الگوریتم ژنتیک با جهش آشوبی هوشمند و ترکیب چندنقطه‌ای مکاشفه‌ای برای حل مسئلهٔ رنگ‌آمیزی گراف

سید علی ساداتی تیله‌بنی، حمید جزایری* و مجتبی ولی‌نتاج

گروه مهندسی کامپیوتر، دانشکده برق و کامپیوتر، دانشگاه صنعتی نوشیروانی بابل، بابل، ایران

چکیده

تخصیص مقدار رنگی را به هر یک از گره‌های گراف، به‌گونه‌ای که هیچ دو گرهٔ مجاوری دارای رنگ یکسانی نباشد و کمترین مقدار رنگی استفاده شود، مسئلهٔ رنگ‌آمیزی گراف گویند. این مسئله به‌عنوان یکی از مسائل NP-hard شناخته می‌شود که کاربردهای مختلفی در زمینهٔ تخصیص پهنای باند، اختصاص حافظه به برنامه‌ها و همچنین، طراحی مدارهای مجتمع دارد. در مقالهٔ حاضر، از الگوریتم ژنتیک و پدیدهٔ آشوب برای حل این مسئله استفاده شده است. در روش پیشنهادی حاضر، عمل‌گر ترکیب چندنقطه‌ای مکاشفه‌ای به نام CMHn معرفی شده است. این عمل‌گر، با انتخاب چند نقطهٔ برش در والدین و معتبرکردن یکی از زیر بخش‌های والدین (دومین زیربخش هر والد می‌تواند معتبر یا غیر معتبر باشد) آنها را با هم، با استفاده از روشی ابتکاری ترکیب می‌کند. برای اینکه بتوان از بهینهٔ محلی فرار کرد و همچنین، برای یافتن فضای جستجوی جدید، از عمل‌گر جهش استفاده می‌شود. در این مقاله، عمل‌گر جهش آشوبی هوشمند معرفی شده است که با استفاده از فرمولی گره‌هایی را که برای جهش مناسب‌ترند، انتخاب و بر روی آنها جهش را اعمال می‌کند. همچنین، نیمی از جمعیت اولیه با استفاده از روش ابتکاری و نیمی از آن با روش تصادفی تولید شده است. به‌منظور ارزیابی الگوریتم پیشنهادی از نمونه گراف‌های DIMACS و Queen استفاده شده است. نتایج به‌دست‌آمده نشان می‌دهد که روش پیشنهادی در بیش‌تر گراف‌ها، به‌خصوص گراف‌های بسیار بزرگ (wap) و گراف‌های Queen جواب بهتری نسبت به تحقیقات مشابه ارائه می‌دهد.

واژگان کلیدی: مسئلهٔ رنگ‌آمیزی گراف، الگوریتم ژنتیک، روش ابتکاری، ترکیب چندنقطه‌ای مکاشفه‌ای، جهش آشوبی هوشمند

Genetic Algorithm with Intelligence Chaotic Algorithm and Heuristic Multi-Point Crossover for Graph Coloring Problem

Seyyed Ali Sadati Tilehboni, Hamid Jazayeriy* & Mojtaba Valinataj

Department of Computer Engineering, Babol Noshirvani University of Technology, Babol, Iran

Abstract

Graph coloring is a way of coloring the vertices of a graph such that no two adjacent vertices have the same color. Graph coloring problem (GCP) is about finding the smallest number of colors needed to color a given graph. The smallest number of colors needed to color a graph G, is called its chromatic number. GCP is a well-known NP-hard problems and, therefore, heuristic algorithms are usually used to solve it. GCP has many applications such as: bandwidth allocation, register allocation, VLSI design, scheduling, Sudoku, map coloring and so on.

* نویسندهٔ عهده‌دار مکاتبات

* Corresponding author

We try genetic algorithm (GA) and chaos theory to solve GCP. We proposed a heuristic algorithm called CMHn to implement multi-point crossover operation in GA. To generate initial population, a fast greedy algorithm is used. In this algorithm, the degree of each node and the number colors in its neighbor is used to assign a color to each node. Mutation operation in GA is used to explore the search space and scape from the local optima. In this study, a chaotic mutation operation is presented to select some vertices and change their color. The crossover and mutation parameters in the proposed algorithm is tuned based on some experiment.

To evaluate the proposed algorithm, some experiment is conducted on DIMACS data set. Among DIMACS sample graphs, *DSJ, Queen, Le450, Wap* are well-known challenging samples for graph coloring. The proposed algorithm is executed 10 times on each sample and the best, worst and mean results are reported.

Results show that the proposed algorithm can effectively solve GCP and have comparable outcome with the recent studies in this field. The proposed method outperforms other algorithms on very large graphs (*Wap* graphs).

Keywords: **Graph coloring problem, genetic algorithm, heuristics, multi-point crossover, chaotic mutation**.

۱- مقدمه

هدف مسئلهٔ رنگ‌آمیزی گراف، تخصیص مقادیر رنگی به هر یک از گره‌های گراف است. البته، با این شروط که نخست این‌که، همهٔ گره‌ها دارای مقدار رنگی شوند و دوم این‌که، دو گرهٔ همسایه (دو گره‌ای که با یالی مشترک به هم متصل هستند) هم‌رنگ نباشند؛ اما، استفاده از کمترین مقدار رنگی برای رنگ‌آمیزی، هدف دیگری است که این مسئله را به گروه مسائل NP برده است[1]. یعنی، مسائلی که تا به حال، برای حل آنها الگوریتم قطعی با زمان چندجمله‌ای یافت نشده است.

مسئلهٔ رنگ‌آمیزی گراف در زمینه‌های زمان‌بندی [3]،[2]، تخصیص پهنای باند فرکانسی [4] و اختصاص حافظه به ثبات‌ها [5] کاربرد دارد. در طی سالیان گذشته الگوریتم‌های متعددی برای حل این مسئله ارائه شده‌اند. از جمله این روش‌ها می‌توان به روش‌های حریصانه [6]،[7] جستجوی محلی[8]،[9]، ابتکاری [12]-[10] و تکاملی [15]-[13] اشاره کرد که هر کدام از روش‌های ارائه شده دارای مزایا و معایبی هستند. برای مثال، روش‌های دقیق [16] جواب بهینه را به‌دست می‌آورند؛ اما، برای گراف‌های بزرگ به مدت زمان زیادی برای به‌دست آوردن جواب نیاز دارند. در این مقاله، الگوریتم ژنتیک آشوبی برای حل مسئلهٔ رنگ‌آمیزی گراف استفاده شده است.

کارایی الگوریتم ژنتیک وابسته به نحوهٔ پیاده‌سازی عمل‌گرهای ترکیب، جهش و تطبیق آنها با مسئلهٔ مورد نظر است. در روش پیشنهادی حاضر، عمل‌گر ترکیب چندنقطه‌ای مکاشفه‌ای جدید و عمل‌گر جهش آشوبی هوشمند برای رسیدن به جواب مناسب و قابل قبول ارائه شده‌اند. همچنین، جمعیت اولیهٔ مورد نیاز الگوریتم ژنتیک با استفاده از روش تصادفی و روشی ابتکاری تولید می‌شود؛ سپس، مقدار برازندگی هر کروموزوم با استفاده از تابع برازندگی محاسبه و براساس مقدار برازندگی هر کروموزوم، عمل‌گرهای ترکیب و جهش پیشنهادی اعمال می‌شوند. برای به‌روزرسانی جمعیت، جمعیت به سه دستهٔ نخبه، متوسط و بد تقسیم می‌شود و براساس درصدهای از قبل مشخص‌شده کروموزوم‌ها، از هر دسته به جمعیت جدید انتقال پیدا می‌کنند. آزمایش‌های انجام‌شده بر روی برخی از گراف‌های DIMACS و Queen نشان می‌دهد که روش پیشنهادی حاضر، می‌تواند جواب‌های قابل قبولی را در زمان مناسب ارائه دهد.

در بخش دوم مقالهٔ حاضر، پژوهش‌های انجام‌شده در مسئلهٔ رنگ‌آمیزی گراف بیان و در بخش سوم، روش پیشنهادی ارائه می‌شود. در بخش چهارم، روش پیشنهادی، پیاده‌سازی و نتایج آن بر روی برخی از گراف‌های DIMACS و Queen ارائه و ارزیابی و درنهایت، در بخش پنجم خلاصه، نتیجه‌گیری و کارهای آینده بیان می‌شود.

۲- پژوهش‌های مرتبط

الگوریتم‌های تکاملی، سالیان زیادی است که به‌عنوان روشی محبوب برای حل مسئلهٔ رنگ‌آمیزی گراف شناخته شده‌اند. نخستین نوع از این الگوریتم‌ها در اوایل سال ۱۹۹۰ براساس اصول ژنتیک و تکامل، توسعه داده شده است [17]. در سالیان بعد، پیشرفت‌های مهمی به‌وجود آمد و تعدادی از روش‌های رنگ‌آمیزی خاص ایجاد شد. پیشرفت‌های اخیر، به مسائلی از قبیل تنوع جمعیت [18]، بالانس بهره‌برداری

اکتشاف[1] [19]، پویایی جمعیت [20] و مواردی از این دست توجه دارند. در [21] از ترکیب الگوریتم ژنتیک و جستجوی ممنوعه برای حل این مسئله استفاده شده است. در این روش، به‌دلیل استفاده از جستجوی ممنوعه در الگوریتم ژنتیک، تعادل قابل قبولی بین اکتشاف و بهره‌برداری ایجاد شده است. بحث مرکزی مرتبط در الگوریتم‌های ژنتیک حفظ تنوع جمعیت است. در اغلب موارد، هدف تنوع به‌طورکلی کنترل پویای جمعیت، توسط زیرجمعیت تنوع‌گرا [18]، سازوکارهای جدید رد فرزند [20] و تطبیق ترکیب [9] است. در [22]، از معیار فاصله برای نگهداشت یا حذف فرزندان به‌منظور حفظ تنوع جمعیت و از عمل‌گر ترکیب GPX که به ترکیب چندوالدینی گسترش یافته برای تولید فرزندان استفاده شده است؛ اما، تنوع جمعیت الگوریتم ژنتیک، بعد از تولید چندین نسل به‌شدت کاهش می‌یابد که دلیلی بر هم‌گرایی زودرس به بهینهٔ محلی است. درحقیقت الگوریتم ژنتیک، مسائل بهینه‌سازی غیرخطی مشکل را حل می‌کند [23]؛ اما، نیاز به زمان زیادی برای هم‌گرایی دارد و ممکن است که در بهینهٔ محلی گیر بیفتد [24]؛ به همین منظور و برای رسیدن به جواب بهینه، ایجاد تنوع جمعیت و جلوگیری از گیرافتادن در بهینهٔ محلی ضروری است [25]. برای حفظ تنوع جمعیت در الگوریتم ژنتیک، می‌توان از پدیدهٔ آشوب بهره گرفت [26]. آشوب با آمار تصادفی متفاوت است و به‌طور ذاتی قادر است تا به‌صورت کارا فضای جستجو را کاوش نماید و عملکرد روال بهینه‌سازی را بهبود بخشد [27]. آشوب می‌تواند به‌عنوان قاعده‌ای بی‌نظم نشان داده شود که در شرایط قطعی دارای رفتار تصادفی غیر قابل پیش‌بینی است. در آشوب، تغییر کوچک در شرایط اولیه، ممکن است باعث تولید خطای بزرگ در جواب نهایی شود. به عبارت دیگر، پدیدهٔ آشوب به‌شدت به شرایط اولیه‌اش وابسته است [28]. یکی دیگر از مشخصه‌های پدیدهٔ آشوب این است که می‌تواند به‌عنوان روشی کارا برای حفظ تنوع جمعیت مسئلهٔ مورد نظر طراحی شود [29].

برای حفظ تنوع جمعیت الگوریتم ژنتیک، در مقالهٔ [30] از پدیدهٔ آشوب استفاده شده است. استفاده از پدیدهٔ آشوب به الگوریتم، برای قرارگیری در نقطهٔ بهینه کمک می‌کند. در این مقاله، از روش کلاسترینگ k-means برای دسته‌بندی جمعیت استفاده شده است که سعی در کم‌کردن تعداد تکرار اجرای الگوریتم دارد. برای حل مسئلهٔ بهینه‌سازی عملیات مخزن آب [31]، از پدیدهٔ آشوب و الگوریتم ژنتیک استفاده شده است تا از گیرافتادن در بهینهٔ محلی جلوگیری کند و سرعت هم‌گرایی الگوریتم ژنتیک را افزایش دهد. در این مقاله، عمل‌گر جهش تبرید آشوبی به‌جای عمل‌گر جهش استاندارد به‌منظور جلوگیری از گیرافتادن در بهینهٔ محلی استفاده شده است. آزمایش‌هایی که در این مقاله انجام‌شده نشان داده است که سرعت هم‌گرایی و دقت جواب، بهبود یافته است.

[1] exploitation-exploration balance

۳-روش پیشنهادی

روش پیشنهادی در این مقاله، از الگوریتم ژنتیک، پدیدهٔ آشوب و روش ابتکاری برای حل مسئلهٔ رنگ‌آمیزی گراف استفاده می‌کند. برای داشتن کارایی مناسب الگوریتم ژنتیک نیاز است تا بتوان عمل‌گرهای این الگوریتم را با مسئلهٔ مورد نظر به‌خوبی تطبیق داد؛ بنابراین در این مقاله، روش ترکیب چندنقطه‌ای مکاشفه‌ای جدید به نام CMHn و جهش آشوبی هوشمند ارائه شده است. نخستین مرحلهٔ الگوریتم ژنتیک مشخص‌کردن نحوهٔ نمایش کروموزوم‌هاست.

۳-۱- نحوهٔ نمایش کروموزوم‌ها

هر گره از گراف نشان‌دهندهٔ یک ژن بوده و مقدار رنگ اختصاص داده‌شده به گره، همان مقدار ژنی خواهد بود. شکل (۱) مثالی را از یک رنگ‌آمیزی ممکن بر روی گراف نشان می‌دهد. در این شکل اعداد درون دایره مقادیر رنگی و اعداد بیرون دایره نام گره است. برای مثال گرهٔ v_1 دارای رنگ شمارهٔ یک و گرهٔ v_2 دارای رنگ شمارهٔ دو و درنهایت گرهٔ v_{10} دارای رنگ شمارهٔ یک است. در این گراف، گره‌های v_4 و v_7 دارای بیشترین مقدار رنگی یعنی رنگ شمارهٔ سه هستند. کروموزوم ch بیان‌گر کروموزوم متناظر رنگ‌آمیزی گراف شکل (۱) است؛ در این صورت، اعداد نوشته‌شده در ch معرف مقادیر رنگی و موقعیت قرارگیری هر یک از اعداد اندیس گره را نشان می‌دهد.

$$ch = [1, 2, 1, 3, 1, 1, 3, 2, 1, 1]$$

۳-۲- تابع ارزیابی

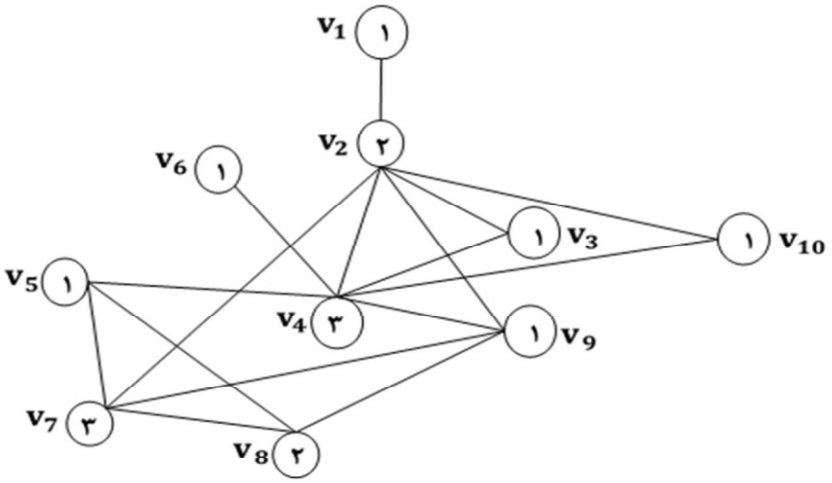
در الگوریتم ژنتیک، ارزش هر یک از کروموزوم‌ها با استفاده از تابع ارزیابی محاسبه می‌شود. در این مقاله، این تابع شامل دو پارامتر است. پارامتر نخست، تعداد تناقضی است که در کروموزوم مربوطه وجود دارد (Conf) و پارامتر دوم، تعداد

طبقهٔ رنگی مجزایی است که برای رنگ‌آمیزی گراف مورد نظر استفاده شده است (MaxColor). ارزش هر یک از کروموزوم‌ها با استفاده از تابع ارزیابی فرمول (۱) محاسبه می‌شود. در این فرمول N تعداد گره‌های گراف بوده و npop تعداد کروموزوم‌های جمعیت است. همچنین، $conf_i$ تعداد تناقض در کروموزوم i اُم و $MaxColor_i$ تعداد طبقهٔ رنگی مجزای استفاده‌شده در کروموزوم i اُم است.

$$g(ch_i) = \frac{T_i}{\sum_{j=1}^{npop} T_j} \quad , \quad T_i = conf_i \times N + MaxColor_i \qquad (۱)$$

هرچه مقدار $g(ch_i)$ کمتر باشد، کروموزوم ch_i شایسته‌تر است. فرض کنید، کروموزوم ch به‌صورت زیر برای شکل (۱) تولید شده باشد:

$ch = [1\ 2\ 2\ 3\ 1\ 3\ 3\ 2\ 1\ 1]$



(شکل-۱): گراف نمونه با ۱۰ گره که با ۳ مقدار رنگی، رنگ‌آمیزی شده است.
(Figure-1): A sample graph colored by 3 colors.

همان‌طورکه دیده می‌شود یال‌های (v_2, v_3) و یال‌های (v_4, v_6) نسبت به هم دارای تناقض هستند. بنابراین، مقدار Conf برای این کروموزوم برابر دو است. این گراف دارای ۱۰ گره است $(n = 10)$. بیشینه تعداد طبقهٔ رنگی مجزایی که برای رنگ‌آمیزی این کروموزوم استفاده شده برابر مقدار سه است $(MaxColor = 3)$. بنابراین، مقدار T برای کروموزوم ch برابر با ۲۳ است $(T_{ch} = 23)$. فرض کنید که جمعیت دارای چهار کروموزوم ch، ch_1، ch_2 و ch_3 باشد که سه کروموزوم ch_1، ch_2 و ch_3 با Tهای محاسبه‌شده برای آنها به‌صورت زیر باشد:

$ch_1 = [2\ 3\ 1\ 2\ 1\ 2\ 1\ 2\ 1\ 3]$; $T_{ch_1} = 43$
$ch_2 = [1\ 2\ 1\ 3\ 1\ 1\ 3\ 2\ 1\ 1]$; $T_{ch_2} = 3$
$ch_3 = [1\ 2\ 1\ 3\ 1\ 2\ 3\ 1\ 1\ 2]$; $T_{ch_3} = 33$

آنگاه مقدار تابع هزینه برای کروموزوم‌های این جمعیت به‌صورت زیر خواهد بود:

$$g(ch_1) = \frac{43}{43+3+33+23} = 0.42$$
$$g(ch_2) = \frac{3}{43+3+33+23} = 0.03$$
$$g(ch_3) = \frac{33}{43+3+33+23} = 0.32$$
$$g(ch) = \frac{23}{43+3+33+23} = 0.23$$

۳-۳- تولید جمعیت اولیه

برای شروع کار الگوریتم ژنتیک باید جمعیت اولیه‌ای وجود داشته باشد. به‌طورمعمول در تولید جمعیت اولیه از روش تصادفی استفاده می‌شود. روش تصادفی در تولید جمعیت اولیه دارای سرعت بالایی است؛ اما جواب‌هایی که ارائه می‌دهد به‌طورعمومی از جواب بهینه به دور هستند؛ بدین‌منظور و برای تولید جمعیت اولیه مناسب از روش ابتکاری می‌توان استفاده کرد. ایجاد جمعیت اولیه با استفاده از روش ابتکاری می‌تواند موجب شود که الگوریتم مورد استفاده زمان خود را صرف بهبود جواب‌های بسیار بد نکند و بتواند در بازه‌ای مناسب از مقادیر رنگی و فضای جستجو، کار خود را شروع کند. البته، واضح است که برای ایجاد تنوع مطلوب باید از روش تصادفی برای تولید بخشی از جمعیت اولیه استفاده کرد. بنابراین در مقاله حاضر، نیمی از جمعیت اولیه با استفاده از روش ابتکاری و نیمهٔ دیگر به صورت تصادفی تولید می‌شود. در روش ابتکاری پیشنهادی برای

تولید جمعیت اولیه، از الگوریتمی حریصانه برای رنگ‌آمیزی گراف استفاده شده است. این الگوریتم جواب‌هایی نزدیک به جواب بهینه تولید می‌کند. در روش‌های حریصانه، ترتیب انتخاب گره‌ها و تخصیص مقدار رنگی مناسب به آنها دارای اهمیت بسیار زیادی است. در این الگوریتم، ترتیب انتخاب گره‌ها، با استفاده از فرمول (۲) تعیین می‌شود:

$$f(v) = \frac{X(v) \times N + Y(v)}{N+1} \qquad (۲)$$

در این فرمول، N تعداد گره‌های گراف، $X(v)$ تعداد همسایه‌های گرهٔ v که درقبل رنگ شده‌اند و $Y(v)$ تعداد همسایه‌های گرهٔ v است؛ در این صورت، گره‌ای که دارای بیشترین مقدار f باشد انتخاب و رنگ‌آمیزی می‌شود که به آن گرهٔ جاری گفته می‌شود. برای انتخاب گرهٔ بعدی، فرمول (۲) برای تمامی همسایه‌های رنگ‌نشدهٔ گرهٔ جاری اعمال و گره‌ای که دارای بیشترین مقدار باشد، برای رنگ‌آمیزی انتخاب می‌شود (گرهٔ جاری جدید). در این الگوریتم، گرهٔ انتخاب‌شده به‌وسیلهٔ متغییر p معرفی شده است. این روال تا زمانی ادامه می‌یابد که در همسایگی گرهٔ جاری، گره‌ای بدون رنگ وجود نداشته باشد. اگر تمامی گره‌های گراف رنگ مناسب را دریافت کرده باشند، کار الگوریتم خاتمه می‌یابد. در غیر این صورت، از بین گره‌هایی که تا به حال مقدار f برای آنها حساب شده ولی رنگی به آنها تخصیص داده نشده است، گره‌ای انتخاب و رنگ مناسبی به آن تخصیص داده می‌شود. این روال تا زمانی‌که همهٔ گره‌ها رنگی را دریافت کنند ادامه می‌یابد. شبه‌کد این روش در شکل (۲) نشان داده شده است. در این شبه کد، مجموعهٔ همسایه‌های یک گره با نماد TN معرفی شده‌اند. به‌عنوان مثال، $TN(p)$، مجموعهٔ گره‌های همسایهٔ گرهٔ p را معرفی می‌کند. برای هر رنگ i، مجموعهٔ تمامی گره‌هایی که دارای رنگ i بوده و در همسایگی گرهٔ p قرار دارند، به‌وسیلهٔ S_i معرفی می‌شوند. خطوط ۱۰-۱۹ شبه‌کد مربوط به انتخاب طبقهٔ رنگی مناسب برای گرهٔ انتخاب‌شده (گرهٔ جاری) است. برای درک بهتر، گراف شکل (۱) با استفاده از این روش رنگ‌آمیزی می‌شود. در مرحلهٔ نخست این الگوریتم، تعداد همسایه‌های هر یک از گره‌ها مشخص و با بردار Y مشخص می‌شود. در ابتدا، با توجه به اینکه رنگ‌آمیزی شروع نشده و هیچ طبقهٔ رنگی‌ای وجود ندارد، مقدار اولیهٔ X برای تمامی گره‌ها برابر با صفر در نظر گرفته می‌شود (در ابتدا هیچ تناقضی وجود ندارد).

$$Y = [1,6,2,6,3,1,4,3,4,2];$$
$$X = [0,0,0,0,0,0,0,0,0,0]$$

با جای‌گذاری این مقادیر در فرمول (۲) مقدار بردار F، که حاوی مقدار تابع f برای تمامی گره‌های گراف است، به‌صورت $F = [0.09,\ 0.55,\ 0.18,\ 0.55,\ 0.27,\ 0.09,\ 0.36,\ 0.27,\ 0.36,\ 0.18]$ می‌شود. بیشترین مقدار f متعلق به گره‌های v_2 و v_4 با مقدار ۰/۵۵ است که به‌صورت پررنگ نیز در آمده است. پس، گرهٔ v_2 انتخاب و مقدار رنگی یک به این گره نسبت داده می‌شود. در گام بعدی، مقدار X و F برای همسایه‌های گرهٔ v_2 به‌صورت زیر محاسبه می‌شود:

$$X = [1,0,1,1,0,0,1,0,1,1];$$
$$F = [\,1,0,1.09,1.45,0,0,1.27,0,1.27,1.09]$$

درنتیجه، بیشترین مقدار f متعلق به گرهٔ v_4 است و مقدار رنگی دو به این گره نسبت داده می‌شود. در ادامه، مراحل اجرای الگوریتم در جدول (۱) نشان داده شده است. با توجه به این جدول، مشاهده می‌شود که در گام هفت مقدار $X = [0,0,0,0,0,0,0,0,0,0]$ شده است. به این معنی‌که، تمامی همسایه‌های گرهٔ v_8 مقدار رنگی مناسب را دریافت کرده‌اند و گره‌ای در همسایگی این گره وجود ندارد که بدون رنگ باقی مانده باشد. در این صورت، گرهٔ بعدی از بین تمامی گره‌های بدون رنگ که مقدار f آنها در مراحل قبل حساب شده است، انتخاب می‌شود. اگر گره‌ای با این شرایط وجود نداشته باشد، گرهٔ بعدی از بین گره‌های بدون رنگ با بیشترین همسایگی انتخاب می‌شود (v_5). در مرحلهٔ شش برای انتخاب مقدار رنگی مناسب برای گره‌ی v_5 دو مقدار رنگی یک و سه قابل تخصیص بودند. در این صورت، اگر گرهٔ v_5 به طبقهٔ رنگی یک وارد شود، به فهرست همسایگی این طبقهٔ رنگی، گرهٔ v_8 اضافه می‌شود؛ اما، با اضافه‌کردن این گره به طبقهٔ رنگی سه دیده می‌شود که گره‌ای به فهرست همسایگی این طبقهٔ رنگی اضافه نمی‌شود؛ پس، مقدار رنگی سه برای این گره مناسب است. درصورتی‌که، مقدار رنگی یک به گره v_5 نسبت داده می‌شد، برای رنگ‌آمیزی این گراف چهار طبقهٔ رنگی مورد نیاز بود. با توجه به اجرای الگوریتم ابتکاری برای شکل (۱)، ترتیب رنگ‌آمیزی به‌صورت زیر است (اعداد درون پرانتز مقدار رنگی نسبت داده شده به هر گره است):

$$v_2(1), v_4(2), v_9(3), v_7(2), \dots, v_{10}(3), v_1(2), v_6(1)$$

```
Function HeuColoring()
1.  C ← zeros(1, N)
2.  Calculate vetor Y
3.  p ← the node with max Y
4.  MaxColor ← 1
5.  C(p) ← MaxColor
6.  while all nodes are not colored do
7.      Calculate f(v) of nodes that are adjacent to p
8.      p ← the node with max f
9.      MinSize ← ∞, SelectedColor ← 0
10.     for i = 1 to MaxColor do
11.         S_i ← ∅
12.         for all v ∈ TN(p) where C(v) = i do
13.             S_i ← S_i ∪ TN(v)
14.         end for
15.         if size(S_i) < MinSize then
16.             MinSize ← Size(S_i)
17.             SelectedColor ← i
18.         end if
19.     end for
20.     if SelectedColor = 0 then
21.         MaxColor ← MaxColor + 1
22.         C(p) = MaxColor
23.     else
24.         C(p) = SelectedColor
25.     end if
26. end while
27. return C
```

(شکل-۲): شبه‌کد الگوریتم انتخاب گره مناسب و رنگ‌آمیزی آن

(Figure-2): Pseudocode for selecting a node to be colored in the next step.

(جدول-۱): ادامهٔ اجرای الگوریتم ابتکاری جهت انتخاب و رنگ‌آمیزی گره‌های گراف

(Table-1): Next steps of graph coloring by selecting a candidate node and assigning a color to it.

Step	Prev. node	Color (C)	X	F
3	v_4	[0 1 0 2 0 0 0 0 0 0]	[0 0 2 0 1 1 0 0 2 2]	[0 0 2 0 1.18 1 0 0 2.18 2]
4	v_9	[0 1 0 2 0 0 0 0 3 0]	[0 0 0 0 0 0 0 2 1 0]	[0 0 0 0 0 0 0 2.18 1.18 0 0]
5	v_7	[0 1 0 2 0 0 2 0 3 0]	[0 0 0 0 2 0 0 2 0 0]	[0 0 0 0 2.09 0 0 2.09 0 0]
6	v_5	[0 1 0 2 3 0 2 0 3 0]	[0 0 0 0 0 0 0 0 3 0]	[0 0 0 0 0 0 0 0 3 0 0]
7	v_8	[0 1 0 2 3 0 2 1 3 0]	[0 0 0 0 0 0 0 0 0 0]	[1 0 2 0 0 1 0 0 0 2]
8	v_3	[0 1 3 2 3 0 2 1 3 0]	[0 0 0 0 0 0 0 0 0 0]	[1 0 0 0 0 1 0 0 0 2]
9	v_{10}	[0 1 3 2 3 0 2 1 3 3]	[0 0 0 0 0 0 0 0 0 0]	[1 0 0 0 0 1 0 0 0 0]
10	v_1	[2 1 3 2 3 0 2 1 3 3]	[0 0 0 0 0 0 0 0 0 0]	[0 0 0 0 0 1 0 0 0 0]
11	v_6	[2 1 3 2 3 1 2 1 3 3]	[0 0 0 0 0 0 0 0 0 0]	[0 0 0 0 0 0 0 0 0 0]

۳-۴- عمل‌گر ترکیب

عمل‌گر ترکیب با ادغام کروموزوم‌ها با یکدیگر سعی دارد، ژن‌های خوب هر کروموزوم را به فرزند منتقل کند تا به این وسیله بتوان جواب کلی الگوریتم برای مسئلهٔ مورد نظر را به جواب بهینه نزدیک کرد. در این مقاله، ترکیبی با نام CMHn جهت تولید دو فرزند معرفی شده است. این ترکیب با استفاده از روش ابتکاری که برای آن تعریف شده است، سعی دارد تا بتواند در اکثر اوقات فرزندان معتبر را تولید کند. همچنین، برای اینکه بتوان تنوع جمعیت را حفظ کرد، این عمل‌گر مبتنی بر ترکیب چندنقطه‌ای تعریف شده است. بنابراین، این عمل‌گر مبتنی بر ترکیب چندنقطه‌ای بوده و با استفاده از روشی ابتکاری به‌طورمعمول منجر به تولید فرزندان معتبر در حل مسئلهٔ رنگ‌آمیزی گراف می‌شود. نحوهٔ کار این عمل‌گر با استفاده از مثالی توضیح داده می‌شود. فرض کنید که برای گراف شکل (۱)، دو والد P_1 و P_2 به‌صورت زیر وجود داشته باشد:

$$P_1 = [1\,2\,1\,3\,1\,1\,3\,1\,1\,1]$$
$$P_2 = [2\,3\,1\,2\,1\,2\,1\,2\,1\,3]$$

که P_1 به‌دلیل هم‌رنگ‌بودن همسایه‌های (v_5, v_8) و (v_8, v_9) و P_2 به‌دلیل هم‌رنگ‌بودن همسایه‌های (v_2, v_{10})، (v_4, v_6)، (v_5, v_7) و (v_7, v_9) دو والد نامعتبر هستند. با فرض اینکه برای والد نخست (P_1)، نقطهٔ برش نخست $C_{11} = 5$ (نخستین نقطه‌ای که دارای تناقض است) و نقطهٔ برش دوم به‌صورت تصادفی و برابر $C_{12} = 8$ انتخاب شده باشند؛ آنگاه، تناقض بین ژن‌های (v_5, v_8) و (v_8, v_9) وجود دارد. در این صورت، یکی از دو سر تناقض از زیرمجموعهٔ نخست والد P_1 حذف می‌شود، تا مقدار ژنی را درصورت امکان از والد دوم دریافت کند. در این صورت، ژن v_8 از زیر مجموعه‌های والد نخست حذف می‌شود. اگر برای والد دوم (P_2)، نقطهٔ برش نخست $C_{21} = 2$ و نقطهٔ برش دوم $C_{22} = 8$ انتخاب شود؛ بخش‌های زیر ایجاد می‌شوند:

$$P_{11} = \{v_1, v_2, v_3, v_4, v_5, v_9, v_{10}\} \qquad P_{12} = \{v_6, v_7, v_8\}$$
$$P_{21} = \{v_1, v_2, v_8, v_9, v_{10}\} \qquad P_{22} = \{v_3, v_4, v_5, v_6, v_7\}$$

نیمهٔ نخست فرزندان $(P_{11}$ و $P_{21})$ همواره معتبر خواهند بود؛ ولی نیمهٔ دوم آنها ممکن است نامعتبر باشد. به همین جهت نیمهٔ دوم به‌صورت ژن‌به‌ژن تا زمانی‌که تناقضی مشاهده نشود به فرزند منتقل می‌شود. درصورت وجود تناقض، کمترین مقدار رنگی ممکن تخصیص داده خواهد شد؛ اگر کمترین مقدار رنگی میسر نباشد، مقدار رنگی به گره‌ای نسبت داده می‌شود که در همسایگی آن کمتر استفاده شده باشد تا کمترین تناقض در فرزند ایجاد شود. ذکر این نکته ضروری است که برای پیمایش نیمهٔ دوم فرزندان اولویت ژن‌ها به جای اینکه به‌صورت ترتیبی (از اندیس کوچک به بزرگ یا از بزرگ به کوچک) باشد، به‌صورت تصادفی انتخاب می‌شود که این کار می‌تواند باعث شود تا تنوع بهتری در تولید فرزندان به وجود آید. البته، در این مثال و برای راحتی کار، اولویت ژن‌ها به‌صورت ترتیبی در نظر گرفته شده است. با توجه به توضیحات، فرزندان زیر تولید خواهند شد:

$$Child_1 = [1\,2\,1\,3\,1\,2\,\bar{3}\,2\,1\,1];$$
$$Child_2 = [2\,3\,1\,\bar{2}\,\bar{\bar{3}}\,1\,\bar{\bar{2}}\,1\,1\,1]$$

در تولید $Child_1$، گرهٔ v_7 نمی‌تواند مقدار رنگی یک را به ارث ببرد؛ لذا، مقدار رنگی سه را، از ژن متناظر در والد نخست (P_1) به ارث می‌برد. در تولید $Child_2$، گره‌های v_4، v_5 و v_7 دارای تناقضند. گره‌های v_5 و v_7 مقدار ژنی هیچ‌کدام از والدین را نمی‌توانند به ارث ببرند؛ بنابراین باید مقدار رنگی‌ای که در همسایگی این گره‌ها استفاده نشده است، انتخاب شود. همان‌طور که مشاهده می‌شود، فرزند $Child_2$ نامعتبر شده. اما، مقدار برازندگی $Child_1$ برابر 04.0 و برازندگی $Child_2$ برابر 16.0 شده و در مقایسه با مقدار برازندگی والدین (برازندگی P_1 برابر 28.0 و برازندگی P_2 برابر 52.0)، فرزندان مناسبی تولید شده است (با توجه به فرمول (۱) هرچه مقدار تابع هزینه کمتر باشد، کروموزوم شایسته‌تر خواهد بود).

۳-۵- عمل‌گر جهش آشوبی هوشمند

عمل‌گر جهش سعی دارد تا با اعمال تغییراتی در ژن‌های کروموزوم، از نقطه‌ای از فضای جستجو به نقطه‌ای دیگر دسترسی پیدا کند. در الگوریتم ژنتیک این عمل‌گر می‌تواند نقش به‌سزایی در فرار از بهینهٔ محلی داشته باشد. به‌طورمعمول برای ایجاد مقادیر ژنی از روش‌های تولید تصادفی اعداد استفاده می‌شود؛ اما، در این مقاله از روشی آشوبی برای تولید اعداد تصادفی استفاده شده است. روش‌های آشوبی با توجه به خاصیتی که دارند، به نسبت

روش‌های تصادفی می‌توانند به‌صورت نویسه و فضای جستجو را بهتر پیمایش کنند؛ اما، اینکه جهش به کدام ژن اعمال شود، مسئله‌ای است که در این مقاله به آن پرداخته شده و برای آن پارامتری در نظر گرفته شده است؛ زیرا، انتخاب ژن مناسب برای جهش در بهبود جواب کلی می‌تواند مؤثر باشد. بنابراین، عمل‌گر جهشی که در این مقاله معرفی شده، عمل‌گر جهش آشوبی هوشمند است. این عمل‌گر، بر روی کروموزوم‌های نامعتبر اعمال می‌شود و ژنی جهش می‌یابد که پیوند ضعیف بیشتری داشته باشد. اگر ژنی فقط با یک ژن در همسایگی خود در تناقض (دو گرهٔ متصل در گراف که دارای رنگ یکسانی باشند) باشد، دارای پیوند ضعیف است. ترتیب انتخاب ژن‌ها برای جهش، براساس فرمول (۳) به‌دست می‌آید. در این فرمول W_i تعداد پیوند ضعیف در اطراف ژن i و $Conf_i$ تعداد تناقضی است که در اطراف آن ژن وجود دارد. ژنی که دارای بیشترین مقدار Z_i باشد، برای رنگ‌آمیزی انتخاب می‌شود.

$$Z_i = \frac{W_i}{Conf_i} \qquad (۳)$$

فرض کنید، کروموزوم $P = [1,3,2,2,2,1,2,3,1,1]$ برای شکل (۱) وجود داشته باشد؛ در این کروموزوم، ژن v_3 با ژن v_4، ژن v_4 با ژن‌های v_3 و v_5، ژن v_5 با ژن‌های v_4 و v_7 و ژن v_7 با ژن v_5 در تناقض هستند. بنابراین، v_3 دارای پیوند ضعیف نیست؛ زیرا، ژن v_4 که با v_3 در تناقض است دارای دو تناقض با ژن‌های v_3 و v_5 است. ژن v_4 دارای یک پیوند ضعیف با v_3 است. بنابراین، ژن‌های v_3 و v_7 پیوند ضعیف ندارند و مقدار Z برای آنها برابر صفر می‌شود و ژن‌های v_4 و v_5 هر کدام دارای یک پیوند ضعیف با یکی از همسایه‌های خود هستند که مقدار Z برای آنها برابر 5.0 می‌شود. در این روش، ژن‌های v_4 و v_5 برای جهش انتخاب می‌شوند.

برای تخصیص مقدار رنگی به ژن انتخاب‌شده (برای مثال ژن v_4)، ابتدا بررسی می‌شود که در اطراف ژن v_4 و یا همسایه‌های آن، مقدار رنگی وجود دارد که در همسایگی آن گره استفاده نشده باشد و بتواند تناقض را برطرف کند. اگر مقدار رنگی مناسب وجود داشت، تخصیص داده می‌شود. در غیر این‌صورت، با استفاده از فرمول آشوبی (۴) مقدار رنگی به ژن مورد نظر تخصیص داده می‌شود. در این فرمول، MaxC بزرگ‌ترین مقدار رنگی در P، P_k مقدار رنگی ژن k در کروموزوم P قبل از جهش، P'_k مقدار رنگی ژن k بعد از جهش و μ میزان نوسانات است. در این مثال، در اطراف ژن v_4 مقدار رنگی مناسب وجود ندارد؛ اما، در اطراف ژن v_3 مقدار رنگی یک قابل تخصیص است و این مقدار به ژن v_3 تخصیص داده می‌شود. بنابراین، کروموزوم نامعتبر در این مثال به کروموزوم معتبر تبدیل می‌شود.

$$P'_k = 1 + (\mu . P_k(MaxC - P_k) \bmod MaxC) \qquad (۴)$$

۳-۶- به‌روزرسانی جمعیت

بعد از اینکه کار عمل‌گرهای ترکیب و جهش به پایان رسید و فرزندان مورد نظر ایجاد شدند؛ سؤال مهم‌تر این است که کدام یک از کروموزوم‌های موجود به نسل آینده منتقل شوند تا الگوریتم کار خود را بر روی آنها ادامه دهد؟ یکی از روش‌هایی که برای به‌روزرسانی جمعیت وجود دارد، استفاده از جمعیت نخبه است, اما، اگر نسل بعد فقط با استفاده از جمعیت نخبه به‌روز شود، آنگاه این امکان وجود دارد که بعد از گذشت چند مرحله و به‌زودی الگوریتم به نقطهٔ بهینهٔ محلی همگرا شود که می‌تواند به‌دلیل ازدست‌رفتن تنوع جمعیت و بسیار شبیه به هم شدن کروموزوم‌ها این امر اتفاق بیفتد. از طرف دیگر مشخص است که برای حفظ تنوع جمعیت و رسیدن به جواب بهینهٔ سراسری، کروموزوم‌های نامعتبر و پرت هم کارا می‌توانند باشند. بنابراین، ایده‌ای که برای این مقاله ارائه شده، روشی ترکیبی است که در آن سعی شده تا از جمعیت‌های نخبه، متوسط و بد جامعه به‌منظور بهبود تنوع جمعیت و فرار از بهینه محلی به‌منظور رسیدن به بهینهٔ سراسری استفاده شود. در این روش ابتدا، اعضای جمعیت براساس میزان برازندگی (تابع هزینه در فرمول (۱)) مرتب و سپس، جمعیت جامعه به سه دستهٔ نخبه، متوسط و بد تقسیم می‌شود. از آنجایی‌که برای رسیدن به جواب بهینه، وجود اعضای نخبه در جمعیت ضروری است، پس پنجاه درصد از جمعیت جدید را جمعیت نخبه تشکیل می‌دهد. آنگاه، سی درصد اعضای دیگر جمعیت با استفاده از روش چرخ رولت و با درنظرگرفتن مقدار تابع برازندگی از جمعیت متوسط جامعه و بیست درصد باقیمانده از بخش بد جامعه با استفاده از روش تصادفی و بدون درنظرگرفتن اولویت برای هر یک از اعضای جامعه انتخاب و به جمعیت جدید اضافه می‌شوند. در این صورت، می‌توان انتظار داشت که نسل جدید دارای پراکندگی مناسبی از فضای جستجو باشد.

۴- پیاده‌سازی و ارزیابی نتایج

برای ارزیابی روش‌های پیشنهادی برای حل مسئلهٔ رنگ‌آمیزی گراف، به‌طورمعمول از گراف‌های DIMACS استفاده می‌شود. برخی از گراف‌های این مجموعه، به‌صورت تصادفی و با استفاده از برنامه‌های تولیدکنندهٔ گراف ساخته شده بودند؛ مانند، گراف‌های سری *DSJ*. برخی از گراف‌ها، از مسائل کاربردی نتیجه می‌شوند؛ به‌عنوان مثال، گراف‌های سری *SCH* که مربوط به مسئلهٔ زمان‌بندی طبقه در هر یک از حالات بدون فضای خالی و با فضای خالی است و یا گراف‌های سری FLAT، که مربوط به مسئلهٔ مربع لاتین است. همچنین، در این رده مسائل، به سری *REG* نیز اشاره می‌توان کرد که بر مبنای مسئلهٔ اختصاص به‌روزرسان است. اغلب، پژوهش‌گران برای بررسی روش‌های خود، از گراف‌های تصادفی *DSJ*، *Le* و *Flat* استفاده کرده‌اند. مهم‌ترین ویژگی‌های حائز اهمیت در هر یک از این گراف‌های آزمون، تعداد رأس‌های آنها و چگالی یال‌هایشان است. نام‌گذاری گراف‌ها نیز بر مبنای این ویژگی‌ها و عدد رنگی گراف درصورت معلوم‌بودن، است. به‌عنوان مثال، گراف DSJC250.5، نام گرافی با تعداد ۲۵۰ رأس و چگالی 5.0 از سری DSJ و یا le450_15c گرافی از سری LE است که ۴۵۰ رأس دارد و عدد رنگی آن پانزده است.

برای ارزیابی روش پیشنهادی در این بخش، ابتدا کارایی عمل‌گر ترکیب در تولید فرزندان معتبر مورد بررسی قرار می‌گیرد؛ بعد از آن با آزمایشی درصد ژن‌های نامعتبر که باید مورد جهش واقع شوند، مشخص می‌شود؛ در ادامه، پیچیدگی زمانی الگوریتم پیشنهادی محاسبه و از نظر تعداد رنگ به‌دست آمده با الگوریتم‌های دیگر مورد مقایسه قرار می‌گیرد.

۴-۱- کارایی عمل‌گر ترکیب ارائه‌شده

در پی این فرضیه که چند درصد از فرزندان تولیدشده با استفاده از عمل‌گر ترکیب CMHn معتبر هستند و کارایی این عمل‌گر در بهبود جواب‌های الگوریتم ژنتیک به چه صورت است، آزمایشی انجام شده و در هر مرحله از اجرای الگوریتم ژنتیک میانگین تابع برازندگی فرزندان تولیدشده با استفاده از این عمل‌گر و همچنین، میزان فرزندان معتبر تولیدشده بر روی برخی از گراف‌های DIMACS محاسبه و آزمایش‌های انجام‌شده بر روی چهار گراف DSJC250.9، Le450_5b، Flatt300 و Zeroni_1 در ادامه و در شکل (۳) گزارش شده است. در آزمایش‌های صورت‌گرفته، مشخص شده است که به‌طور میانگین بیش از ۶۰٪ فرزندان تولیدشده با استفاده از این عمل‌گر، معتبر می‌باشند. با توجه به نتایج به‌دست‌آمده از این آزمایش‌ها می‌توان نتیجه گرفت که این عمل‌گر، می‌تواند عمل‌گر مناسبی در جهت حرکت به‌سوی جواب بهینه باشد.

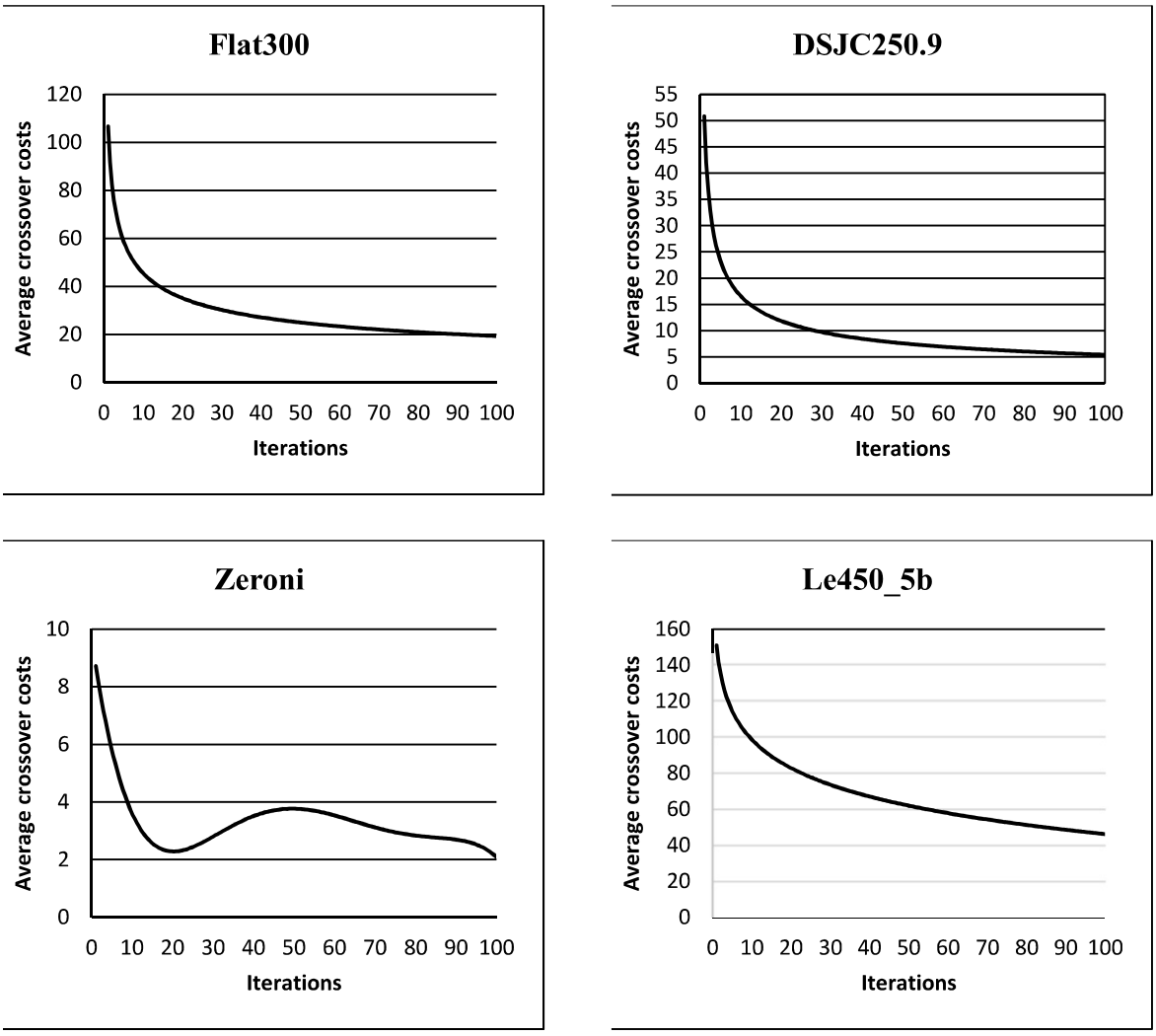
۴-۲- بررسی درصد ژن‌های کروموزوم برای جهش

برای انجام این آزمایش، درصد ژن‌های نامعتبر از هر کروموزوم که باید جهش یابند از 1.0 تا 9.0 در نظر گرفته شده است و مشاهده شد که میزان جهش ژن‌ها از 4.0 به بالا جواب‌های بدتری تولید می‌کنند؛ به‌همین‌دلیل، آزمایش‌های گزارش‌شده در شکل (۳) از 1.0 تا 35.0 با فاصلهٔ 05.0 در نظر گرفته شده است. نتایج این آزمایش‌ها نشان می‌دهد که درصد بهینهٔ تعداد ژن‌های نامعتبر برای جهش 1.0 و 15.0 است.

۴-۳- بررسی کارایی الگوریتم ارائه‌شده از نظر پیچیدگی زمانی

برای بررسی پیچیدگی زمانی روش ارائه‌شده، هر یک از بخش‌های الگوریتم ژنتیک به‌صورت جداگانه و براساس شبه‌کد آنها، تحلیل و پیچیدگی زمانی هر یک محاسبه می‌شود. با توجه به این موضوع، مشخص است که گام نخست در روش پیشنهادی ایجاد جمعیت اولیه است که از روشی ابتکاری استفاده شده که شبه‌کد این مرحله در شکل (۲) نشان داده شده است. همان‌طورکه از این شبه‌کد مشخص است، پیچیدگی زمانی این روش ابتکاری O(N) است که مربوط به قطعه کد درون حلقهٔ While است. این قطعه کد برای انتخاب و رنگ‌آمیزی گره‌های گراف نوشته شده است.

یکی دیگر از بخش‌های مهم روش پیشنهادی عمل‌گر ترکیب است. در روش پیشنهادی حاضر، روش ترکیبی ابتکاری به نام CMHn معرفی شده است که شبه‌کد آن در شکل (۵) آورده شده است. البته، این شبه‌کد برای ایجاد یک فرزند است و برای ایجاد فرزند دوم باید دوباره این تابع با تغییر در اولویت والدین در حین صدا زدن فراخوانی شود. در این شبه‌کد بیشترین پیچیدگی در حلقه‌های For وجود دارد. اگر فرض بر این باشد که به جای داشتن چندین نقطهٔ برش،



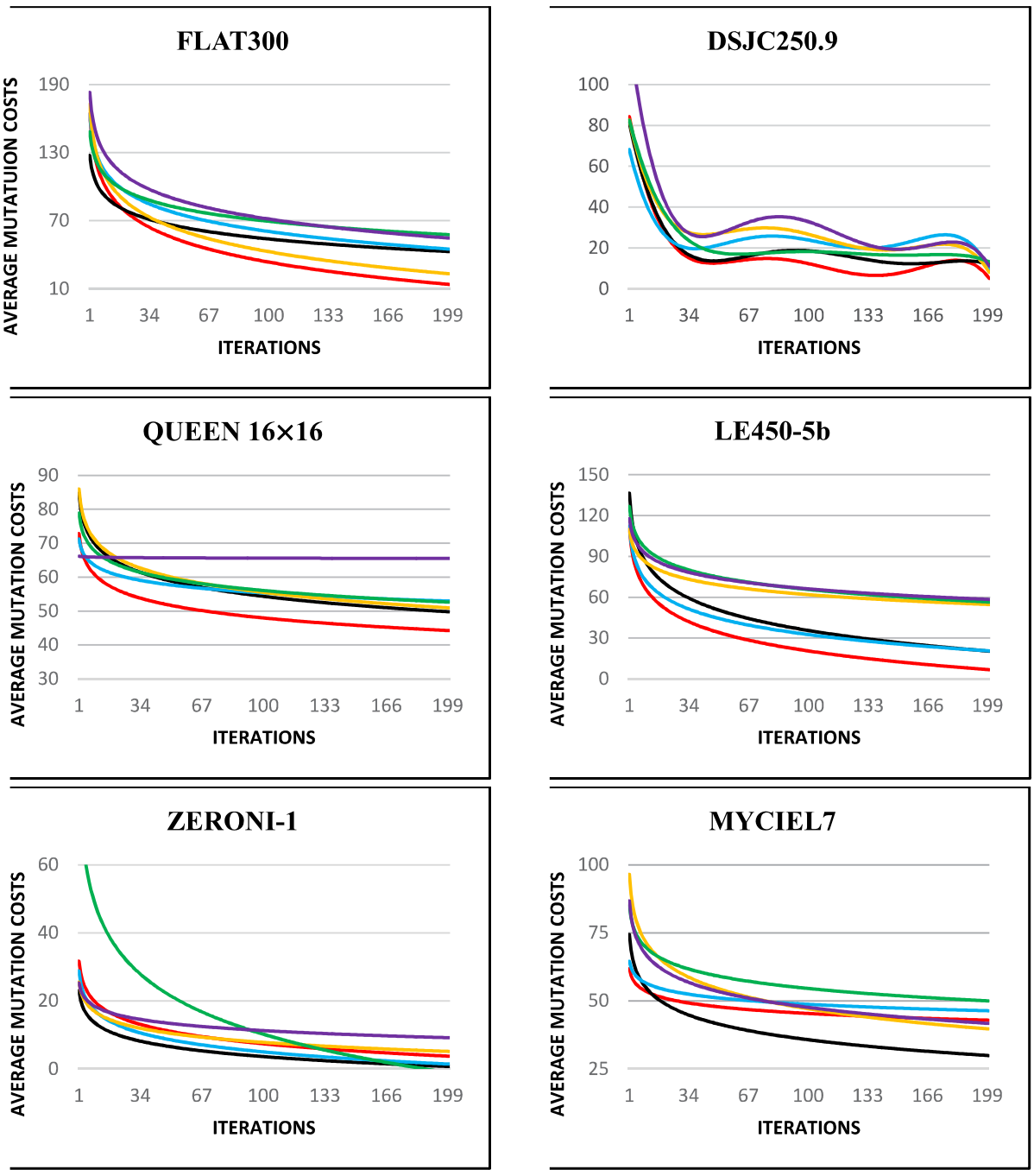
(شکل-۳): نتایج برخی آزمایش‌های انجام‌شده برای بررسی کارایی عمل‌گر ترکیب ارائه‌شده.

(Figure-3): Evaluating the proposed crossover to set crossover parameters.

یک نقطهٔ برش انتخاب و این نقطهٔ برش در اولین ژن کروموزوم انتخاب شود، می‌توان پیچیدگی زمانی این عمل‌گر را O(N) در نظر گرفت. عمل‌گر بعدی که می‌توان بر روی پیچیدگی زمانی آن می‌توان بحث کرد، عمل‌گر جهش است. این عمل‌گر، به‌عنوان جهش آشوبی هوشمند معرفی شده که شبه‌کد آن در شکل (۶) آورده شده است. با توجه به شبه‌کد نوشته‌شده و با توجه به اینکه در بدترین وضعیت اگر فرض شود که تمامی ژن‌های کروموزوم دارای تناقض باشند، آنگاه پیچیدگی زمانی آن برابر O(N) می‌شود. البته، بیشترین پیچیدگی زمانی برای روش پیشنهادی در محاسبه تعداد تناقض هر کروموزوم است که از حلقه تو در توی For برای پیمایش کروموزوم تشکیل می‌شود. این پیچیدگی برابر با $O(N^2)$ است؛ بنابراین، نتیجه می‌توان گرفت که پیچیدگی زمانی روش حاضر $O(N^2)$ است.

۴-۴- بررسی کارایی الگوریتم ارائه‌شده از نظر تعداد رنگ استفاده‌شده

برای بررسی کارایی روش حاضر از برخی نمونه گراف‌های DIMACS و گراف‌های Queen استفاده شده است. مشخصات گراف‌های مورد مقایسه در جدول (۲) آمده و بهترین جواب، بدترین جواب و میانگین ده بار اجرای روش پیشنهادی بر روی هر یک از گراف‌های مورد مقایسه در جدول (۳) آورده شده است؛ درنهایت، بهترین جواب روش پیشنهادی با الگوریتم‌های ارائه‌شده توسط سایر پژوهش‌گران مقایسه و نتایج آن در جدول‌های (۴) تا (۸) ارائه شده است. در این جداول، k^* عدد رنگی بهینه و علامت "-" نشان‌دهندهٔ این است که برای آن گراف در مقالهٔ مربوطه مقداری گزارش نشده است. همچنین، در بررسی‌های



(شکل-۴): آزمایش‌های انجام‌شده برای بررسی درصد بهینه از تعداد ژن‌های نامعتبر یک کرموزوم که باید جهش یابند.

(Figure-4): Setting mutation parameter based on some experiments.

انجام‌شده جواب‌هایی که بهتر از دیگران شده به‌صورت پر رنگ در آمده است.

تعداد اعمال عمل‌گر ترکیب در هر تکرار هفتاد درصد جمعیت، تعداد اعمال عمل‌گر جهش در هر تکرار بیست درصد جمعیت و درصد ژن‌های جهش‌یافته برای هر کروموزوم در جهش آشوبی هوشمند پانزده درصد ژن‌های نامعتبر در نظر گرفته شده است. روش پیشنهادی ده بار و هر بار ۱۵۰ تکرار بر روی هر گراف اجرا و بهترین جواب در نتایج آورده شده است. همچنین، این نکته قابل ذکر است که الگوریتم پیشنهادی با استفاده از نرم‌افزار متلب ۲۰۱۲ شبیه‌سازی و بر روی سیستمی با پردازنده اینتل core i5-3337U 1.8GHZ با چهار گیگابایت حافظهٔ اصلی و ویندوز ۸ اجرا شده است.

1. Function CMHn(P_1, P_2) // P_1, P_2 are input parents
2. Create an empty Child
3. If P_1 is valid
4. $CutPoint1$= select a number in [**1**..N-**1**] randomly
5. Else
6. $CutPoint1$ = the first conflicting position in P_1
7. End
8. $CutPoint2$= select a number in [$CutPoint1$..N-**1**] randomly
9. $Temp_1 = P_1(CutPoint2 + 1 : N)$
10. $Child(1 : CutPoint1) = P_1(1 : CutPoint1)$
11. $P_{12} = P_1(CutPoint1 + 1 : CutPoint2)$
12. For i=**1** to $SizeTemp_1$
13. If $Temp_1(i)$ hasn't any conflicting color in $Child()$
14. $Child(index\ of\ Temp_1(i)\ in\ P_1) = Temp_1(i)$
15. Else
16. $P_{12}(SizeP_{12} + 1) = Temp_1(i)$
17. End if
18. End for
19. For i = **1** to $SizeP_{12}$
20. If $P_{22}(i)$ hasn't any conflicting color in $Child()$
21. $Child(index\ of\ P_{22}(i)\ in\ P_2) = P_{22}(i)$
22. Else if $P_{12}(i)$ hasn't any conflicting color in $Child()$
23. $Child(index\ of\ P_{12}(i)\ in\ P_1) = P_{12}(i)$
24. Else
25. $Child(index\ of\ P_{22}(i)\ in\ P_2) = \emptyset$
26. End if
27. End for
28. Color all empty gens in Child by heuristic algorithm.
29. Return $Child$

(شکل- ۵): شبه‌کد عمل‌گر ترکیب CMHn

(Figure-5): Pseudocode for crossover operator.

1. Input: L //Number of gens to be mutated
2. Output: Child // a offspring by mutation
3. b = Select a chromosome which has conflict from population
4. Child = b
5. For i=**1** to conflict number of child
6. Calculate the Z_i of equation (**3**)
7. End for
8. for i = **1** to L
9. k = Select a conflict gen of chromosome b by roulette wheel selection mechanism
10. μ = Select a number in [**1**..N] randomly
11. $Child_k = 1 + (\mu . b_k(MaxC - b_k) \bmod MaxC)$
12. End for
13. return Child

(شکل- ۶): شبه‌کد عمل‌گر جهش آشوبی.

(Figrue-6): Pseudocode for mutation operator.

(جدول-۲): مشخصات گراف‌های مورد مقایسه.

(Table-2): Graphs in dataset and their specifications.

نام گراف	یال	رأس	K*
DSJC**125.1**	7361	125	5
DSJC**125.5**	38226	125	17
DSJC**125.9**	69613	125	44
DSJC**250.1**	27897	250	8
DSJC**250.5**	156464	250	28
DSJC**250.9**	279338	250	72
DSJC**500.1**	62624	500	12
DSJC**500.5**	312780	500	48
DSJC**500.9**	1121749	500	127
DSJC**1000.1**	249826	1000	20
DSJR**500.1c**	121275	500	85
DSJR**500.5**	588662	500	122
wap**01a**	110871	2368	41
wap**02a**	111742	2464	40
wap**03a**	286722	4730	50
wap**04a**	294902	5231	42
wap**05a**	503081	4464	50
wap**06a**	404571	2947	40
wap**07a**	403368	2809	41
wap**08a**	404176	2870	45
Queen **6×6**	290	36	7
Queen **8×8**	728	64	9
Queen **9×9**	1056	81	10
Queen **10×10**	1470	100	11
Queen **11×11**	1980	121	11

نام گراف	یال	رأس	K*
le**450_5**a	5714	450	5
le**450_5**b	5734	450	?
le**450_5**c	9803	450	?
le**450_5**d	9757	450	?
le450_1**5**a	8168	450	5
le450_1**5**b	8169	450	5
le450_1**5**c	16680	450	5
le450_1**5**d	16750	450	5
le450_2**5**a	8260	450	5
le450_2**5**b	8263	450	5
le450_2**5**c	17343	450	5
le450_2**5**d	17425	450	5
Latin **qg.order40.col**	307350	3600	7
qg.order60.col	612400	6600	10
ash331GPIA	4185	662	?
ash608GPIA	7844	1216	?
ash958GPIA	12506	1916	?
1-Insertions_5	1227	502	?
2-Insertions-4	541	149	?
Queen **12×12**	2596	144	12
Queen **13×13**	3328	169	13
Queen **14×14**	4186	196	14
Queen **15×15**	5180	225	15
Queen **16×16**	6320	256	16

در جدول (۴) نتایج الگوریتم پیشنهادی با الگوریتم سایر پژوهش‌گران در گراف‌های DSJ که از گراف‌های تصادفی هستند، در یافتن کمترین مقدار رنگی مقایسه شده است. همان‌طورکه در این جدول مشاهده می‌شود، الگوریتم پیشنهادی در دوازده گراف مورد مقایسه در این جدول، نسبت به الگوریتم [32] در هفت مورد جوابی بهتر و در دو مورد جوابی برابر داده است (در سه مورد جوابی برای این الگوریتم در مقالهٔ مربوطه ارائه نشده است). همچنین، در این مقایسه روش پیشنهادی نسبت به الگوریتم [33] در یازده گراف جواب بهتر و در یک مورد جواب برابر، نسبت به الگوریتم [34] در هشت مورد جواب بهتر، در سه مورد جواب برابر (در یک مورد جواب ارائه نشده است)، نسبت به الگوریتم [35] در شش مورد جواب بهتر، در دو مورد جواب برابر و در دو مورد جواب بدتر (در دو مورد جوب ارائه نشده است)، نسبت به الگوریتم [36] در پنج مورد جواب بهتر، در سه مورد جواب برابر و در دو مورد جواب بدتر (در دو مورد جواب ارائه نشده است)، نسبت به الگوریتم [37] در پنج مورد جواب بهتر (در هفت مورد جواب ارائه نشده است)، نسبت به الگوریتم [38] در یک مورد جواب بهتر، در پنج مورد جواب برابر و در شش مورد جواب بدتر شده است. همان‌طورکه در این مقایسه مشخص است، روش پیشنهادی فقط نسبت به الگوریتم [38] جوابی بدتر ارائه داده است و در مقایسه با بقیهٔ الگوریتم‌ها به‌طورتقریبی برتر بوده است.

(جدول-۳): نتایج بهترین، بدترین و میانگین جواب در 10 بار اجرای روش پیشنهادی.
(Table-3): Best, worst and mean results in 10 times executions.

نام گراف	K*	بهترین	بدترین	میانگین	نام گراف	K*	بهترین	بدترین	میانگین
DSJC**125.1**	5	5	5	5	le**450_5**a	5	5	5	5
DSJC**125.5**	17	17	18	2.17	le**450_5**b	5	5	5	5
DSJC**125.9**	44	44	47	45	le**450_5**c	5	5	6	4.5
DSJC**250.1**	8	8	9	4.8	le**450_5**d	5	5	6	2.5
DSJC**250.9**	72	75	76	4.75	le450_15a	15	16	17	2.16
DSJC**250.5**	28	30	30	30	le450_15b	15	18	18	18
DSJC**500.1**	12	14	14	14	le450_15c	15	17	18	4.17
DSJC**500.5**	48	52	53	2.52	le450_15d	15	17	18	6.17
DSJC**500.9**	126	131	135	2.132	le450_25a	25	25	25	25
DSJC**1000.1**	20	22	22	22	le450_25b	25	25	25	25
DSJR**500.1**c	84	85	86	6.85	le450_25c	25	27	28	4.27
DSJR**500.5**	122	124	126	8.124	le450_25d	25	28	30	8.28
wap**01**a	؟	43	45	44	Latin	97	103	108	4.105
wap**02**a	؟	43	45	44	qg.order**40**.col	40	40	40	40
wap**03**a	؟	48	49	6.48	qg.order**60**.col	60	60	61	4.60
wap**04**a	؟	46	46	46	ash**331**GPIA	؟	4	4	4
wap**05**a	؟	45	46	8.45	ash**608**GPIA	؟	4	4	4
wap**06**a	؟	43	45	8.43	ash**958**GPIA	؟	4	4	5
wap**07**a	؟	43	44	8.43	**1**-Insertions_**5**	؟	5	6	2.5
wap**08**a	؟	41	45	2.42	**2**-Insertions-**4**	4	4	5	4.4

Queen **6×6**	7	7	8	6.7	Queen **12×12**	12	14	15	6.14
Queen **8×8**	9	9	11	10	Queen **13×13**	13	16	17	4.16
Queen **9×9**	10	10	12	2.11	Queen **14×14**	14	17	19	18
Queen **10×10**	11	11	12	4.11	Queen **15×15**	15	17	20	18
Queen **11×11**	11	12	14	8.12	Queen **16×16**	16	18	20	8.18

بررسی‌های انجام‌شده به‌منظور ارزیابی کارایی الگوریتم پیشنهادی در گراف‌های Le450 با الگوریتم‌های پیشنهادی سال‌های 2008، 2012، 2014 و 2016 در جدول (۵) آورده شده است. با توجه به نتایج این جدول، مشخص است که الگوریتم پیشنهادی نسبت به الگوریتم‌های [33]،[36] بهتر و نسبت به الگوریتم‌های [13]،[39] به‌طورتقریبی بدتر شده است. به‌خصوص نسبت به الگوریتم [13] که الگوریتم پیشنهادی حتی در یک گراف از نوع گراف‌های Le450 نتوانسته جوابی بهتر را ارائه دهد. البته، این نکته نیز قابل ذکر است که الگوریتم [13] در این مقایسه نسبت به تمامی الگوریتم‌های مورد مقایسه جوابی بهتر ارائه داده است؛ اما، روش پیشنهادی در مقایسه با الگوریتم [39] توانسته در دو مورد جوابی بهتر ارائه دهد. همچنین، می‌توان اذعان داشت که الگوریتم پیشنهادی به‌طورتقریبی با الگوریتم [34] در گراف‌های Le450 یکسان عمل می‌کند؛ هر چند که الگوریتم پیشنهادی توانسته در دو مورد نسبت به این الگوریتم جوابی بهتر و در یک مورد نیز جوابی بدتر ارائه دهد؛ اما در مجموع می‌توان این دو الگوریتم را برای رسیدن به کمترین مقدار رنگی در گراف‌های Le450 یکسان فرض کرد.

(جدول-۴): مقایسهٔ الگوریتم پیشنهادی با الگوریتم‌های پیشنهادی سایر پژوهش‌گران در گراف‌های DSJ.

(Table-4): Comparison of proposed methods with recent studies for DSJ graphs in DIMACS dataset.

نام گراف	K*	روش پیشنهادی	مرجع [۳۲]	مرجع [۳۳]	مرجع [۳۴]	مرجع [۳۵]	مرجع [۳۶]	مرجع [۳۷]	مرجع [۳۸]
DSJC**125.1**	5	5	6	5	5	8	7	6	6
DSJC**125.5**	17	17	18	20	19	20	18	19	17
DSJC**125.9**	44	44	44	47	46	46	44	45	44
DSJC**250.1**	8	8	9	9	9	9	9	10	8
DSJC**250.9**	72	75	75	88	82	74	74	-	72
DSJC**250.5**	28	30	31	36	34	32	31	33	28
DSJC**500.1**	12	14	15	15	14	14	14	-	12
DSJC**500.5**	48	52	55	63	62	53	53	-	48
DSJC**500.9**	126	131	136	161	-	130	126	-	126
DSJC**1000.1**	20	22	-	26	25	22	22	-	20
DSJR**500.1c**	84	85	-	87	85	-	-	-	85
DSJR**500.5**	122	124	-	131	130	-	-	-	124
بهتربودن الگوریتم پیشنهادی			7	11	8	6	5	5	1
برابربودن الگوریتم پیشنهادی			2	1	3	2	3	0	5
بدتربودن الگوریتم پیشنهادی			0	0	0	2	2	0	6

(جدول-۵): مقایسهٔ الگوریتم پیشنهادی با الگوریتم‌های پیشنهادی سایر پژوهش‌گران در گراف *Le450*.

(Table-5): Comparison of proposed methods with recent studies for Le450 graphs in DIMACS dataset.

نام گراف	K*	روش پیشنهادی	مرجع [۳۳]	مرجع [۱۳]	مرجع [۳۴]	مرجع [۳۶]	مرجع [۳۹]
le**450_5**a	5	5	9	5	–	5	10
le**450_5**b	5	5	9	5	–	5	10
le**450_5**c	5	5	-	5	5	5	5
le**450_5**d	5	5	10	5	5	5	5
le450_15a	15	16	17	15	16	19	-
le450_15b	15	18	17	15	16	19	15
le450_15c	15	17	24	15	22	16	15
le450_15d	15	17	23	15	23	17	15
le450_25a	25	25	–	25	25	27	25
le450_25b	25	25	–	25	25	26	25
le450_25c	25	27	28	26	–	31	26
le450_25d	25	28	28	26	–	29	27
بهتربودن الگوریتم پیشنهادی			7	0	2	6	2
برابربودن الگوریتم پیشنهادی			1	6	5	5	4
بدتربودن الگوریتم پیشنهادی			1	6	1	1	5

در ارزیابی بعدی، الگوریتم پیشنهادی با الگوریتم‌های [13]،[33]،[34]،[36] در گراف‌های بزرگ wap مقایسه و نتایج آن در جدول (۶) آورده شده است. با توجه به این جدول مشخص است که الگوریتم پیشنهادی در این مقایسه توانسته در برابر تمامی الگوریتم‌های مورد مقایسه جوابی بهتر ارائه دهد. همان‌طورکه در این جدول ملاحظه می‌شود، حتی در یک مورد الگوریتم‌های مورد مقایسه نتوانستند جوابی برابر با الگوریتم پیشنهادی ارائه دهند و در این مقایسه الگوریتم پیشنهادی برتری محسوسی نسبت به الگوریتم‌های مورد مقایسه دارد.

(جدول-۶): مقایسهٔ الگوریتم پیشنهادی با الگوریتم‌های پیشنهادی سایر پژوهش‌گران در گراف‌های *Wap*

(Table-6): Comparison of proposed methods with recent studies for Wap graphs in DIMACS dataset.

نام گراف	K*	روش پیشنهادی	مرجع [۳۳]	مرجع [۱۳]	مرجع [۳۴]	مرجع [۳۶]
wap**01**a	؟	43	46	45	46	–
wap**02**a	؟	43	45	44	46	–
wap**03**a	؟	48	56	53	–	–
wap**04**a	؟	46	50	48	–	–
wap**05**a	؟	45	51	50	50	51
wap**06**a	؟	43	44	44	47	48
wap**07**a	؟	43	46	45	44	50
wap**08**a	؟	41	47	45	45	50
بهتربودن الگوریتم پیشنهادی			8	8	6	4
برابربودن الگوریتم پیشنهادی			0	0	0	0
بدتربودن الگوریتم پیشنهادی			0	0	0	0

در جدول (۷) مقایسه‌ای بر روی برخی دیگر از گراف‌های DIMACS ارائه شده است. همان‌طور که در این جدول نیز مشاهده می‌شود، الگوریتم پیشنهادی در هیچ‌یک از گراف‌های مورد آزمایش جوابی بدتر از الگوریتم‌های مورد مقایسه نداده است. حتی در سه مورد Latin، 1-Insertions_5 و 2-Insertions_4 جواب‌ها بهتر از تمامی الگوریتم‌های مورد مقایسه شده است. ذکر این نکته نیز ضروری است که الگوریتم [37] در هیچ‌یک از موارد نتوانسته جوابی برابر با الگوریتم پیشنهادی ارائه کند که این خود نشان از کارایی الگوریتم پیشنهادی در یافتن جواب‌های بهینه دارد.

(جدول-۷): مقایسهٔ الگوریتم پیشنهادی با الگوریتم‌های پیشنهادی سایر پژوهش‌گران در برخی دیگر از گراف‌ها.

(Table-7): Comparison of proposed methods with recent studies for some samples of DIMACS dataset.

نام گراف	K^*	روش پیشنهادی	مرجع [۳۳]	مرجع [۱۳]	مرجع [۳۴]	مرجع [۳۷]
Latin	97	103	129	104	130	-
qg.order**40**.col	40	40	42	-	40	41
qg.order**60**.col	60	60	63	60	61	-
ash**331**GPIA	؟	4	4	5	4	5
ash**608**GPIA	؟	4	4	5	4	5
ash**958**GPIA	؟	4	5	6	4	5
1-Insertions_**5**	؟	5	6	6	6	6
2-Insertions-**4**	4	4	5	5	5	5
بهتربودن الگوریتم پیشنهادی			6	6	4	6
برابربودن الگوریتم پیشنهادی			2	1	4	0
بدتربودن الگوریتم پیشنهادی			0	0	0	0

در مقایسه‌ای دیگر که در جدول (۸) ارائه شده است روش پیشنهادی با الگوریتم‌های دیگر در یافتن کمترین مقدار رنگی در گراف‌های *Queen* مقایسه شده است. در این جدول مشخص است که الگوریتم پیشنهادی تنها در دو مورد جوابی بدتر از الگوریتم [40] ارائه داده و در تمامی موارد دیگر نسبت به الگوریتم‌های دیگر جواب بدتری را ارائه نکرده است. به‌طورتقریبی، می‌توان این ادعا را داشت که این الگوریتم در این مقایسه نسبت به الگوریتم‌های قابل مقایسه، دارای برتری محسوسی است و این الگوریتم‌ها تنها در سه گراف از ده گراف مورد مقایسه (در بهترین حالت) توانستند که جوابی برابر با الگوریتم پیشنهادی ارائه دهند که آن هم متعلق به الگوریتم [40] است.

(جدول-۸): مقایسهٔ الگوریتم پیشنهادی با الگوریتم‌های پیشنهادی سایر پژوهش‌گران در برخی از گراف‌های *Queen*

(Table-8): Comparison of proposed methods with recent studies For Queen graphs in DIMACS dataset.

نام گراف	K^*	روش پیشنهادی	مرجع [۳۳]	مرجع [۴۰]	مرجع [۳۴]	مرجع [۳۷]
Queen **6×6**	7	7	-	-	8	8
Queen **8×8**	9	9	10	9	10	11
Queen **9×9**	10	10	11	10	11	10
Queen **10×10**	11	11	12	12	12	14
Queen **11×11**	11	12	14	13	14	-

Queen **12×12** 12	14	15	14	15	–
Queen **13×13** 13	16	16	15	16	–
Queen **14×14** 14	17	17	16	17	–
Queen **15×15** 15	17	18	18	18	–
Queen **16×16** 16	18	20	19	19	–
بهتربودن الگوریتم پیشنهادی		8	4	8	3
برابربودن الگوریتم پیشنهادی		2	3	2	1
بدتربودن الگوریتم پیشنهادی		0	2	0	0

درنتیجه و با توجه به مقایسه‌های انجام‌شده در جدول‌های (۴) تا (۸)، ملاحظه می‌شود که روش پیشنهادی جواب‌های قابل قبولی را در یافتن کمترین مقدار رنگی می‌تواند ارائه دهد. به‌خصوص در گراف‌های بزرگ *Wap* (مقایسهٔ جدول (۶))، گراف‌های کوچک (مقایسهٔ جدول (۷)) و گراف‌های *Queen* (مقایسهٔ جدول (۸))، روش پیشنهادی توانسته جوابی بهتر از روش‌های مقایسه‌شدهٔ دیگر به‌دست آورد.

۵-نتیجه‌گیری و کارهای آینده

در این پژوهش، به حل مسئلهٔ رنگ‌آمیزی گراف با استفاده از الگوریتم ژنتیک و پدیدهٔ آشوب پرداخته شد. مسئلهٔ رنگ‌آمیزی گراف، تخصیص رنگ به گره‌های گراف با درنظرگرفتن اینکه، هیچ دو گرهٔ متصل به هم با یالی مشترک هم‌رنگ نباشند و کمترین تعداد مقدار رنگی برای رنگ‌آمیزی استفاده شود می‌باشد. در روش حاضر، برای ایجاد جمعیت اولیه از روشی ابتکاری استفاده شده است تا الگوریتم بخشی از زمان خود را صرف بهبود جواب‌های بسیار بد نکند و بتواند در بازه‌ای مناسب از مقادیر رنگی و فضای جستجو، کار خود را شروع کند. البته، برای حفظ تنوع جمعیت استفاده از روش تصادفی برای ایجاد جمعیت اولیه مناسب است؛ بنابراین، در روش حاضر نیمی از جمعیت اولیه با استفاده از روش ابتکاری و نیمی دیگر با استفاده از روش تصادفی ایجاد شده است. همچنین، در روش حاضر عمل‌گر ترکیب چندنقطه‌ای ابتکاری و عمل‌گر جهش آشوبی هوشمند ارائه شده است. در عمل‌گر ترکیب، چند نقطهٔ برش برای کروموزوم در نظر گرفته می‌شود و براساس نقطه برش‌های انتخاب‌شده یکی از زیرمجموعه‌ها معتبر و دیگری می‌تواند نامعتبر باشد. در این صورت، با تثبیت زیر مجموعهٔ نخست در فرزند و پیمایش زیرمجموعهٔ دوم در والد بعدی و تثبیت مقدار رنگی در فرزند، در صورتی‌که تناقض بوجود نیاید، فرزند مورد نیاز ساخته می‌شود. این عمل‌گر با استفاده از روشی ابتکاری سعی در تولید فرزندان همیشه‌معتبر دارد. در عمل‌گر جهش آشوبی هوشمند، برای اینکه بتوان به‌صورت کاراتر و بهتر فضای جستجو را پیمایش کرد از فرمولی آشوبی برای ایجاد جهش استفاده شده است. البته، برای ایجاد جهش از روش تصادفی نیز استفاده می‌توان کرد؛ اما، روش تصادفی فقط عدد تصادفی ایجاد می‌کند و توجهی به اینکه مقدار رنگی ژن در زمان فعلی چیست ندارد؛ بنابراین، می‌توان نتیجه گرفت که روش تصادفی نمی‌تواند به‌صورت منظم و منسجم فضای جستجو را پیمایش کند. در عمل‌گر جهش معرفی‌شده به این نکته که کدام ژن جهش یابد نیز توجه شده است و برای آن فرمولی که بر پایهٔ پیوند ضعیف است، تعریف شده است و طبق آن گره‌ای انتخاب و مورد جهش قرار می‌گیرد. در این صورت، اگر رنگی در اطراف گرهٔ خالی باشد به گره نسبت داده می‌شود. در غیر این صورت، با استفاده از فرمول آشوبی، مقدار رنگی جدیدی با توجه به مقدار رنگی قبلی به گره نسبت داده می‌شود. همچنین، طبق آزمایش‌هایی نشان داده شده است که حالت بهینه برای درصد ژن‌های مورد جهش، ۱۵٪ است.

در روش حاضر، برای به‌روزرسانی جمعیت از روشی ترکیبی استفاده شده که در آن به وجود هر قشر از جامعه (خوب، متوسط و بد) در جمعیت جدید توجه شده است؛ زیرا اگر فقط از جمعیت نخبه در جمعیت جدید استفاده شود، آنگاه این امکان وجود دارد که بعد از گذشت چند مرحله تنوع جمعیت از بین رود و الگوریتم به هم‌گرایی زودرس دست یابد که به‌طورمعمول این هم‌گرایی به نقطهٔ بهینهٔ محلی صورت می‌گیرد. از طرفی دیگر، مشخص است که وجود جمعیت بد در جمعیت جدید می‌تواند در بهبود جواب و حفظ تنوع جمعیت مؤثر باشد. به همین دلایل در

این روش، از روشی ترکیبی که جمعیت را به سه قسمت خوب، متوسط و بد تقسیم می‌کند و از هر بخش به‌ترتیب مقدار ۵۰، ۳۰ و ۲۰ درصد از جمعیت جدید را بر می‌دارد، استفاده شده است. با این روش می‌توان ادعا داشت که تنوع مناسبی از جمعیت در هر بار اجرای الگوریتم پیشنهادی وجود خواهد داشت.

برای ارزیابی روش پیشنهادی حاضر از نمونه گراف‌های DIMACS و Queen استفاده شده است. بررسی‌ها بر روی این گراف‌ها و مقایسهٔ آنها با الگوریتم‌های ارائه‌شده توسط سایر پژوهش‌گران، حاکی از آن است که الگوریتم پیشنهادی جواب‌های قابل قبول در یافتن کمترین تعداد رنگی را به‌دست می‌آورد. با توجه به اینکه، با افزایش ابعاد گراف‌ها، کارایی روش پیشنهادی برتری چشم‌گیری نسبت به سایر روش‌ها دارد. در پژوهش آینده، پیشنهاد می‌شود به‌روزرسانی جمعیت براساس فاصله باشد؛ به‌طوری‌که بتوان تنوع جمعیت را بهبود بخشید. همچنین، می‌توان نظارت بهتری بر پارامترهای الگوریتم ژنتیک داشت تا بتوان در شرایط مختلف و براساس نیاز، مقدار آنها را تغییر داد. به‌عنوان‌مثال، درصد ترکیب و جهش و حتی انتخاب کدام ترکیب و جهش برای اعمال بر والدین می‌تواند انتخاب و تنظیم پویا انجام شود. علاوه‌براین، استفاده از روش پیوند مسیر (path-relinking) به‌عنوان جستجوی محلی ممکن است منجر به جواب‌های بهتری شود [41].

6-Refrences ۶- مراجع

[1] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*, ed: Springer, 1972, pp. 85-103.

[2] F. T. Leighton, "A graph coloring algorithm for large scheduling problems," *Journal of research of the national bureau of standards,* vol. 84, pp. 489-506, 1979.

[3] N. R. Sabar, M. Ayob, R. Qu, and G. Kendall, "A graph coloring constructive hyper-heuristic for examination timetabling problems," *Applied Intelligence,* vol. 37, pp. 1-11, 2012.

[4] J. Riihijarvi, M. Petrova ,and P. Mahonen, "Frequency allocation for WLANs using graph colouring techniques," in *Wireless On-demand Network Systems and Services, 2005. WONS 2005. Second Annual Conference on*, 2005, pp. 216-222.

[5] G. Chaitin, "Register allocation and spilling via graph coloring," *Acm Sigplan Notices,* vol. 39, pp. 66-74, 2004.

[6] D. Brélaz, "New methods to color the vertices of a graph," *Communications of the ACM,* vol. 22, pp. 251-256, 1979.

[7] J. C. Culberson and F. Luo, "Exploring the k-colorable landscape with iterated greedy," *Cliques, coloring, and satisfiability: second DIMACS implementation challenge,* vol. 26, pp. 245-284, 1996.

[8] C. Avanthay, A. Hertz, and N. Zufferey, "A variable neighborhood search for graph coloring," *European Journal of Operational Research,* vol. 151, pp. 379-388, 2003.

[9] D. C. Porumbel, J.-K. Hao, and P. Kuntz, "A search space “cartography” for guiding graph coloring heuristics," *Computers & Operations Research,* vol. 37, pp. 769-778, 2010.

[10] M. Chams, A. Hertz, and D. De Werra , "Some experiments with simulated annealing for coloring graphs," *European Journal of Operational Research,* vol. 32, pp. 260-266, 1987.

[11] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning," *Operations research,* vol. 39, pp. 378-406, 1991.

[12] Y. Wang, C. Zhang, and Z. Liu, "A matrix approach to graph maximum stable set and coloring problems with application to multi-agent systems," *Automatica,* vol. 48, pp. 1227-1236, 2012.

[13] P. Galinier, A. Hertz, and N. Zufferey, "An adaptive memory algorithm for the k-coloring problem," *Discrete Applied Mathematics,* vol. 156, pp. 267-279, 2008.

[14] J.-P. Hamiez and J.-K. Hao, "Scatter search for graph coloring," in *Internation-al Conference on Artificial Evolution (Evolution Artificielle)*, 2001, pp. 168-179.

Electrical and Computer Engineering, vol. 31, pp. 149-158, 2006.

[25] M. Rocha and J. Neves, "Preventing premature convergence to local optima in genetic algorithms via random offspring generation," in *International Conference on Industrial, Engineering and Other Applica-tions of Applied Intelligent Systems*, 1999, pp. 127-136.

[26] Q. Lü, G. Shen, and R. Yu, "A chaotic approach to maintain the population diversity of genetic algorithm in network training," *Computational Biology and Chemistry,* vol. 27, pp. 363-371, 2003.

[27] G. Sadeghi Bajestani, A. Monzavi, and S. M. R. Hashemi Golpaygani, "Precisely chaotic models survey with Qualitative Bifurcation Diagram," *Signal and Data Processing,* vol. 13, pp. 17-34, 2016.

[28] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of the atmospheric sciences,* vol. 20, pp. 130-141, 1963.

[29] J. Determan and J. A. Foster, "Using chaos in genetic algorithms," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, 1999, pp. 2094-2101.

[30] M.-Y. Cheng and K.-Y. Huang, "Genetic algorithm-based chaos clustering approach for nonlinear optimization," *Journal of Marine Science and Technology,* vol. 18, pp. 435-441, 2010.

[31] C.-T. Cheng, W.-C. Wang, D.-M. Xu, and K. Chau, "Optimizing hydropower reservoir operation using hybrid genetic algorithm and chaos," *Water Resources Management,* vol. 22, pp. 895-909, 2008.

[32] E. Salari and K. Eshghi, "An ACO algorithm for graph coloring problem," in *Computational Intelligence Methods and Applications, 2005 ICSC Congress on*, 2005, p. 5 pp.

[33] I. Méndez-Díaz and P. Zabala, "A cutting plane algorithm for graph coloring," *Discrete Applied Mathematics,* vol. 156, pp. 159-179, 2008.

[34] P. San Segundo, "A new DSATUR-based algorithm for exact vertex coloring," *Comp-*

[15] E. Malaguti, M. Monaci, and P. Toth, "A metaheuristic approach for the vertex coloring problem," *INFORMS Journal on Computing,* vol. 20, pp. 302-316, 2008.

[16] E. Malaguti, M. Monaci, and P. Toth, "An exact approach for the vertex coloring problem," *Discrete Optimization,* vol. 8, pp. 174-190, 2011.

[17] I. Blöchliger and N. Zufferey, "A graph coloring heuristic using partial solutions and a reactive tabu scheme," *Computers & Operations Research,* vol. 35, pp. 960-975, 2008.

[18] T. Park and K. R. Ryu, "A dual-population genetic algorithm for adaptive diversity control," *IEEE transactions on evolutionary computation,* vol. 14, pp. 865-884,2010.

[19] Y. Tan, G. Tan, and X. Wu, "Hybrid real-coded genetic algorithm with chaotic local search for global optimization," *JOURNAL OF INFORMATION &COMPUTATIONAL SCIENCE,* vol. 8, pp. 3171-3179, 2011.

[20] M. Last and S. Eyal, "A fuzzy-based lifetime extension of genetic algorithms," *Fuzzy sets and systems,* vol. 149, pp. 131-147, 2005.

[21] C. A. Glass and A. Prügel-Bennett, "Genetic algorithm for graph coloring: exploration of Galinier and Hao's algorithm," *Journal of Combinatorial Optimization* ,vol. 7, pp. 229-236, 2003.

[22] D. C. Porumbel, J.-K. Hao, and P. Kuntz, "An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring," *Computers & Operations Research,* vol. 37, pp. 1822-1832, 2010.

[23] f .hoseinkhani and b. nasersharif, "Two Featuer Transformation Methods Based on Genetic Algorithm for Reducing Support Vector Machine Classification Error," *Signal and Data Processing,* vol. 12, pp. 23-39, 2015.

[24] S. Garcia, M. Saad, and O. Akhrif, "Nonlinear tuning of aircraft controllers using genetic global optimization: A new periodic mutation operator," *Canadian Journal of*

uters & Operations Research, vol. 39, pp. 1724-1733, 2012.

[35] S. M. Douiri and S. Elbernoussi, "An Effective Ant Colony Optimization Algor-ithm for the Minimum Sum Coloring Problem," in *International Conference on Computational Collective Intelligence*, 2013, pp. 346-355.

[36] Y. Jin, J.-K. Hao, and J.-P. Hamiez, "A memetic algorithm for the minimum sum coloring problem," *Computers & Operations Research,* vol. 43, pp. 318-327, 2014.

[37] S. Mahmoudi and S. Lotfi, "Modified cuckoo optimization algorithm (MCOA) to solve graph coloring problem," *Applied soft computing,* vol. 33, pp. 48-64, 2015.

[38] S. M. Douiri and S. Elbernoussi" ,Solving the graph coloring problem via hybrid genetic algorithms," *Journal of King Saud University-Engineering Sciences,* vol. 27, pp. 114-118, 2015.

[39] R. Marappan and G. Sethumadhavan, "Solution to graph coloring problem using divide and conquer based genetic method," in *Information Communication and Embedd-ed Systems (ICICES), 2016 Internat-ional Conference on*, 2016, pp. 1-5.

[40] B. Ray, A. J. Pal, D. Bhattacharyya, and T. Kim, "An efficient ga with multipoint guided mutation for graph coloring problems," *International Journal of Signal Processing, Image Processing and Pattern Recognition,* vol. 3, pp. 51-58, 2010.

[41] S. Bakhtar, H. Jazayeriy, and M. Valinataj, "A multi-start path-relinking algorithm for the flexible job-shop scheduling problem," in *Information and Knowledge Technology (IKT),2015 7th Conference on*, 2015, pp. 1-6.

سید علی ساداتی تیلـه‌بنی مدرک کاردانی را در سال ۱۳۸۷ از دانشگاه فنی شهید چمران گرگان و در سال ۱۳۹۰ مدرک کارشناسی را از دانشگاه غیرانتفاعی قائم قائم‌شهر و مدرک کارشناسی ارشد خود را در سال ۱۳۹۳ از دانشگاه صنعتی نوشیروانی بابل در رشته مهندسی کامپیوتر دریافت کرد. زمینه‌های پژوهشی مورد علاقه ایشان الگوریتم‌های تکاملی، پنهان‌نگاری اطلاعات در تصویر و ویدئو و امنیت در شبکه‌های بین خودرویی است.

نشانی رایانامهٔ ایشان عبارت است از:

seyyedali.sadati@stu.nit.ac.ir

حمید جزایری مدرک کارشناسی خود را در رشته مهندسی‌رایـــانه- سخت افزار در سال ۱۳۷۵ از دانشگاه تهران دریافت کرد. ایشان سپس دوره‌های کارشناسی ارشد و دکترای خود را در رشته مهندسی‌رایـانـــه - نرم افزار از دانشگاه‌های اصفهان و پوترای مالزی در سال‌های ۱۳۷۹ و ۱۳۹۰ دریافت کرد. ایشان در حال حاضر استادیار دانشکده مهندسی برق و رایـــانـه دانشگاه صنعتی نوشیروانی بابل بوده و زمینه‌های پژوهشی ایشان عبارتند از: الگوریتم‌های تکاملی، تئوری گراف و سامانه‌های توزیع‌شده هوشمند.

نشانی رایانامهٔ ایشان عبارت است از:

jhamid@nit.ac.ir

مجتبی ولی‌نـتاج مدارک کارشناسی، کارشناسی ارشد و دکترای خود را در مهندسی رایـــانـه هر سه از دانشگاه تهران، به‌ترتیب در سال‌های ۱۳۷۹، ۱۳۸۱ و ۱۳۸۹ دریافت کرد. ایشان از سال ۱۳۸۹ به‌عنوان عضو هیأت علمی در دانشکده مهندسی برق و رایـــانـه دانشگاه صنعتی نوشیروانی بابل در بابل مشغول به فعالیت است. زمینه‌های پژوهشی اصلی مورد علاقه ایشان شامل طراحی سامانه‌های قابل‌اطمینان، شبکه‌روی‌تراشه، حساب رایـانـه‌ای ، معماری رایـــانـه و سامانه‌های چندپردازنده‌ای است.

نشانی رایانامهٔ ایشان عبارت است از:

m.valinataj@nit.ac.ir