

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

محمد خورشیدی روزبهانی

شماره ۴۰۲۱۵۷۴۱۰۰۲۰۳۲

شماره ۴۰۲۱۵۷۴۱۰۰۲۰۱۳

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 رؤوس به  $k$  کلاس رنگی مستقل است که تعداد رؤوس در این کلاس‌ها حداکثر می‌تواند یکی اختلاف داشته باشد، در حالی که هدف مسئله کمینه‌سازی جمع رنگ‌های اختصاص یافته به رؤوس است. همچنین، وزن‌ها می‌توانند به رؤوس 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 ثابت‌ها [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 رؤوس بدون رنگ خواهند ماند. اهمیت رؤوس ممکن است برابر نباشد، بنابراین از یک مقدار وزن برای نشان دادن اهمیت آن‌ها استفاده می‌شود. کار پیشنهادی ما مسئله رنگ‌آمیزی  $k$ -GCP را با استفاده از یک گراف دارای وزن رأسی با هدف کمینه‌سازی جمع وزن کل رؤوس 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 رؤوس را نشان می‌دهد،  $E$  مجموعه یال‌ها را نشان می‌دهد و  $W$  مجموعه مقادیر وزن رؤوس در  $V$  است که تاکیدی بر اهمیت آن‌ها دارند. هدف  $k$ -GCP انجام رنگ‌آمیزی

رئوس در ۷ 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 هیوریستیک 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 پیشینه‌ای 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 پیشینه 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 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 ثابت‌ها 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 ثابت‌ها [۱۹]، هدف این است که حداکثر تعداد متغیرها را به حداقل تعداد ثابت‌ها 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 رؤس 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 رؤس نمایش داده شود. اگر فاصله بین دو سنسور بیشتر از یک حد پیش‌تعیین شده باشد، این سنسورها نمی‌توانند از همان باند استفاده کنند، بنابراین رؤس نماینده آنها توسط یک یال متصل می‌شوند. هدف این مسئله کمینه کردن تعداد باندها (رنگ‌ها) برای تمامی سنسورها است. همچنین وزن رؤس 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 رؤس نمایش داده می‌شوند و تضادهای بین این عناصر با یال‌ها در گراف نمایش داده می‌شوند. اهمیت عناصر با وزن رؤس 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 ایتريشن‌ها 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 رؤس و یال‌ها هستند، و  $w$  مجموعه‌ی مقادیر وزن رؤس 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  $C = \{C_1, C_2, C_3, \dots, C_k\}$  است، به طوری

$$1 \leq k \leq n$$

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.

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

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).

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

## 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  $S_i = \{C_1, C_2, C_3, \dots, C_k\}$  است و هر کلاس رنگ  $C_j$  شامل یک گروه از رؤوس 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 رؤوس را به سه روش مختلف مرتب می‌کند. معیار درجه لکه از وزن رؤوس،  $w(v_i)$  و درجه رؤوس که با  $d(v_i)$  نشان داده می‌شود، استفاده می‌کند. برای تنوع در جمعیت، درجات لکه برای ۴۰٪، ۴۰٪ و ۲۰٪ از کل رؤوس با استفاده از (۳، ۴ و ۵)، به ترتیب، تنظیم می‌شود.

$$S_1(v_i) = w(v_i) \times d(v_i) \quad (۳)$$

$$S_2(v_i) = w(v_i) \times d(v_i)^2 \quad (۴)$$

$$S_3(v_i) = w(v_i) \quad (۵)$$

رؤوس 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 مجموعه رؤوس 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 رؤوسی 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 تمامی رؤوس به یک کلاس رنگ نگاشته شوند.

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 رؤوس بدون تضاد را در هر کلاس رنگ افزایش دهد، به گونه‌ای که تعداد کل رؤوس بدون رنگ بیشینه شود.

دو کلاس رنگ از والدین انتخاب شده و ترکیب می‌شوند تا احتمال یافتن یک گروه بزرگتر و بهتر از رؤوس بدون تضاد افزایش یابد. اپراتور ترکیب یکپارچه ما یک استخر را برای نگه داشتن رؤوس تضادی که نمی‌توانند به کلاس رنگی فعلی تعلق گیرند، پیشنهاد می‌دهد. فرزندان به طور معمول بیش از یک کلاس رنگ خواهند داشت، بنابراین رؤوسی 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 رؤوس باقی‌مانده در استخر دارد که با قرار دادن رؤوس 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  $S_1 = \{C_0^1, C_1^1, C_2^1, \dots, C_{k-1}^1\}$  و  $S_2 = \{C_0^2, C_1^2, C_2^2, \dots, C_{k-1}^2\}$  نشان داده می‌شوند در الگوریتم ترکیب (مراجعه به الگوریتم ۲). اپراتور یک فرزند  $S_0 = \{C_0, C_1, C_2, \dots, C_{k-1}\}$  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  $C_i$  نمایش داده می‌شود، با استفاده از دو تقسیم تصادفی  $C_x^1$  و  $C_y^2$  از هر دو والد ایجاد می‌شود. رؤوس در این تقسیم‌بندی‌ها گروه‌بندی شده و کلاس رنگی از فرزند با رؤوس بدون تضاد ایجاد می‌شود. اگر رؤوس تضادی وجود داشته باشند که نمی‌توانند به کلاس رنگ فعلی اختصاص یابند، الگوریتم سعی می‌کند این رؤوس 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 رؤوسی 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 رؤوسی که به کلاس‌های رنگی تخصیص نیافته‌اند، پیشنهاد دادیم. حتی اگر برخی از رؤوس در استخر بتوانند به کلاس‌های رنگی از فرزندان 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 رؤوس 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 رؤوسی است که با کلاس‌های رنگی از پیش تولید شده تضاد ندارند. به عبارت دیگر، این عملیات تعداد رؤوس بدون تضاد را در هر کلاس رنگ افزایش می‌دهد و تعداد رؤوس 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. رؤوس ۲، ۷، ۸ در  $C_1^1$  و رؤوس ۵، ۹ در  $C_2^2$  ترکیب شده و به کلاس رنگ اول  $C_0$  از فرزند  $S_0$  اضافه می‌شوند. این رؤوس به عنوان تخصیص داده شده علامت‌گذاری شده و از کلاس‌های رنگی والدین حذف می‌شوند که در مراحل بعدی رؤوس نامرئی برای مراحل بعدی شوند. تضادهای بین رؤوس در  $C_0$  بر اساس گراف در شکل ۱ محاسبه می‌شوند. بیشترین رؤوس دارای تضاد با ۳ تضاد، رؤوس ۸ است بنابراین به طور مستقیم در استخر  $P$  انداخته می‌شود. رؤوس ۸ سه یال با رؤوس ۱، ۲ و ۹ دارد. پس از حذف رؤوس ۸ از  $C_0$ ، تضادهای رؤوس ۱، ۲ و ۹ به ترتیب یک واحد کاهش می‌یابد. رؤوس ۲، ۷ و ۱ بیشترین تعداد تضادها را دارند، بنابراین ارزش‌های وزن آن‌ها (در شکل ۱) مقایسه می‌شوند. از آنجا که رؤوس ۷ ارزش وزن کمتری با ارزش وزن ۱ دارد، به عنوان رؤوس با بیشترین تضاد انتخاب می‌شود و به استخر  $P$  انداخته می‌شود و از  $C_0$  حذف می‌شود. رؤوس 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  $C_1^2$  و  $C_1^1$  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 عملکرد 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 عملکرد InCX پیدا می‌شوند. این درصدها نشان می‌دهد که عملکرد ترکیب جدید ما احتمال یافتن یک راه‌حل بدون تضاد را افزایش می‌دهد.

### ۳.۴ تکنیک جستجوی محلی

هدف از عملکرد 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 عملکرد 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 عملکرد 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 عملکرد 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 عملکرد 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 (W-SWAP) Weighted-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 رؤس با  $v_p$  در همه  $k$  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. رؤس 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 رؤس رنگ‌آمیزی نشده برای  $S_0$  را نگه می‌دارد. جستجوی محلی ادامه می‌یابد تا هر  $v_p$  در  $P$  پردازش شود و  $P$  خالی شود. همه  $v_p$  رؤس 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، رؤس 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  $P$  رؤس ۹ دارد بنابراین جستجوی محلی به منظور قرار دادن این رؤس در یکی از کلاس‌های رنگی برای کمینه کردن مجموع وزن رؤس 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  $C_1$ ، رؤس 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 با 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 رؤسی که رنگ‌آمیزی نشده‌اند، به عبارت دیگر رؤس 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 رؤس 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 HEA، COMA 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 iV 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.

هر دو HEA و COMA یک فرزند را در پایان مرحله تلاقی ایجاد می‌کنند و سعی می‌کنند این فرزند را در مرحله جستجوی محلی بهبود دهند. بنابراین همه الگوریتم‌ها یک فرد در یک تکرار ایجاد می‌کنند. از آنجا که تمام الگوریتم‌ها با استفاده از همان اندازه جمعیت و همان تعداد تکرارها ارزیابی می‌شوند، تعداد ارزیابی‌های تابع تناسب [۵۰] توسط الگوریتم‌ها برابر است، به جز آزمایشی که با استفاده از یک محدودیت زمان ثابت انجام شده است. برای این آزمایش، تعداد تکرارهای استفاده شده توسط تمام الگوریتم‌ها را نیز ارائه داده‌ایم. برای آزمایش‌های زیر، تمام الگوریتم‌ها

پارامترهای یکسانی را دریافت می‌کنند و هیچ‌کدام از آن‌ها از پارامترهای کنترل یا تنظیم‌کننده‌هایی مانند REVAC، F-Race یا CRS-Tuning استفاده نمی‌کنند.

معیارهای عملکرد عملگرها در الگوریتم ما را برای ارزیابی، کار ارائه شده را با مطالعه اولیه ما PBEA [۴۷] با استفاده از سه نمودار که  $n = 500$  و  $\alpha = \{0.250.600.90\}$  مقایسه کرد. عملکرد الگوریتم‌ها با و بدون استفاده از تکنیک‌های جستجوی محلی آن‌ها اندازه‌گیری شد. بنابراین، PBEA و InCEA که فقط با عملگرهای تراکم کار می‌کنند به ترتیب به عنوان PBC و InCX مشخص می‌شوند. عملکرد الگوریتم‌ها ۲۰ بار در هر گراف با استفاده از ۲۰ کلاس رنگ ارزیابی شد. مقایسه عملکرد الگوریتم‌ها از نظر ارزش‌های تناسب و تعداد رأس‌های بدون رنگ در شکل ۷ نشان داده شده است. زمانی که ارزش‌های تناسب آن‌ها را مدنظر قرار دهیم، عملگر تراکم ارائه شده InCX به کمک عملیات جستجوی معکوس، عملگر PBC را پیش‌تیبانی می‌کند اما PBEA با تکنیک جستجوی محلی خود InCX را پیش‌تیبانی می‌کند. انتظار می‌رود که از آنجایی که InCX وزن رؤس را در نظر نمی‌گیرد اما به دنبال به دست آوردن بزرگترین گروه رأس غیر تعارضی است، این عملکرد را انجام دهد. وزن رؤس رنگارنگ را در تکنیک جستجوی محلی پیشنهادی ما (W-SWAP) می‌تواند افزایش دهد. از طرف دیگر، با هدف کاهش مقدار ارزش تناسب، رویکرد ما بهترین مقدار ارزش تناسب را در تمام موارد آزمایشی ارائه شده دارد.

زمانی که تعداد رؤس بدون رنگ مدنظر قرار گرفته است، این تعداد در InCEA نسبت به InCX بیشتر است، زیرا هدف تکنیک جستجوی محلی جدید ما از کاهش مقدار ارزش تناسب است. بنابراین، ممکن است با جایگزینی یک رأس بدون رنگ با وزن بالاتر با بیش از یک رأس دارای مجموع وزن کمتر، تعداد رؤس بدون رنگ افزایش یابد. بنابراین این ممکن است تعداد رؤس بدون رنگ را افزایش دهد تا مقدار ارزش تناسب را کاهش دهد. InCX با موفقیت بزرگترین گروه رؤس غیر تعارضی را به دست می‌آورد و تعداد کمتری از رؤس بدون رنگ را باقی می‌گذارد. از طرف دیگر، PBC و PBEA بیشترین تعداد رؤس را می‌پاشند زیرا هنگامی که کلاس‌های رنگ از طریق عملگرهای تراکم ساخته می‌شوند، اعضا آن‌ها ثابت هستند که باعث کاهش فضای جستجوی راه‌حل می‌شود.

## ۱.۵ نمودارهای تولید شده به صورت تصادفی

برای تولید گراف‌های تصادفی یک مولد گراف تصادفی پیاده‌سازی شده است که از سه پارامتر ورودی استفاده می‌کند. این پارامترها شامل تعداد رؤس  $n$ ، محدوده‌ای از ارزش‌های وزن  $\gamma$  برای اختصاص ارزش‌های وزن به رؤس، و چگالی لبه  $\alpha$  برای تولید لبه‌های اتصال بین رؤس است. تعداد کل لبه‌ها در گراف‌های تولید شده به نزدیکی به  $2 \times \alpha \times (n \times (n - 1)) \div 2$  است. از آنجا که لبه‌ها به صورت تصادفی ایجاد می‌شوند، برای مقادیر کم چگالی لبه، ممکن است برخی از رؤس جدا شوند، بنابراین هیچ تضمینی برای به دست آوردن گراف‌های متصل وجود ندارد. مولد گراف ما یک لبه بین دو رأس جدا شده را که به صورت تصادفی انتخاب شده‌اند، اضافه می‌کند. پس از ساخت گراف، ارزش‌های وزن رؤس به صورت تصادفی با استفاده از توزیع یکنواخت تنظیم می‌شوند.

الگوریتم‌ها با استفاده از تعداد ثابت و پیش‌فرضی از کلاس‌های رنگ برای گراف‌های تولید شده آزمایش می‌شوند. تعداد کلاس‌های رنگ برابر با  $(n \times \beta)$  است، جایی که  $\beta$  چگالی کلاس‌های رنگ است. محدوده مقادیر برای پارامترهای تولید گراف و چگالی کلاس‌های رنگ در جدول ۱ لیست شده است.

### ۱.۱.۵ تنظیمات پارامترهای الگوریتم‌ها

به عنوان بخشی از مطالعه آزمایشی ما، عملکرد الگوریتم‌ها با استفاده از گراف‌های مختلف برای تنظیم اندازه جمعیت الگوریتم‌ها و تعداد تکرارهای انجام شده توسط الگوریتم‌ها اندازه‌گیری شده است. نتایج نشان می‌دهند که عملکرد الگوریتم‌ها به اندازه اولیه جمعیت بستگی ندارد، بنابراین برای تمام الگوریتم‌ها به ۱۰۰ تنظیم شده است.

تأثیر تعداد تکرار برای COMA HEA و InCEA بر روی یک گراف با  $n = 500$  و  $\alpha = 0.90$  مشاهده شده است. گراف تولید شده تقریباً کاملاً متصل است، بنابراین الگوریتم‌ها برای به دست آوردن بهترین نتایج خود نیاز به بیشترین تعداد تکرار دارند. عملکرد الگوریتم ما پس از تکرار ۱۰۰۰ ایستا می‌ماند، اما COMA و HEA پس از تکرار ۲۰۰ به بهترین نتایج خود می‌رسند. بنابراین، ما تعداد تکرار را برای همه الگوریتم‌ها به ۱۰۰۰ تنظیم می‌کنیم. در بقیه آزمایش‌ها، اندازه جمعیت و تعداد تکرارها به ترتیب به ۱۰۰ و ۱۰۰۰ تنظیم شده است مگر اینکه خلاف آن ذکر شده باشد.

عملکرد الگوریتم‌ها برای جفت  $(n\alpha)$  با ارزش‌های متغیر  $\beta$  در جدول ۲ و جدول ۳ نشان داده شده است. هر ردیف در جدول ۲ نتایج میانگین به دست آمده از ۱۲۵ مورد آزمایش را نشان می‌دهد. با افزایش مقدار  $\alpha$ ، تعداد بیشتری از یال‌ها بین رأس‌ها به گراف اضافه می‌شود و این باعث افزایش هم تعداد رأس‌های بدون رنگ و هم مقدار تناسب الگوریتم‌ها می‌شود. این همچنین باعث افزایش زمان اجرا می‌شود، زیرا فضای جستجوی الگوریتم‌ها افزایش می‌یابد. این روند را می‌توان با استفاده از نتایج به دست آمده از عملکرد الگوریتم‌ها به راحتی مشاهده کرد. نتایج



نشان می‌دهد که الگوریتم پیشنهادی نسبت به دو الگوریتم از ادبیات از نظر مقدار تناسب، تعداد راس‌های بدون رنگ و زمان اجرای کل بهتر عمل می‌کند.

## ۲.۱.۵ ارزیابی عملکرد الگوریتم‌ها

عملکرد COMA HEA و InCEA بر روی گراف‌های تصادفی با ویژگی‌های مختلف ارزیابی می‌شود. برای هر ترکیب از  $n$  و  $\alpha$  که در جدول ۱ آمده است، پنج گراف تصادفی ایجاد می‌شود. بنابراین، کلیه ۱۵۰ گراف مختلف برای اندازه‌گیری عملکرد الگوریتم‌ها با پنج مقدار مختلف  $\beta$  استفاده می‌شود. برای ۷۵۰ مورد آزمایش، الگوریتم‌ها در این بخش پنج بار اجرا می‌شوند.

جدول ۳ مقایسه‌ای جفتی انجام می‌دهد با استفاده از دو ستون، و هر ستون نشان‌دهنده تعداد کل موردهای آزمایش (از ۱۲۵ مورد) است که الگوریتم اول میزان بهتر، برابر یا بدتری نسبت به الگوریتم دوم را دارد. وقتی  $\alpha$  برابر با ۱۰.۰ باشد، الگوریتم‌ها حداقل در ۶۰٪ از موارد آزمایشی نتایج یکسانی را دریافت می‌کنند زیرا گراف‌ها تقریباً خالی از یال هستند. الگوریتم پیشنهادی در بیشتر موارد آزمایشی نتایج بدتری نسبت به هر دو COMA و HEA دارد هنگامی که مقدار  $\alpha$  افزایش می‌یابد. در ستون سوم، تعداد موارد آزمایشی که همه راس‌ها می‌توانند رنگ‌آمیزی شوند آمده است. تعداد رنگ‌هایی که الگوریتم‌ها استفاده می‌کنند به نسبت  $n$  است؛ بنابراین برای ارزیابی‌های با ارزش  $n$  بزرگتر، تعداد کل موارد آزمایشی که الگوریتم‌ها می‌توانند تمام راس‌ها را رنگ‌آمیزی کنند افزایش می‌یابد. وقتی  $\alpha = \{0.75, 0.90\}$ ، گراف تقریباً کاملاً متصل است؛ بنابراین در هیچ یک از موارد آزمایشی، الگوریتم‌ها قادر به رنگ‌آمیزی تمام راس‌های گراف نیستند. الگوریتم پیشنهادی به ترتیب در ۷۵٪ و ۷۶٪ از کل موارد آزمایشی COMA و HEA را پیش می‌گیرد. الگوریتم پیشنهادی در ۳۶٪ از موارد آزمایشی توانسته است تمام راس‌ها را رنگ‌آمیزی کند، در حالی که نتایج برای COMA و HEA هر دو ۱۷٪ است.

کارایی الگوریتم‌ها نیز برای جفت  $(n, \beta)$  با ارزش‌های متغیر  $\alpha$  ارزیابی می‌شود. مقادیر پایین‌تر  $\beta$  نشان‌دهنده استفاده کمتر از تعداد رنگ‌ها توسط الگوریتم‌ها است، بنابراین هم ارزش فیتنس و هم تعداد راس‌های بدون رنگ برای مقادیر پایین  $\beta$  بالا است، همانطور که در جدول ۴ قابل مشاهده است. عملکرد الگوریتم‌ها برای هر جفت  $(n, \beta)$  اندازه‌گیری شده و در هر ردیف، نتایج میانگین (از بین ۱۵۰ مورد آزمایشی) برای فیتنس و تعداد راس‌های بدون رنگ داده شده است. کار پیشنهادی ما در هر ردیف بهترین نتایج را به دست می‌دهد.

در جدول ۵، مقایسه دو به دو الگوریتم‌ها آورده شده است. برای مقادیر  $(\beta)$  بزرگتر، تعداد رنگ‌های استفاده شده توسط الگوریتم‌ها افزایش می‌یابد و این منجر به اختصاص بیشتر راس‌ها به رنگ‌های داده شده می‌شود. الگوریتم ما بیشترین تعداد موردهای آزمایشی را دارد که تمام راس‌ها با موفقیت رنگ‌آمیزی شده‌اند.

در کار پیشنهادی ما، راس‌های غیر رنگ‌آمیزی شده که کمترین مقدار تابع فیتنس را حاصل می‌کنند، پس از اتمام تکنیک جستجوی محلی در  $V_{tabu}$  ذخیره می‌شوند. بنابراین، الگوریتم ما نیازی به محاسبات اضافی برای یافتن بهترین راس (ها) برای بدون رنگ کردن کمترین مقدار فیتنس ندارد. اما در COMA و HEA یک جستجوی کامل در تکنیک جستجوی محلی و در انتخاب راس‌های بدون رنگ برای محاسبه فیتنس وجود دارد، به همین دلیل زمان اجرای الگوریتم پیشنهادی به طور قابل توجهی بهتر از COMA و HEA است که به وضوح از جداول ۲ و ۴ مشهود است.

## ۲.۵ معیارهای DIMACS

با توجه به توضیحات، این ۷۳ نمونه از نمودارهای DIMACS دارای پنج نوع مختلف هستند که برای مسأله رنگ‌آمیزی گراف با وزن راس استفاده می‌شوند. نمودارهایی که برای مسأله تخصیص ثبت نام به منابع ایجاد شده‌اند با نام "Rn\_ag{b}\*.col" نشان داده می‌شوند. نمودارهای GEOM نمودارهای هندسی هستند و با "GEOM{a,b}n.col" نشان داده می‌شوند که  $a$  و  $b$  به درصد چگالی متناظر می‌انجامد. نمودارهای queen با استفاده از مسأله صفحه شطرنج  $n \times n$  ایجاد می‌شوند، که با "queenn\_n.col" نمایش داده می‌شوند. نمودارهای DSJC نمودارهای تصادفی برای مسأله انجمن‌سازی تقلیدی هستند و با "DSJCn\_ag{b}\*.col" نشان داده می‌شوند. نمودارهای myciel بر اساس تبدیل Mycielski ایجاد می‌شوند و با "myciel{5,6,7}g{b}\*.col" نشان داده می‌شوند. در نام‌های نمودارهای R، queen، myciel و DSJC، محدوده وزن راس‌ها با "g" ( $\gamma = [1..5]$ ) یا "gb" ( $\gamma = [1..20]$ ) نمایش داده می‌شود.

برای آزمایش‌های مربوط به مجموعه داده‌های DIMACS، ابتدا کمترین مقدار  $k$  برای هر الگوریتم برای به دست آوردن یک رنگ‌آمیزی  $k$ -معتبر یافت می‌شود. آزمایش‌ها با مقدار کافی بزرگی از  $k$  شروع می‌شود و  $k$  کاهش می‌یابد تا یک راه‌حل غیرقانونی پیدا شود که برخی از راس (ها) از گراف‌ها رنگ‌آمیزی نشده‌اند. کمترین  $k$  که توسط الگوریتم‌ها یافت می‌شود به عنوان حداقل تعداد رنگ استفاده شده توسط الگوریتم‌ها مشخص می‌شود. یک‌بار که بهترین مقدار  $k$  ( $k_{best}$ ) برای هر نمونه یافت شود، عملکرد الگوریتم‌ها با استفاده از سناریوهای مختلف اندازه‌گیری می‌شود. جدول ۶ و جدول ۷ مقایسه عملکرد الگوریتم‌ها روی ۷۳ مثال از مجموعه داده‌های DIMACS را ارائه می‌دهند، جایی که هر الگوریتم پنج

بار اجرا شده است تا بهترین نتیجه با استفاده از ۳۰۰۰۰ تکرار به دست آید. نام گراف، تعداد رأس و تعداد یال‌ها در گراف‌ها در سه ستون اول جداول ذکر شده است. برای هر الگوریتم، تعداد کمینه رنگ‌های استفاده شده و زمان اجرا در دو ستون جداگانه لیست شده است. از سوی دیگر، ستون مشخص شده توسط  $\Delta_1$  در جداول اختلاف بین تعداد رنگ‌های استفاده شده توسط الگوریتم InCEA و HEA را نشان می‌دهد، یعنی  $k_{InCEA} - k_{HEA}$ . به طور مشابه، ستون  $\Delta_2$  اختلاف  $k_{InCEA} - k_{COMA}$  را برای گراف‌های مورد بررسی در آزمایش‌های ما نشان می‌دهد.

InCEA در ۳۰٪ و ۳۳٪ از نمونه‌ها، نسبت به HEA و COMA عملکرد بهتری داشته است، در حالی که در سه نمونه، به نام‌های queen10\_10g.col، queen10\_10gb.col و GEOM120a.col؛ InCEA از یک رنگ اضافی استفاده کرده است. برای این سه نمونه، ما علاوه بر بررسی موردی برای رفتار الگوریتم‌ها، سه نمونه‌ی دیگر R100\_1gb.col، R100\_5gb.col و R100\_9gb.col را اضافه کرده‌ایم که InCEA نتایج بهتری به دست آورده است. برای این شش نمونه، الگوریتم‌ها ۲۰ بار با استفاده از  $k_{best}$ ، مقادیر تناسب و تعداد رأس‌های غیر رنگ‌آمیز را نشان می‌دهند در شکل ۸ و شکل ۹ به ترتیب. به عنوان مثال، تعداد کمینه رنگ‌های استفاده شده توسط HEA و COMA برای رنگ‌آمیزی R100\_1gb.col برابر ۶ است، در حالی که InCEA با استفاده از ۵ رنگ موفق به رنگ‌آمیزی این نمونه شده است. بنابراین،  $k_{best}$  برای به دست آوردن یک رنگ‌آمیزی قانونی  $k$  برای این نمونه ۵ است، و تمامی الگوریتم‌ها با استفاده از ۵ تعداد کلاس رنگی اجرا می‌شوند. ارزش تناسب برای یک راه‌حل مجاز صفر است، در صورتی که یک راه‌حل نامجاز یافت شود، سپس ارزش تناسب الگوریتم به عنوان مجموع وزن رأس(ها) رنگ‌آمیزی نشده محاسبه می‌شود.

هنگامی که ارزش‌های تناسب الگوریتم‌ها در نظر گرفته می‌شود، نتایج کار ما برای سه نمونه اول، R100\_1gb.col، R100\_5gb.col و R100\_9gb.col، متمرکزتر است. در InCEA، GEOM120a.col، R100\_9gb.col، R100\_1gb.col و R100\_5gb.col، اما این یک نقطه پرت برای الگوریتم ما است زیرا در ۸۰٪ از اجراها الگوریتم ما مقدار ۱ را به دست می‌آورد، در حالی که HEA و COMA به ترتیب در ۷۰٪ و ۶۵٪ از اجراها بهترین ارزش تناسب ۰ را بدست می‌آورند. برای هر دو queen10\_10g.col و queen10\_10gb.col، InCEA می‌تواند بهترین ارزش تناسب را در ۱۵٪ و ۱۰٪ از اجراها بدست آورد و این به عنوان یک نقطه پرت در نظر گرفته می‌شود زیرا الگوریتم ما نسبت به HEA و COMA کمتر واریانس دارد و در اکثر موارد ارزش تناسب ۱ را تولید می‌کند، بنابراین این موارد پیش‌بینی‌پذیرتر هستند. در شکل ۹ برای سه نمونه اول، InCEA تعداد کمینه رأس‌های رنگ‌آمیزی نشده را دارد. حتی حداقل مقادیر مشاهده شده توسط HEA و COMA بزرگتر از نقطه پرت یا مقدار Q۳ InCEA هستند. اما برای GEOM120a.col، queen10\_10g.col و queen10\_10gb.col، رفتار الگوریتم‌ها بسیار شبیه به ارزش‌های تناسب است.

برای نمایش عملکرد الگوریتم‌ها برای تعداد متغیری از کلاس‌های رنگ، الگوریتم‌ها به اندازه ۱۰ بار برای مقادیر کلاس رنگ کمتر یا مساوی با  $k_{best}$  اجرا می‌شوند. مقادیر میانگین ارزش‌های تناسب برای همان شش نمونه در شکل ۱۰ آورده شده است. HEA و COMA الگویی تقریباً مشابه دارند با اختلافات کوچک، در حالی که InCEA با استفاده از تعداد کمتری از رنگ‌ها برای رنگ‌آمیزی نمونه‌های اولیه به حداقلین ارزش تناسب می‌رسد. InCEA برای رنگ‌آمیزی GEOM120a.col عملکرد بدتری دارد و بیشترین تعداد رنگ را استفاده می‌کند. تمام الگوریتم‌ها برای رنگ‌آمیزی queen10\_10g.col و queen10\_10gb.col عملکرد مشابهی دارند.

همچنین عملکرد الگوریتم‌ها برای یک محدودیت زمانی ثابت اندازه‌گیری می‌شود که به ۵۰۰ ثانیه تنظیم شده است. در این آزمایش‌ها نیز شش نمونه و مقادیر kbest آنها استفاده می‌شوند. شکل ۱۱ مقادیر ارزش تناسب میانگین به دست آمده با استفاده از پنج اجرا را نشان می‌دهد. برای نمونه‌های اولیه R100\_1gb.col، R100\_5gb.col و R100\_9gb.col، الگوریتم ما به‌طور موفقیت‌آمیز رنگ‌آمیزی-k مجاز را بدست می‌آورد و هر دو HEA و COMA را شکست می‌دهد. در آزمایش‌های قبلی، تعداد تکرارها پارامتر اصلی است که به ۳۰۰۰۰ تنظیم شده و الگوریتم ما در بیشتر موارد موفق به رنگ‌آمیزی GEOM120a.col، queen10\_10g.col و queen10\_10gb.col نمی‌شود. این نهال با استفاده از تعداد بیشتری از تکرارها به‌طور موفقیت‌آمیز این سه نمونه را رنگ‌آمیزی می‌کند. هنگامی که مقادیر ارزش تناسب میانگین در نظر گرفته می‌شوند، COMA به سریع‌ترین زمان ممکن برای GEOM120a.col بهترین نتیجه را بدست می‌آورد. برای همان نمونه، InCEA در اکثر موارد یک رنگ‌آمیزی مجاز-k را پیدا می‌کند. InCEA عملکرد بهتری را در queen10\_10g.col و queen10\_10gb.col از نظر ارزش تناسب میانگین به‌دست می‌آورد.

HEA و COMA در تمام مراحل الگوریتم‌های خود تصمیمات خود را بر اساس معیارهای هوشمندی می‌گیرند، به طوری که به یک راه‌حل می‌رسند که دیگر نمی‌تواند بهبود یابد. الگوریتم ما همچنین مکانیسم‌های هوشمندی مانند عملیات جستجو به عقب یا عملیات مبادله وزن‌دار را شامل می‌شود، اما در مرحله ترکیب تصادفی کلاس‌های رنگ انتخاب می‌کند. بنابراین، اگر اجازه داده شود تعداد زیادی تکرار استفاده شود، به کاوش فضای جستجو ادامه می‌دهد.

در تنظیمات آزمایشات یکسان، توزیع میانگین زمان محاسباتی (به صورت درصد) الگوریتم‌ها و میانگین تعداد تکرارهایی که استفاده می‌کنند، اندازه‌گیری شده است. همانطور که به وضوح از جدول ۸ مشاهده می‌شود، هر دو HEA و COMA بیشترین بخش زمان خود را در مرحله جستجوی محلی صرف می‌کنند، در حالی که به دلیل استفاده از استخر و لیست تابو، InCEA کمتر از ۵٪ و ۱٪ از زمان محاسباتی خود را در مرحله جستجوی محلی و محاسبه فیتنس، به ترتیب صرف می‌کند. این آزمایش تفاوت‌های الگوریتمی الگوریتم‌ها و دلایل پشت تعداد مختلف تکرارها برای یک محدودیت زمانی ثابت را آشکار می‌کند.

در این مقاله، یک الگوریتم تکاملی نوین برای مسأله رنگ آمیزی وزنی- $k$  رأس‌ها پیشنهاد می‌شود. این الگوریتم دو هدف دارد: (۱) بیشینه کردن تعداد رأس‌های غیرتداخلی که می‌توانند به همان رنگ اختصاص داده شوند با استفاده از اپراتور ترکیب یکپارچه، (۲) کمینه کردن وزن کل رؤس بدون رنگ کردن با استفاده از تکنیک جستجوی محلی به نام وزن‌دار-تعویض. اپراتور ترکیب یکپارچه (IncX) از یک استخر و یک عملیات بازگشت جستجو استفاده می‌کند. استخر رؤسی را که حداقل یک تداخل با رؤس موجود در کلاس‌های رنگ فرزند دارند، نگه می‌دارد، بنابراین اجزای ممکن (رؤس اختصاص یافته به کلاس‌های رنگ فرزند) و غیرممکن (رؤس موجود در استخر) از راه‌حل جدا می‌شوند. عملیات بازگشت جستجو سعی می‌کند تعداد رؤس موجود در استخر را با اختصاص آن‌ها به یک کلاس رنگ در فرزندان کاهش دهد، اگر امکان پذیر باشد. تکنیک جستجوی محلی به کاهش مجموع وزن‌های رؤس بدون رنگ کردن با در نظر گرفتن راه‌حل‌های همسایه هدف دارد. هنگامی که تکنیک جستجوی محلی پایان می‌یابد،  $V_{tabu}$  شامل رؤس بدون رنگ می‌شود، بنابراین بدون نیاز به جستجوی جامع، مقدار سلامتی به سرعت بسیار زیادی محاسبه می‌شود. ما یک ارزیابی تجربی با استفاده از بنچمارک‌های سنتی و گراف‌های DIMACS انجام داده‌ایم. بنچمارک‌های سنتی شامل ۱۵۰ گراف و ۳۷۵۰ حالت آزمایشی هستند، در حالی که DIMACS دارای ۷۳ گراف معروف است. کار پیشنهادی با یک الگوریتم تکاملی ترکیبی و یک الگوریتم ممیتیک که هدف حل همان مسئله است مقایسه شده است. مطالعه تجربی نشان می‌دهد که الگوریتم پیشنهادی نتایج موفقیت‌آمیزی را تولید کرده است و با توجه به تعداد کل رنگ‌های استفاده شده، مقدار سلامتی و زمان اجرا، عملکرد الگوریتم‌های موجود در ادبیات را پیشی گرفته است. مسئله رنگ‌آمیزی گراف نقش مهمی در حل بسیاری از مسائل نظری و عملی مانند مسئله تخصیص منابع و مسئله تخصیص ثبت‌ها ایفا می‌کند. به عنوان یک کار آینده، الگوریتم پیشنهادی می‌تواند برای حل بسیاری از مسائل واقعی با دقت مطلوب در زمانی معقول مورد استفاده قرار گیرد. الگوریتم ما همچنین می‌تواند برای حل مسئله رنگ‌آمیزی گراف پویا مناسب باشد، زیرا به راحتی می‌تواند با تغییرات پویای در گراف سازگار شود. همچنین، اپراتور ترکیب یکپارچه پیشنهادی می‌تواند برای یک طیف گسترده از مسائل گروه‌بندی یا خوشه‌بندی که موارد مشکلات در آن‌ها وجود دارد، استفاده شود.