساس درجه لکه خود به ترتیب نزولی مرتب می‌شوند و به کلاس‌های رنگی فردانشان اضافه می‌شوند. برای تولید هر فرد، رأس با بیشترین درجه لکه از مجموعه‌ی رئوس ناشده انتخاب شده و به یک کلاس رنگ اضافه می‌شود. از اولین کلاس رنگ شروع شده و الگوریتم یک کلاس رنگ را پیدا می‌کند که رأس یک مجموعه بدون تضاد با سایر رئوسی که به همان کلاس تعلق دارند، فراهم می‌کند. رأس به اولین کلاس رنگ بدون تضاد اختصاص داده می‌شود. اگر هیچ کلاسی برای چنین رأسی یافت نشود، رأس به یک کلاس رنگ تصادفی اضافه می‌شود. این فرایند تکرار می‌شود تا همه‌ی رئوس به یک کلاس رنگ نگاشته شوند.

۲.۴ اپراتور متقاطع یکپارچه

در این کار، ما یک اپراتور ترکیب جدید ارائه می‌دهیم که هدف آن راهنمایی افراد برای رسیدن به بهینه جهانی با اطلاعات خاص مسئله در کلاس‌های رنگی می‌باشد. هدف از اپراتور ترکیب یکپارچه این است که کلاس‌های رنگی را از والدین ترکیب کرده و تعداد رئوس بدون تضاد را در هر کلاس رنگ افزایش دهد، به گونه‌ای که تعداد کل رئوس بدون رنگ بیشینه شود.

دو کلاس رنگ از والدین انتخاب شده و ترکیب می‌شوند تا احتمال یافتن یک گروه بزرگتر و بهتر از رئوس بدون تضاد افزایش یابد. اپراتور ترکیب یکپارچه ما یک استخر را برای نگه داشتن رئوس تضادی که نمی‌توانند به کلاس رنگی فعلی تعلق گیرند، پیشنهاد می‌دهد. فرزندان به طور معمول بیش از یک کلاس رنگ خواهند داشت، بنابراین رئوسی که به استخر پرتاب شده‌اند، فرصتی برای قرار گرفتن در کلاس‌های رنگی ایجاد شده پیش از این خواهند داشت. از طرف دیگر، اپراتور ترکیب یکپارچه ما یک عملیات جستجو به عقب برای کمینه کردن تعداد رئوس باقی‌مانده در استخر دارد که با قرار دادن رئوس در استخر به یک کلاس رنگی بدون تضاد، انجام می‌شود.

تحقیقات قبلی در ادبیات کلاس رنگ از یکی از والدین کلاس رنگ با بیشترین مجموعه بدون تضاد را انتخاب کرده و این مجموعه مستقیماً به فرزندان کپی می‌شود. آن‌ها در نظر نگرفتند که اطلاعات خاص مسئله را در هر دو کلاس رنگ به دلیل تضادها ترکیب کنند.

۱.۲.۴ اپراتور متقاطع مبتنی بر استخر

اپراتور ترکیب یکپارچه با انتخاب تصادفی دو تنظیمات والدین شروع می‌شود که والد اول و والد دوم به ترتیب با $S_1 = \{C_0^1, C_1^1, C_2^1, \ldots, C_{k-1}^1\}$ و $S_2 = \{C_0^2, C_1^2, C_2^2, \ldots, C_{k-1}^2\}$ نشان داده می‌شوند در الگوریتم ترکیب (مراجعه به الگوریتم ۲). اپراتور یک فرزند $S_0 = \{C_0, C_1, C_2, \ldots, C_{k-1}\}$ را ایجاد می‌کند و یک استخر P را آماده می‌کند که همچنین توسط تکنیک جستجوی محلی استفاده خواهد شد.

اپراتور ترکیب یکپارچه به صورت تکراری فرزندان را ایجاد می‌کند و در هر تکرار یک کلاس رنگ از فرزند که به عنوان Ci نمایش داده می‌شود، با استفاده از دو تقسیم تصادفی C_x^1 و C_y^2 از هر دو والد ایجاد می‌شود. رئوس در این تقسیم‌بندی‌ها گروه‌بندی شده و کلاس رنگی از فرزند با رئوس بدون تضاد ایجاد می‌شود. اگر رئوس تضادی وجود داشته باشند که نمی‌توانند به کلاس رنگ فعلی اختصاص یابند، الگوریتم سعی می‌کند این رئوس را به کلاس‌های رنگی از پیش تولید شده فرزند S_0 با استفاده از عملیات جستجو به عقب اختصاص دهد که در زیر بخش بعدی به تفصیل شرح داده شده است. اگر چنین کلاس رنگی وجود نداشته باشد، آن‌ها به استخر پرتاب می‌شوند تا با رئوسی که در تکرارهای بعدی انتخاب می‌شوند ترکیب شوند.

۲.۲.۴ عملیات جستجو در عقب

در کار اولیه ما، الگوریتم تکاملی مبتنی بر استخر (PBEA) [۴۷]، ما یک استخر ابتدایی برای ذخیره رئوسی که به کلاس‌های رنگی تخصیص نیافته‌اند، پیشنهاد دادیم. حتی اگر برخی از رئوس در استخر بتوانند به کلاس‌های رنگی از فرزندانی که در تکرارهای قبلی تولید شده‌اند تخصیص داده شوند، مکانیزم هوشمندی برای شناسایی آنها پیشنهاد نشده بود. به عبارتی، نگهداری از استخر به مشکل خواهد خورد زیرا رئوس تضادی جدید در هر تکرار به آن اضافه می‌شوند.

در این مطالعه، ما عملیات جستجو به عقب را به همراه استخر به عنوان بخشی از اپراتور ترکیب یکپارچه پیشنهاد می‌دهیم. عملیات جستجو به عقب هدفش حذف رئوسی است که با کلاس‌های رنگی از پیش تولید شده تضاد ندارند. به عبارت دیگر، این عملیات تعداد رئوس بدون تضاد را در هر کلاس رنگ افزایش می‌دهد و تعداد رئوس موجود در استخر را کاهش می‌دهد.

برای توضیح اجرای اپراتور ترکیب یکپارچه ما ،(InCX) یک نمونه گراف وزن‌دار و دو تنظیم والدین را در شکل ۱ و شکل ۲ در نظر می‌گیریم، به ترتیب. توجه داشته باشید که همان گراف و والدین در یک کار قبلی [۴۴] ارائه شده‌اند.

عکس ۱

از گام ۰ شروع می‌کنیم، کلاس رنگ دوم C_1^1 از والد اول S_1 و کلاس رنگ سوم C_2^2 از والد دوم S_2 به صورت تصادفی انتخاب می‌شوند. رئوس ۲، ۷، ۸ در C_1^1 و رئوس ۰، ۱، ۹ در C_2^2 ترکیب شده و به کلاس رنگ اول C_0 از فرزند S_0 اضافه می‌شوند. این رئوس به عنوان تخصیص داده شده علامت‌گذاری شده و از کلاس‌های رنگی والدین حذف می‌شوند که در مراحل بعدی رئوس نامرئی برای مراحل بعدی شوند. تضادهای بین رئوس در C_0 بر اساس گراف در شکل ۱ محاسبه می‌شوند. بیشترین رئوس دارای تضاد با ۳ تضاد، رئوس ۸ است بنابراین به طور مستقیم در استخر P انداخته می‌شود. رئوس ۸ سه یال با رئوس ۱، ۲ و ۹ دارد. پس از حذف رئوس ۸ از C_0، تضادهای رئوس ۱، ۲ و ۹ به ترتیب یک واحد کاهش می‌یابد. رئوس ۲، ۷ و ۱ بیشترین تعداد تضادها را دارند، بنابراین ارزش‌های وزن آن‌ها (در شکل ۱) مقایسه می‌شوند. از آنجا که رئوس ۷ ارزش وزن کمتری با ارزش وزن ۱ دارد، به عنوان رئوس با بیشترین تضاد انتخاب می‌شود و به استخر P انداخته می‌شود و از C_0 حذف می‌شود. رئوس باقی‌مانده با

تضاد تنها رئوس ۰ و ۱ هستند، جایی که رئوس ۱ به استخر P انداخته می‌شود و به دلیل داشتن وزن کمتر از C_0 حذف می‌شود. C_0 بدون تضاد می‌شود و اولین تکرار ترکیب پایان می‌پذیرد.

در تکرار بعدی، کلاس‌های رنگ اول C_0^1 و C_0^2 از والدین به صورت تصادفی انتخاب می‌شوند. رئوس بی‌نسبت ۳، ۴ و ۶ در $(C_0^1 \cup C_0^2)$ به عنوان تخصیص داده شده به کلاس رنگ دوم C_1 از S_0 اضافه می‌شوند و علامت‌گذاری می‌شوند. رئوس ۸، ۷ و ۱ در P نیز با رئوس در C_1 ترکیب می‌شوند. تضادهای رئوس محاسبه می‌شود و رئوس ۳، ۸ و ۶ به P انداخته می‌شوند تا کلاس رنگ C_1 بدون تضاد شود. از آنجا که فرزند S_0 یک کلاس رنگی C_0 را قبلاً تولید کرده است، رئوس بی‌نسبت در استخر P فرصتی برای قرارگیری در C_0 دارند. عمل جستجوی باز به قرار دادن رئوس در P به C_0 ادامه می‌دهد. رئوس ۳ با رئوس ۰ تضاد دارد، رئوس ۸ با رئوس ۲ و ۹ تضاد دارد، بنابراین آن‌ها در P نگه داشته می‌شوند. با این حال، رئوس ۶ با رئوس در C_0 تضادی ندارد، بنابراین به C_0 نقشه‌برداری می‌شود. در تکرار نهایی، دو کلاس رنگ باقیمانده که قبلاً انتخاب نشده‌اند به عنوان C_2^1 و C_1^2 انتخاب می‌شوند. در هر دو کلاس رنگ، تنها یک رئوس بی‌نسبت وجود دارد که رئوس ۵ است، بنابراین با رئوس ۳ و ۸ در دسترس در P ترکیب شده و به کلاس رنگ سوم C_2 از S_0 اضافه می‌شوند. بر اساس گراف داده شده در شکل ۱، یک مجموعه بدون تضاد برای C_2 به دست می‌آید. با تولید تعداد از پیش تعریف شده کلاس‌های رنگ (که در اینجا $k = 3$ است) برای S_0، عملیات ترکیب به پایان می‌رسد.

در پایان عملیات ترکیب یکپارچه، یک فرزند S_0 با ۳ کلاس رنگ و یک استخر P به دست می‌آید. زمانی که P شامل رئوس بی‌نسبت است، به این معناست که گراف داده شده نمی‌تواند با k رنگ رنگ‌آمیزی شود و روش جستجوی محلی پیشنهادی که در بخش ۲.۲.۴ توضیح داده شده، به S_0 و P اعمال می‌شود. در غیر این صورت، الگوریتم پیشنهادی با موفقیت تمام رئوس را رنگ‌آمیزی کرده است، بنابراین نیازی به اعمال روش جستجوی محلی نیست. یک سناریوی مثال در شکل ۲ نشان داده شده است.

برای گراف وزن‌دار مثال داده شده در شکل ۱، خروجی عملیات ترکیب یکپارچه ما (InCX) و سه خروجی از عملیات‌های ترکیبی ارائه شده در ادبیات در شکل ۳ نشان داده شده است، که شامل ترکیب بخشریزی بی‌تضاد ،(CFPX) ترکیب هزینه‌مند (COPX) و ترکیب مبتنی بر استخر (PBC) هستند. فرزندان تولیدشده توسط ،CFPX COPX و PBC هنوز رئوس بی‌رنگی دارند. CFPX و PBC نمی‌توانند رأس ۶ را رنگ‌آمیزی کنند که وزن آن ۳ است. COPX قادر به رنگ‌آمیزی رأس ۳ نیز نبود که وزن آن ۱ است. با این حال، InCX موفق به رنگ‌آمیزی همه رئوس می‌شود بدون اعمال روش جستجوی محلی.

CFPX و COPX همیشه کلاس‌های رنگی را انتخاب می‌کنند که دارای بیشترین زیرمجموعه بدون تضاد هستند، در حالی که PBC و InCX کلاس‌های رنگی را به صورت تصادفی از هر دو پدر انتخاب می‌کنند، بنابراین راه‌حل‌ها می‌توانند از ترکیب‌های مختلفی به دست آیند. در شکل ۲، PBC و InCX می‌توانند کلاس‌های رنگی از پدرها را به ۳۶ روش مختلف ترکیب کنند و از این ترکیبات ۳۶ راه‌حل بدست آورند. PBEA می‌تواند از ٪۵۰ از این ترکیبات یک راه‌حل بدون تضاد پیدا کند، ٪۲۲ از راه‌حل‌های بدون تضاد پس از PBC و ٪۲۸ از آنها پس از عملگر جستجوی محلی آن پیدا می‌شوند. در حالی که، InCEA می‌تواند از ٪۵۸ از این ترکیبات یک راه‌حل بدون تضاد بدست آورد و ٪۵۵ از این راه‌حل‌های بدون تضاد پس از عملگر InCX پیدا می‌شوند. این درصدها نشان می‌دهد که عملگر ترکیب جدید ما احتمال یافتن یک راه‌حل بدون تضاد را افزایش می‌دهد.

۳.۴ تکنیک جستجوی محلی

هدف از عملگر InCX کسب بزرگترین گروه‌های رأس بدون تضاد در هر کلاس رنگی از فرزند است. هنگام انتقال رئوس تضادی به استخر، ارزش‌های وزنی رئوس تنها در مواقعی مدنظر قرار می‌گیرد که نیاز به تصمیم‌گیری در میان بیش از دو رأس با تعداد تضاد یکسان وجود دارد. در پایان عملگر ترکیب یکپارچه ما، اگر هنوز رئوسی در استخر وجود داشته باشد، این رئوس حداقل با یکی از رئوس هر کلاس رنگی از فرزند تضاد دارند و ممکن است راه‌حلی بدون تضاد وجود نداشته باشد.

هدف الگوریتم ما کمینه کردن وزن کل رئوسی است که نمی‌توانند رنگ‌آمیزی شوند. از آنجا که عملگر InCX وزن رئوس را در نظر نمی‌گیرد، تکنیک جستجوی محلی هدفش کاهش مجموع وزن رئوسی است که رنگ‌آمیزی نشده‌اند با در نظر گرفتن راه‌حل‌های مجاور. در پایان عملگر ترکیب، رئوسی که رنگ‌آمیزی نشده‌اند در استخر حضور دارند، بنابراین تکنیک جستجوی محلی سعی می‌کند این رئوس را با عملیات جابجایی به کلاس‌های رنگی منتقل کند.

الهام گرفته شده از عملگر جهش سنتی SWAP که موقعیت دو ژن انتخاب شده را در نمایش ژن مبتنی بر ترتیب تعویض می‌کند، یک تکنیک جستجوی محلی جدید به نام Weighted-Swap (W-SWAP) پیشنهاد می‌دهیم. W-SWAP شامل جستجو و تعویض رئوس بین استخر و کلاس‌های رنگی است. اگر مجموع وزن رئوسی که قرار است از یکی از کلاس‌های رنگی جابجا شوند، کمتر از وزن رئوس در استخر باشد، عملیات جابجایی انجام می‌شود. در پایان جستجوی محلی، مجموع وزن رئوسی که رنگ‌آمیزی نشده‌اند کمینه می‌شود و همچنین احتمال وجود یک راه‌حل بدون تضاد وجود دارد.

در ابتدای تکنیک جستجوی محلی، رئوس در P بر اساس ارزش وزنی خود به ترتیب نزولی مرتب می‌شوند تا رئوس با بیشترین وزن اولویت

داشته باشند، اگر امکان وجود داشته باشد. برای هر v_p در P، الگوریتم مجموع وزن رئوس با v_p در همه‌ی k کلاس S_0 را محاسبه کرده و کمینه مجموع را پیدا می‌کند. اگر این مقدار کمتر از $w(v_p)$ باشد، عملیات جابجایی انجام می‌شود. رئوس در کلاس رنگی به P پرتاب شده و v_p به کلاس رنگی منتقل می‌شود. اگر چنین کلاسی وجود نداشت، v_p از P حذف شده و در یک لیست ممنوعه V_{tabu} قرار می‌گیرد که رئوس رنگ‌آمیزی نشده برای S_0 را نگه می‌دارد. جستجوی محلی ادامه می‌یابد تا هر v_p در P پردازش شود و P خالی شود. همه‌ی رئوس رنگ‌آمیزی نشده در V_{tabu} حضور دارند، بنابراین این لیست می‌تواند در محاسبه تناسب فرزند استفاده شود که در زیربخش بعدی شرح داده شده است. در نهایت، رئوس در V_{tabu} به صورت تصادفی به کلاس‌های رنگی اختصاص داده می‌شوند.

شکل ۵ تکنیک جستجوی محلی پیشنهادی را با استفاده از گراف داده شده در شکل ۴ نشان می‌دهد. فرض کنید عملگر جابجایی یکپارچه روی دو تنظیمات والدین اعمال شده و یک فرزند با پول غیرخالی به دست آمده است. در P رئوس ۹ دارد بنابراین جستجوی محلی به منظور قرار دادن این رئوس در یکی از کلاس‌های رنگی برای کمینه کردن مجموع وزن رئوس رنگ‌آمیزی نشده است. در C_0، رأس ۹ با رأس ۳ که وزن آن ۱ است درگیری دارد که کمتر از وزن رأس ۹ است، بنابراین رأس ۳ در $V_{spilled}$ ذخیره شده و مقدار کمینه به عنوان ۱ تنظیم می‌شود. در C_1، رئوس ۱ و ۸ با رأس ۹ درگیری دارند و مجموع ارزش وزن آن‌ها ۵ است که بیشتر از حداقل است بنابراین C_1 برای رأس ۹ مناسب نیست. در نهایت، در C_2 رأس ۴ و رأس ۹ با یکدیگر درگیری دارند و وزن رأس ۴ برابر با ۳ است که دوباره بیشتر از مقدار کمینه است. تمامی کلاس‌های رنگی بازدید شده و رأس ۹ از P حذف شده و به C_0 انتقال داده می‌شود، در حالی که رأس ۳ در $V_{spilled}$ حذف شده و به P انتقال می‌یابد. از آنجا که P خالی نیست، کلاس‌های رنگی برای رأس ۳ بازدید می‌شوند. رأس ۳ در تمام کلاس‌های رنگی درگیری دارد و کمترین وزن را دارد، بنابراین رأس ۳ در V_{tabu} قرار می‌گیرد. P خالی می‌شود، بنابراین W-SWAP کامل می‌شود.

۴.۴ عملکرد تناسب اندام

تابع تلفیق به عنوان مجموع وزن رئوسی که رنگ‌آمیزی نشده‌اند، به عبارت دیگر رئوس حاضر در V_{tabu} پس از اتمام تکنیک جستجوی محلی محاسبه می‌شود. بهترین سناریو این است که لیست V_{tabu} خالی باشد که به این معناست که همه رئوس با k رنگ رنگ‌آمیزی شده‌اند و مقدار تابع تلفیق $f(k)$ برابر با صفر است. در شکل ۵، رأس ۳ برای تمامی کلاس‌های رنگی S_0 تابو می‌شود و مقدار تابع تلفیق S_0 به وزن رأس ۳ که برابر با ۱ است اختصاص داده می‌شود.

شکل ۶ نمایش‌دهنده حل‌های الگوریتم‌ها در پایان عملیات جستجوی محلی است. هر دو الگوریتم HEA و COMA نمی‌توانند دو رأس با وزن کل ۴ را به کلاس‌های رنگی اختصاص دهند، بنابراین مقدار تابع تلفیق این حل‌ها ۴ است. PBEA و InCEA فقط یک رأس بدون رنگ دارند و رأس ۳ کمترین وزن را دارد، بنابراین کار پیشنهادی کمترین مقدار تابع تلفیق را با مقدار ۱ به دست می‌آورد، در حالی که PBEA یک حل با مقدار تابع تلفیق ۲ تولید می‌کند.

اگر مقدار تابع تلفیق حاصله برای فرزند بهتر از یکی یا هر دوی والدین باشد، الگوریتم والدین را با بدترین مقدار تابع تلفیق انتخاب کرده و فرزند را با این والدین در جایگزینی جابجا می‌کند.

۵ مطالعه تجربی

در این بخش، عملکرد کار پیشنهادی با استفاده از گراف‌های تصادفی و گراف‌های مشتق شده از بنچمارک‌های DIMACS با COMA HEA، و کار اولیه ما با نام PBEA مقایسه می‌شود. برای هر گراف، هزینه کل، تعداد راس‌هایی که رنگ نشده‌اند و زمان اجرای الگوریتم‌ها از معیارهای مقایسه اصلی هستند.

آزمایش‌ها بر روی یک کامپیوتر با پردازنده Intel Core i۷ ۴۷۷۰ با فرکانس ۴.۳ گیگاهرتز و ۸ گیگابایت حافظه RAM انجام شد. پیاده‌سازی InCEA در زبان برنامه‌نویسی C^1 است و با استفاده از gcc کامپایل شده است. همچنین HEA و COMA در $C++^2$ پیاده‌سازی شده و با استفاده از g++ کامپایل شده‌اند. استفاده از زبان‌های برنامه‌نویسی مختلف برای پیاده‌سازی الگوریتم‌ها می‌تواند به عنوان یک تهدید برای اعتبارسنجی [۵۰] در نظر گرفته شود به منظور به‌دست آوردن زمان اجرای دقیق الگوریتم‌ها.