جستجوی مدار هامیلتونی در گراف با روش های تکاملی

محمد رضا میبدی
mmeybodi@aut.ac.ir

کیوان اصغری
k.asghari@yahoo.com

چکیده: در این مقاله مسئله جستجوی مدار هامیلتونی در گراف را که یک مسئله NP-Complete می باشد، با استفاده از الگوریتمهای ژنتیکی، آتاماتای یادگیر و یک الگوریتم تکاملی جدید مورد بررسی قرار داده ایم. ابتدا به بررسی روشهای حریصانه و کاهشی برای حل این مسئله پرداخته و سپس الگوریتمهای ژنتیکی و ممتیکی و آتاماتای یادگیر و الگوریتم تکاملی جدید را برای حل این مسئله پیاده سازی کرده ایم. همچنین یک راهکار اکتشافی برای مقدار دهی الگوریتمهای تصادفی ارایه کرده ایم. در روش تکاملی جدید، هر کروموزوم از یک آتاماتای یادگیر تشکیل یافته است که در حین فرایند تکامل ژنتیکی، عمل یادگیری انجام داده و سعی در بهبود راه حل نهفته در خود دارد. نتایج بدست آمده از الگوریتم های مختلف را برای گراف های تصادفی دارای تعداد راسهای مختلف و با چگالی یالهای متفاوت، مورد مقایسه قرار داده و اقدام به تنظیم پارامترهای الگوریتم تکاملی جدید نموده ایم. نتایج بدست آمده، حاکی از برتری الگوریتم ژنتیکی و الگوریتم تکاملی جدید نسبت به دیگر الگوریتمها از لحاظ پیدا کردن مدار هامیلتونی در مدت زمان معقول می باشند.

۱- مقدمه

گراف ها و بویژه گراف های برچسب دار ابزار های قدرتمند و کارایی هستند که به طور گسترده در کاربردهای مختلفی مورد استفاده قرار می گیرند. روش های متعددی برای بررسی و تجزیه و تحلیل گراف ها نظیر پیدا کردن کوتاهترین مسیر، تشخیص مدار های هامیلتونی، رنگ آمیزی گراف و ... وجود دارد. در این مقاله مسئله جستجوی مدار هامیلتونی در گرافهای بدون جهت مورد بررسی قرار گرفته است. جوابهای بدست آمده برای این مسئله را می توان بعنوان راه حلهایی برای مسئله مشکل پیدا کردن طولانی ترین مسیر در گراف نیز بکار برد. حالت خاصی از این مسئله، بعنوان مسئله فروشنده دوره گرد مورد بررسی های بسیار قرار گرفته است. مسئله جستجوی مدار و مسیر هامیلتونی، یک مسئله تصمیم گیری می باشد که جواب مسئله بصورت وجود یا عدم وجود در گراف تعیین می شود. در مقالات[۴-۱] از الگوریتم ژنتیکی برای جستجوی مدار هامیلتونی در گرافهای چند سطحی، استفاده شده است. در مرجع [۲] عملگرهای جابجایی و جهش برای این الگوریتم ژنتیکی، مورد بحث قرار گرفته است. در مرجع [۵] نیز یک روش جستجوی اکتشافی برای جستجوی مدارهای هامیلتونی در گرافهای مکعبی پیشنهاد شده است. در [۶] روشهای اکتشافی برای جستجوی مدارهامیلتونی در گرافهای مشکل ارائه شده است. در [۷] یک روش الهام گرفته از کلونی مورچه ها برای تشخیص هامیلتونی بودن گراف با استفاده از پوشش تکراری راسهای گراف پیشنهاد شده است. در [۸] حداکثر تعداد مدارهای هامیلتونی در یک گراف هامیلتونی مورد بررسی قرار گرفته است. بیشتر کارهای موجود در مورد تشخیص گرافهای هامیلتونی، روی انواع خاصی از گرافها انجام شده است. با توجه به وجود مسئله فروشنده دوره گرد که یک مسئله بهینه سازی می باشد، مسئله مسیر هامیلتونی کمتر مورد توجه واقع شده است. اما با توجه به اینکه مقادیر ورودی مسئله مدار هامیلتونی با مسئله فروشنده دوره گرد کاملا متفاوت است، لذا ما این مسئله را با استفاده از روشهای تصادفی مختلف مورد بررسی قرار داده ایم. الگوریتم هایی که برای جستجوی مدار هامیلتونی در این مقاله ارایه شده اند، در مورد اکثر گرافها قابل استفاده هستند.

۲- تعریف مسئله

در زمینه ریاضی از تئوری گراف، یک مسیر هامیلتونی در یک گراف بدون جهت، مسیری است که از تمام راس های گراف عبور کرده و هر راس از گراف را دقیقا یکبار، ملاقات می کند. یک مدار یا حلقه هامیلتونی در یک گراف بدون جهت، هر راس را دقیقا یک بار ملاقات کرده و نهایتا به گره شروع بر می گردد. به گرافی که دارای یک مدار هامیلتونی باشد، گراف هامیلتونی گفته می شود. مسئله پیدا کردن مدار و مسیر هامیلتونی در گرافها یکی از مسائل NP-Complete است[۹] که تاکنون توجه کمی در مورد حل آن وجود داشته است. اکثر محققانی که در مورد مسائل اکتشاف گراف کار کرده اند، توجه خود را بیشتر به مسئله فروشنده دوره گرد که حالت خاصی از مسئله پیدا کردن مدار هامیلتونی در گراف است، معطوف ساخته اند. اما با توجه به اینکه ورودیهای این مسئله برای حل آن با مسئله فروشنده دوره گرد تفاوت دارد، خود این مسئله می تواند دارای کاربردهای دیگری در مسائل مسیریابی شبکه های ارتباطی و ... باشد. از سوی دیگر، مسئله پیدا کردن مسیر هامیلتونی حالت خاصی از مسئله مشکل طولانی ترین مسیر در گراف برای حالتی است که مسیری دارای n-1 یال بین دو راس از گراف جستجو می شود[۱۰].

در بیشتر مسایل استاندارد موجود برای مسئله فروشنده دوره گرد، داده های مسئله عبارت از لیست مختصات راس های گراف و در مواردی ماتریس مجاورت گراف می باشد. برای مسئله مدار هامیلتونی، داده های مسئله عبارت از لیست یالهای گراف و در مواردی ماتریس مجاورت گراف است اما با این تفاوت که درایه های ماتریس، تنها یکی از اعداد ۰ یا ۱ است. در مسئله فروشنده دوره گرد در اکثر موارد، گراف مسئله کامل در نظر گرفته می شود. یعنی در یک سری از مسائل استاندارد که لیست مختصات رئوس داده می شود، پیشفرض مسئله کامل بودن گراف و وجود یالی بین هر دو

راس دلخواه از گراف می باشد. در مسائلی نیز که ورودی مسئله، ماتریس مجاورت گراف است، تمام درایه های ماتریس موجود بوده و هر کدام نشان دهنده وزن یال بین دو راس مربوطه است. در هیچ کدام از این مسائل هدف کاوش گراف برای پیدا کردن یک مسیر نیست بلکه هدف، کمینه کردن طول تور فرضی موجود است. اما در مسئله پیدا کردن مدار هامیلتونی، بیشتر توجه مان روی کاوش گراف و پیدا کردن مداری خواهد بود که کل راس های گراف را بهم وصل کند[۱۱]. مسئله پیدا کردن مدار هامیلتونی در یک گراف یک مسئله تصمیم گیری است بر خلاف مسئله فروشنده دوره گرد که با توجه به خروجی مسئله بصورت طول تور بهینه، یک مسئله بهینه سازی است. روشی که ما در این بررسی در پیش گرفته ایم، حل مسئله پیدا کردن مدار هامیلتونی با درصدی از خطا است. درصد خطا نیز به این صورت تعیین می شود که خروجی الگوریتمهای ما دنباله ای از رئوس گراف است که مشخص کننده ترتیب ملاقات راسهای گراف در مدار هامیلتونی می باشند اما اگر بین دو راس از راسهای موجود در دنباله یالی در گراف موجود نباشد بعنوان موردی از خطا محسوب می شود. در نهایت تعداد یالهای موجود بین راسهای ارایه شده بر تعداد کل رئوس که مشخص کننده تعداد یالهای مدار هامیلتونی واقعی است از مقدار یک کسر شده و درصد ضرب می شود تا درصد خطا را نشان دهد. یعنی در حقیقت مسئله مسیر هامیلتونی را با در نظر گرفتن تعریفی از خطا در پیدا کردن آن بصورت یک مسئله بهینه سازی حل نموده ایم[۱۵-۱۲].

۳-الگوریتم حریصانه

یکی از ساده ترین روشهای جستجوی آگاهانه بهترین جواب که در اینجا پیدا کردن مدار هامیلتونی است، روش حریصانه می باشد. در این روش در هر مرحله، راسی که دارای بهترین معیار حریصانه، در حالت فعلی است، انتخاب می شود. برای الگوریتم حریصانه پیدا کردن مدار هامیلتونی در گراف از یک روش اکتشافی استفاده می کنیم که سعی در پیدا کردن طولانی ترین مسیر ها با هدف پیدا کردن مدار هامیلتونی در گراف دارد.

۳-۱- روش اکتشافی[1] برای پیدا کردن طولانی ترین مسیرها در گراف

با فرض داشتن گرافی که بین تعدادی از راس های آن یالی وجود داشته و بین تعدادی نیز یالی وجود نداشته باشد، می خواهیم از راسی شروع کرده و تا زمانی که مسیری وجود داشته باشد راس های گراف را طی کنیم. روش اکتشافی پیشنهادی به این صورت است که ابتدا یک راس تصادفی را انتخاب می کند. سپس یالی را در گراف جستجو می کند که این راس در آن یال شرکت داشته باشد. سپس راس دیگر یال پیدا شده را در لیست یالها جستجو می کند تا یال دیگری متصل به یال قبلی پیدا کند. و این روال ادامه می یابد تا زمانی که یالی برای راس مورد جستجو پیدا نشود. در اینصورت یک راس تصادفی دیگر پیدا شده و کارهای قبلی روی آن انجام می گیرد. بنابراین روال مذکور شبیه یک جستجوی عمقی راس ها در گراف می باشد که حاصل کار عبارت از کل راس های گراف است که توسط یک یا چند مسیر از یالها بهم متصل شده اند[۱۶،۱۷]. شبه کد مربوط به این روش اکتشافی در شکل ۱ آمده است.

```
Procedure DFS_Initial()
Begin
    node1 = Random Number Between 1 to Number of nodes in Graph
    Initial_Cycle(1) = node1
    For i = 2 To n
      find_flag = False
      For j = 1 To Edge_Count
        If There is a Edge in Graph that Contains node1 in one side And
           The other side of This Edge is not Exist in Initial_Cycle Then
           Node2= Node in the other side of this Edge
           Initial_Cycle(i) = node2
           node1 = node2
           find_flag = True
           Exit For j
        End If
      Next j
      If Not find_flag Then
        node2 = Random Number Between 1 to Graph nodes
        number that not Exist in Initial Cycle
        Initial_Cycle(i) = node2
        node1 = node2
      End If
    Next i
End Procedure
```

شکل ۱- شبه کد روش اکتشافی برای پیدا کردن طولانی ترین مسیرها در گراف

۴- روشهای کاهشی برای پیدا کردن مدار هامیلتونی

این روشها، راهکارهای معاوضه دوبدو (جفت جفت) شامل DES و DESO هستند. آنها با دنباله هایی بدست آمده توسط اعمال یک هیوریستیک بخش ۳-۱ شروع می کنند و راس های (u,v) از (۲, ۱) به (۳, ۱) , (۴, ۱), ... , و (n, n-1) را بصورت پی در پی برای بهبود جواب اولیه با یکدیگر تعویض می کنند که (u,v) به معنی تعویض راس موجود در موقعیت u ام با راس موجود در موقعیت v ام می باشد. روش DES، روش کاهشی سخت گیری است که در آن، تنها معاوضه راس های دوبدو مورد قبول واقع می شوند که منجر به کاهش در تابع هدف شود که تابع هدف در اینجا افزایش تعداد یالهای حلقه هامیلتونی و در نتیجه کاهش درصد خطا است. در نتیجه اگر جواب مسئله بوجود آمده از تعویض اخیر موجب کاهش در تابع هدف نشود، تعویض برگردانده می شود. اما در روش DESO (روش کاهشی با تعویض صفر)، تعویض راس های دوبدو که منجر به تغییر در تابع هدف نشود نیز بعلاوه تعویضاتی که باعث کاهش در تابع هدف می شود، مورد پذیرش واقع می شود. پذیرش تعویضی بدون تغییر در تابع هدف بدین دلیل است که ممکن است نتیجه تعویض موجب خارج شدن یالهای قبلی از حلقه و وارد شدن دوباره یالهای جدید شود که این عمل ممکن است در تکرار های بعدی منجر به کاهش تابع هدف گردد.

۵- الگوریتم ممتیکی

الگوریتم ممتیکی ترکیبی میان جستجوی سراسری مبتنی بر جمعیت و جستجوی محلی اکتشافی است که روی هر یک از افراد صورت می گیرد. هر فرد نشان دهنده یک راه حل ممکن است و جمعیت مذکور توسط عملگرهای جابجایی و جهش همراه با بهبود محلی با استفاده از هیوریستیکها رشد می کند. در اکثر مقالات چنین گفته می شود که الگوریتم ممتیکی برگرفته از اصول تکامل

[1] Heuristic

فرهنگی در جوامع انسانی است. بدین صورت که هر راه حل بعنوان یک عقیده یا نظر فرض می شود. هر عقیده ابتدا توسط شخص دارنده آن عقیده که در الگوریتم ما کروموزوم مورد نظر است مورد بررسی قرار گرفته و در صورت امکان بهبود می یابد و به اشخاص یا کروموزومهای دیگر منتقل می گردد.الگوریتم ممتیکی پیشنهاد شده در این مقاله ، دنباله ای بدست آمده از الگوریتم حریصانه (جستجوی عمقی) و دنباله‌ای تولید شده بصورت تصادفی را بعنوان والدین اولیه بکار می برد. روش نمایش راه حل در الگوریتم ممتیکی به این صورت است که دنباله ای از راس ها بصورتی که در یک مدار هامیلتونی ملاقات خواهند شد، بصورت یک کروموزوم نمونه نمایش داده می شوند. یکی از عملگرهای جابجایی پیاده سازی شده برای الگوریتم ممتیکی، مشهور به جابجایی ترتیبی (OX) است که توسط گلد برگ توصیف شده است [۱۸،۱۹]. همچنین دو عملگر جابجایی دیگر Cycle Crossover و Partially Mapped Crossover که برای کار با جایگشت ها مناسب می باشند، در پیاده سازی الگوریتم ممتیکی و الگوریتم ژنتیکی و الگوریتم تکاملی جدید، بکار رفته اند. یک روش جابجایی جدید نیز که الهام گرفته از روش جابجایی ترتیبی (OX) است در تمام الگوریتمها پیاده سازی شده است که Reverse Ordered Crossover نام گرفته است که به توضیح آنها خواهیم پرداخت. فرزند حاصل از عملگر جابجایی، دنباله (σi) را نشان می دهد و مقدار برازندگی هر کروموزوم با استفاده از شمارش تعداد یالهای موجود در حلقه هامیلتونی محاسبه می شود. استراتژی جهش پیاده سازی شده برای الگوریتم ممتیکی، مبتنی بر تعویض راسها می باشد. در این عملگر، دو موقعیت بطور تصادفی انتخاب می شوند و ژنهای موجود در این محلها مقادیرشان را با هم عوض می کنند. این عمل منجر به دنباله جدیدی بنام σ_j و مقدار تابع برازندگی برای این دنباله $Z(\sigma_j)$ می شود. سپس تکنیک بهبود محلی در این مکان بکار گرفته می شود. اگر $Z(\sigma_i)<Z(\sigma_j)$ دنباله بهبود یافته σ_j به مجموعه جمعیت اضافه می شود. اگر شرط برقرار نشود، روال جهش تا زمانی تکرار می شود که یک بهبود محلی حاصل شود. از آنجاییکه تنها جفتی از والدین بکار رفته اند، احتمال جابجایی و جهش، مقدار ۱ فرض شده اند. بنابراین فرزند بطور محلی بهبود داده شده و با بدترین والد برای سیر تکاملی، جایگزین می گردد. تعداد نسلها بعنوان معیار پایان کار عمل می کند و بهترین راه حل بدست می آید.

۶- الگوریتم ژنتیکی

الگوریتمهای ژنتیکی اساسا توسط جان هلند پیشنهاد شدند[۲۰]. این الگوریتمها، روشهای جستجویی هستند که یک فضای جستجو را کاوش کرده و فرایند تکامل بیولوژیکی را تقلید می کنند. الگوریتم های ژنتیکی بر روی جمعیتی از راه حلهای بالقوه یا کروموزوم ها که هر یک می‌توانند بعنوان پاسخی از مسئله تلقی شوند، با اعمال عملگرهای ژنتیکی به جستجوی راه حل نهایی می پردازند. برای مسئله پیدا کردن مدار هامیلتونی در گراف، هر کروموزوم با نمایش جایگشت طبیعی راه حل، جایگشتی از اعداد صحیح ۱, ... ,n است که ترتیب ملاقات n راس را روی مدار هامیلتونی تعریف می کند. بمنظور تخمین یک راه حل بهینه بصورت دقیقتر، جمعیت اولیه کروموزومها توسط روش اکتشافی[۲] گفته شده در بخش ۳-۱در ترکیب با روش تصادفی که کروموزومها را بصورت تصادفی تولید می کند، ایجاد می شود. از این طریق فضای جستجو کاهش می یابد زیرا با ساده سازی فضای راه حل، مسئله به سمت یک راه حل سریع بهینه راهنمایی می شود.

هنگامی که یک جمعیت ایجاد شد، هر کرموزوم ارزیابی می شود و برازندگی آن بصورت زیر مورد محاسبه قرار می گیرد. تعداد یالهای موجود در مدار هامیلتونی و در نتیجه در صد خطا برای هر کروموزوم محاسبه می شود. تعداد یالهای موجود در هر مدار مشخص شده توسط هر کروموزوم بعنوان میزان برازندگی هر کروموزوم مقدار دهی می شود. بنابراین هر کروموزومی که راس های موجود در آن تعداد یال بیشتری بینشان داشته باشند، دارای برازندگی بیشتری خواهد بود و حد اکثر میزان برازندگی برابر با تعداد راس های گراف و حداقل آن برابر با صفر است. با استفاده از روشهای انتخاب، کروموزومها (والدین) از جمعیت انتخاب می شوند تا با هم ترکیب شده و کروموزومهای جدید (فرزندان) را برای بکار بردن عملگرهای ژنتیکی تولید کنند. در پیاده سازی ما عملگرهای انتخاب چرخ رولت و رتبه بندی بکار رفته اند. نقش یک عملگر جابجایی، ترکیب عناصر دو کروموزوم والد برای تولید یک یا چند کروموزوم فرزند است. در این عملگر دو جایگشت والد انتخاب، و سپس طبق یکی از روشهای جابجایی، عمل جابجایی روی والدین انجام می گیرد. با این عمل دو جایگشت جدید حاصل می شود که اصطلاحا فرزندان دو جایگشت والد خوانده می شوند. روش های جابجایی گفته شده در الگوریتم ممتیکی برای الگوریتم ژنتیک نیز پیاده سازی شده اند. عملگرهای جابجایی برای N/2 جفت از کروموزومهای انتخاب شده بصورت تصادفی بکار می روند که N اندازه جمعیت است. نقش یک عملگر جهش نیز فراهم کردن و نگهداری تنوع و گوناگونی در یک جمعیت به گونه ای است که دیگر عملگرها بتوانند به کارشان ادامه دهند. چهار عملگر جهش Swap، Insertion، Inversion و Scramble که برای کار با جایگشت ها مناسب می باشند، برای حل مسئله با الگوریتم ژنتیک معمولی و الگوریتم تکاملی جدید، پیاده سازی شده اند که در توضیح الگوریتم تکاملی جدید به آنها خواهیم پرداخت. عملگر جایگزینی بکاررفته در الگوریتم ژنتیکی پیاده سازی شده برای این مسئله، مبتنی بر نخبه سالاری است[۱۸]. این عمل نگهداری بهترین کروموزومهای جمعیت فعلی و فرزندان آنهاست. این بهترین کروموزومها از جمعیت جدید برای نسل بعدی زنده نگه داشته می شوند.

۷- آتاماتای یادگیر

روش آتاماتای یادگیر در یادگیری عبارت است از تعیین اقدام بهینه از مجموعه محدودی از اقدامهای از پیش تعریف شده که قابل

[2] heuristic dispatching rules

انجام در یک محیط تصادفی[3] می باشند. یک آتاماتا در یک مدار بازخوردی[4] با محیط تصادفی ناشناس، تراکنش انجام می دهد. آتاماتا در هر لحظه اقدامی را از مجموعه اقدامهای خود انتخاب و به محیط اعلام می نماید. محیط در پاسخ به اقدام انجام شده، یک خروجی از مجموعه خروجی های تعریف شده را تولید و به آتاماتا اعلام می نماید و آتاماتا با دریافت پاسخ محیط، شیوه تصمیم گیری خود را در انتخاب اقدام بعدی به هنگام می نماید. فرض می شود که بین هر اقدام آتاماتا و پاسخ محیط یک رابطه احتمالی وجود داشته باشد و این رابطه که در واقع مشخصات داخلی محیط است، در طول یادگیری توسط آتاماتا شناخته می شود [۲۱]. برای یک مسئله کاوش گراف جهت پیدا کردن مدار هامیلتونی در گراف دارای n راس، !n جایگشت مختلف وجود دارد و در صورتی که از آتاماتاهای یادگیر برای حل مساله مدار هامیلتونی استفاده شود، آتاماتای یادگیر باید !n اقدام داشته باشد که تعداد زیاد اقدام ها باعث کاهش سرعت همگرایی می شود و به همین منظور از آتاماتای یادگیر مهاجرت اشیاء[5] که توسط اومن[6] و ما[7] پیشنهاد شده است، استفاده می شود[۲۲].

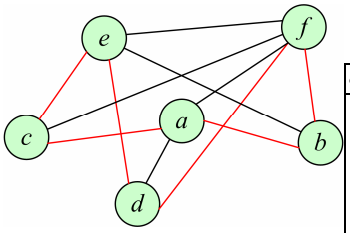
۸- الگوریتم تکاملی جدید

در الگوریتم تکاملی جدید برای حل مسئله جستجوی مدار هامیلتونی در گراف، هر راه حل یا کروموزوم با یک آتاماتای یادگیر نشان داده می شود. بنابراین هر یک از اعضای جمعیت در سیر تکاملی الگوریتم ژنتیکی دارای قابلیت یادگیری خواهند بود و با توجه به این قابلیت می توانند راه حل نهفته در ساختار خود را بطور مستقل از سایر اعضای جمعیت، بهبود بخشند. بعلاوه قابلیت یادگیری، مانع از بدام افتادن الگوریتم در حداقل های محلی شده و سرعت رسیدن به جواب نیز افزایش پیدا می کند. در مراجع [۲۶-۲۳] روشهای مشابهی برای حل مسایل فروشنده دوره گرد، بهینه سازی پرس و جو های پایگاه داده ای و تناظر گراف، ارایه شده است. در ادامه مطالب بطور خلاصه نحوه نگاشت مسئله را به آتاماتای مهاجرت اشیاء که تشکیل دهنده یک کروموزوم در الگوریتم تکاملی جدید است، مطرح می کنیم. در مقایسه نتایج الگوریتمها از GALA برای نشان دادن نتایج حاصل از الگوریتم تکاملی جدید استفاده شده است.

۸-۱- آتاماتای یادگیر بعنوان کروموزوم

در الگوریتم تکاملی جدید، هر کروموزوم توسط یک آتاماتای یادگیر از نوع مهاجرت اشیاء نشان داده می شود بطوریکه محتوای هر کدام از ژن های کروموزوم بصورت یک شیء ، به یکی از اقدام های آتاماتا نسبت داده می شود و در عمق مشخصی از آن اقدام قرار می گیرد. در این روش، شیء هر اقدام از آتاماتا نشان دهنده یک راس گراف می باشد. در این آتاماتا $\underline{\alpha}=\{\alpha_1,...,\alpha_k\}$ مجموعه اقدام های مجاز برای آتاماتای یادگیر است. این آتاماتا k اقدام دارد (تعداد اقدام های این آتاماتا برابر با تعداد راس های گراف است که همه آنها باید در یک مدار هامیلتونی ملاقات شوند). اگر راس s از گراف، در اقدام m قرار گرفته باشد، در اینصورت راس s در ترتیب ملاقات راس ها، m امین راسی خواهد بود که ملاقات می شود. $\underline{\phi}=\{\phi_1, \phi_2, ... , \phi_{KN}\}$ مجموعه وضعیت ها و N عمق حافظه برای آتاماتا می باشد. مجموعه وضعیت های این آتاماتا به K زیر مجموعه $\{\phi_1, \phi_2, ... , \phi_N\}$ و $\{\phi_{N+1}, \phi_{N+2}, ... , \phi_{2N}\}$ و ... و $\{\phi_{(K-1)N+1}, \phi_{(K-1)N+2}, ... , \phi_{KN}\}$ افراز می شود و راس ها بر اساس این که در کدام وضعیت قرار داشته باشند دسته بندی می گردند. اگر راس s از مجموعه راس ها در مجموعه وضعیت های $\{\phi_{(j-1)N+1}, \phi_{(j-1)N+2}, ... , \phi_{jN}\}$ قرار داشته باشد در اینصورت راس s در ترتیب ملاقات راس ها، j امین راسی خواهد بودکه ملاقات می شود. در مجموعه وضعیت های اقدام j، به وضعیت $\phi_{(j-1)N+1}$ وضعیت داخلی و به وضعیت ϕ_{jN}، وضعیت مرزی گفته می شود. راسی که در وضعیت $\phi_{(j-1)N+1}$ قرار دارد راس با اهمیت بیشتر و راس در وضعیت ϕ_{jN} راس با اهمیت کمتر نامیده می شود.

در اثر پاداش دادن یا جریمه کردن یک اقدام، وضعیت راس وابسته به آن اقدام تغییر می کند. اگر راسی در وضعیت مرزی یک اقدام قرار داشته باشد، جریمه شدن اقدام آن باعث تغییر اقدامی که راس به آن وابسته است، می شود و در نتیجه باعث ایجاد جایگشت جدیدی می گردد. حال بعنوان مثال، گراف بدون جهت شکل ۲ را که دارای ۶ راس و تعدادی یال (جدول۱) می باشد، در نظر بگیرید. این گراف دارای چند مسیر هامیلتونی می باشد که یکی از آنها با یالهای قرمز رنگ نشان داده شده است.



شکل ۲- گراف هامیلتونی دارای ۶ راس

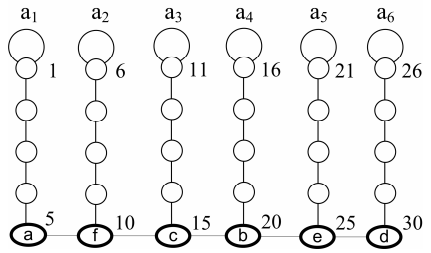
جدول ۱- لیست راس ها و یال های تشکیل دهنده گراف

لیست راس های گراف	لیست یال های گراف
a	a-b , a-f
b	a-d , a-c
c	c-e , c-f
d	e-d , e-f
e	e-b , b-f
f	d-f

جایگشت (a,f,c,b,e,d) را بعنوان یک جایگشت اولیه برای ترتیب ملاقات راس ها در مدار هامیلتونی در نظر بگیرید. نحوه نمایش این جایگشت با آتاماتای مهاجرت اشیای مبتنی بر اتصالات آتاماتای ستلین، بصورت شکل ۳ است. هر آتاماتا دارای ۶ اقدام $\{\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_5,\alpha_6\}$ (به تعداد راس ها) و عمق ۵ می باشد. مجموعه وضعیت های {1,6,11,16,21,26} وضعیت های داخلی و مجموعه وضعیت های {5,10,15,20,25,30} وضعیت های مرزی آتاماتا هستند. در ابتدا هر یک از راس ها در وضعیت مرزی اقدام مربوطه قرار دارند. در الگوریتم تکاملی جدید ، هر ژن از کروموزوم

[3] Random Enviroment
[4] Feedback
[5] Object Migrating Automata (OMA)
[6] Oommen
[7] Ma

معادل یک اقدام آتاماتا می باشد و لذا می توان در ادامه این دو واژه را به جای یکدیگر بکار برد.



شکل ۳- نمایش جایگشت (a,f,c,b,e,d) بوسیله آتاماتای یادگیر مهاجرت اشیای مبتنی بر اتصالات آتاماتای ستلین

۸-۲- جمعیت اولیه

با فرض اینکه تعداد اعضای جمعیت p باشد، p-1 عضو جمیعت با ایجاد p-1 جایگشت تصادفی تولید می شوند. برای تولید آخرین عضو جمعیت، از روش اکتشافی گفته شده در قسمت ۳-۱ استفاده می کنیم. به این جایگشت، جایگشت تقریبی[8] می گوییم. آخرین عضو اضافه شده به جمعیت بیشترین تشابه را با جواب نهایی دارد. به عنوان مثال نحوه تشکیل جمعیت اولیه برای گراف شکل ۲ با فرض p=6 برای تعداد اعضای جمعیت در ادامه توضیح داده شده است. پنج عضو اول جمعیت به وسیله پنج جایگشت تصادفی (f,c,a,d,b,e)، (e,b,d,f,a,c) ، (b,c,e,f,d,a) ، (d,e,a,c,b,f)، (f,e,c,d,b,a) تولید می شود. برای ایجاد جایگشت ششم از روش اکتشافی گفته شده استفاده می کنیم. حاصل استفاده از این روش جایگشت (b,e,d,c,f,a) است. جمعیت اولیه تولید شده برای مسئله مطرح شده بصورت شکل ۴ می باشد. در ابتدا هر راس در وضعیت مرزی اقدام خود قرار دارد.

۸-۳- تابع برازندگی

در الگوریتم های ژنتیک تابع برازندگی، شاخص زنده ماندن کروموزوم ها است. در جستجوی مدار هامیلتونی در گراف، هدف یافتن جایگشتی از راس های گراف مثل σ است که تعداد یالهای دیده شده در جایگشت راس های ملاقات شده در گراف بیشینه باشد. لذا تابع برازندگی $f(\sigma)$ در مساله مدار هامیلتونی در گراف به صورت زیر تعریف می شود.

تعداد یالهای موجود در ترتیب فعلی ملاقات راس های (σ) =f(σ)

۸-۴- عملگرهای الگوریتم تکاملی جدید

از آنجا که در الگوریتم تکاملی جدید، هر کروموزوم به صورت یک آتاماتای یادگیر نمایش داده می شود، عملگرهای جابجایی و جهش مشابه عملگرهای سنتی جابجایی ژنتیک نیستند.

الف) عملگر انتخاب: برای انتخاب آتاماتاهای یادگیر (کروموزوم) بعنوان والدین جهت انجام عملگرهای جهش و یا جابجایی دو روش انتخاب چرخ رولت و رتبه بندی پیاده سازی شده است.

ب) عملگر ترکیب یا جابجایی: برای الگوریتم تکاملی جدید، چهار عملگر جابجایی Ordered، Cycle، Partially Mapped و Reverse Ordered که برای کار با جایگشت ها مناسب می باشند، پیاده سازی شده است که به توضیح مختصر آنها می پردازیم.

Ordered : این روش جابجایی، ترتیب نسبی ژنها در کروموزوم را تا حد ممکن حفظ می کند. با در نظر گرفتن جایگشت های والد اول و دوم, دو جایگشت فرزند به وسیله انتخاب دو نقطه برش در جایگشت های والدین و کپی کردن المان های بین دو نقطه برش در فرزندان و پر کردن باقیمانده جایگشت های فرزندان با المان های استفاده نشده از والد دیگر با شروع از نقطه برش دوم به بعد، ایجاد می شوند.

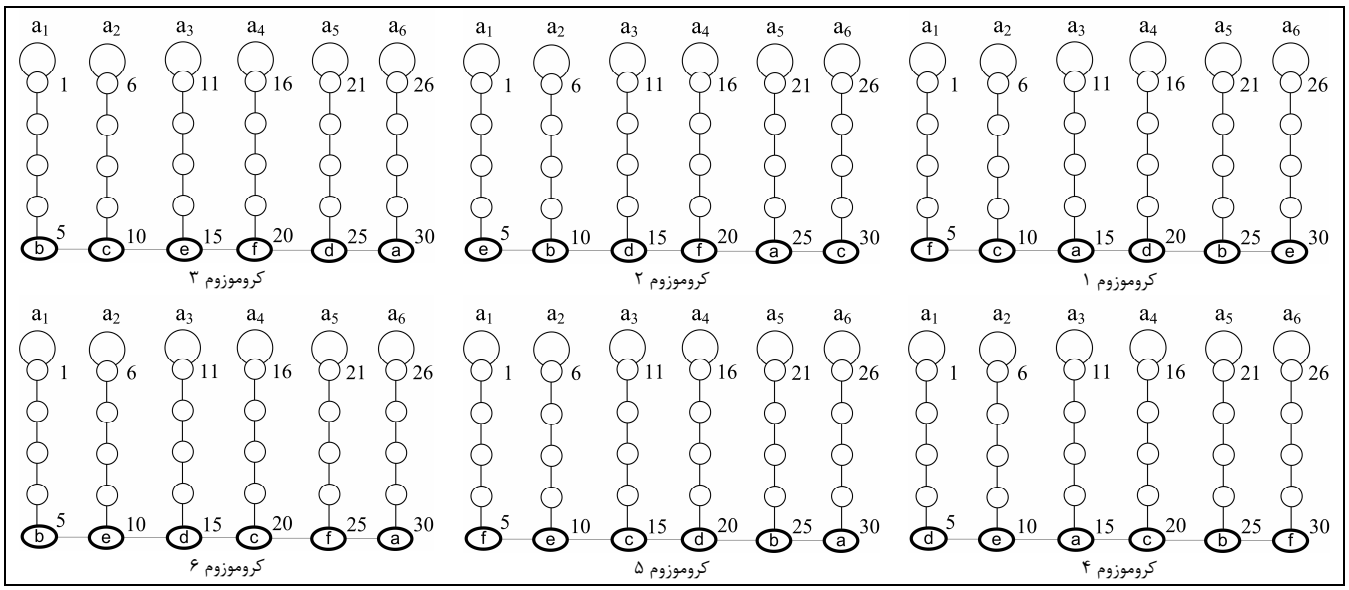
Reverse Ordered: این نوع عملگر جابجایی پیشنهاد شده شبیه روش Ordered است با این تفاوت که در این روش ژنهای خارج از دو نقطه برش در فرزندان کپی می شوند بر خلاف روش Ordered که ژنهای بین دو نقطه برش در فرزندان کپی می شوند.

Partially Mapped: در این نوع عملگر جابجایی، دو جایگشت فرزند به وسیله انتخاب دو نقطه برش در جایگشت های والدین و کپی کردن المان های بین دو نقطه برش در فرزندان و پر کردن باقیمانده جایگشت های فرزندان با المان های متناظر آنها از والدین، ایجاد می شوند. توجه کنید که اگر یکی از المانهای خارج از ناحیه جابجایی فرزندی با یکی از المان های ناحیه جابجایی خود یکسان باشد آن المان به عنوان حفره در نظر گرفته می شود و باید مقدار آن عوض شود. نحوه پر کردن یک حفره به این صورت است که مثلا اگر حفره در فرزند اول باشد، ابتدا موقعیت ژنی از والد دوم را که دارای مقدار مساوی با مقدار حفره است پیدا می کنیم. سپس از والد اول مقدار ژنی را که در موقعیت یکسانی با ژن پیدا شده قرار دارد، به عنوان مقدار حفره در نظر می گیریم.

Cycle: در این عملگر جابجایی، دو جایگشت فرزند بوسیله شکل دادن یک سیکل در بین والدین ایجاد می شوند. به عنوان مثال برای ایجاد فرزند اول، با شروع از اولین المان والد ۱، آن را به اولین ژن در والد دوم نگاشت می کنیم. سپس اولین ژن والد دوم را در فرزند اول پیدا کرده و به ژن هم مکان خود در والد دوم نگاشت می دهیم و این کار تا کامل شدن سیکل کامل ادامه می یابد. المان هایی از سیکل را که در والد ۱ هستند در فرزند اول قرار می دهیم. سپس مکان های خالی فرزند اول را با المان های متناظر در والد ۲ پر می کنیم. فرزند دوم نیز به همین ترتیب ایجاد می شود.

همچنین در این بخش دو عملگر جابجایی Exchange و Smart Exchange را نیز معرفی می کنیم که عملکرد بهتری در الگوریتم تکاملی جدید دارند. در عملگر جابجایی Smart Exchange که در شکل ۵ شبه کد آن آمده است، دو کروموزوم والد انتخاب می شوند و به صورت تصادفی دو ژن i و j در یکی از دو کروموزوم والد انتخاب می شوند. سپس همین دو ژن در کروموزوم

[8] Approximate Mapping



شکل ۴- جمعیت اولیه برای مسئله جستجوی مدار هامیلتونی در گراف بدون جهت دارای ۶ راس

دیگر انتخاب می شوند. مجموعه ژنهای با شماره های بین i و j را مجموعه جابجایی[9] می نامیم. سپس ژن های هم شماره در دو مجموعه جابجایی با یکدیگر جابجا می شوند (مثلا ژن شماره i از مجموعه جابجایی اول با ژن شماره i از مجموعه جابجایی دوم، ژن شماره i+1 از مجموعه جابجایی اول با ژن شماره i+1 از مجموعه جابجایی دوم و ... جابجا می شوند). با این عمل دو کروموزوم جدید حاصل می شوند که اصطلاحاً فرزندان دو آتاماتای والد خوانده می شوند. از آنجا که در این الگوریتم از n کروموزوم (آتاماتا) استفاده می‌شود و هر آتاماتا دارای مشخصه‌های اختصاصی مربوط به خود (وضعیت، اقدام و شیء متناظر هر اقدام) می‌باشند، جهت خوانایی بیشتر شبه کد، این مشخصه ها را با پیشوند نام آتاماتا و جداساز نقطه نشان می دهیم. مثلا برای نشان دادن وضعیت راس روی اقدام u از آتاماتای i از نمایش LAi.Action(u).State استفاده شده است.

```
Procedure Crossover ( LA1, LA2 )
Begin
  Generate two random numbers r1 and r2 between 1 to n=|V_G|
  r1 = Random *n;
  r2 = Random *n;
  r1 = Min (r1, r2);
  r2 = Max (r1, r2);
  For i = r1 to r2 do
    If (Edge_Count_i( LA1) > Edge_Count_i( LA2) ) Then
      j = Action of LA2 Where
      LA2.Action (j).Object == LA1.Action (i).Object;
      Swap (LA2.Action (i).Object, LA2.Action (j).Object) ;
    Else
      j = Action of LA1 Where
      LA1.Action ( j ) .Object == LA2.Action( i ).Object;
      Swap (LA1.Action (i).Object, LA1.Action (j).Object);
    End if
  End For
End Procedure
// Edge_Count_i(LA_k) = Number of Edges that connect Vertex in Action i
to the Vertices in Action i-1 and Action i+1 in Learning Automata
Number k
```

شکل ۵- شبه کد عملگر جابجایی Smart Exchange Crossover

عملگر جابجایی Exchange Crossover نیز مانند عملگر Smart Exchange Crossover است با این تفاوت که شرط If(Edge_Counti(LA1)>Edge_Counti(LA2)) در عملگر جابجایی Exchange وجود ندارد. در این شرط راس های متعلق به اقدام های i ام از دو آتاماتای LA1 و LA2 با هم مقایسه می شوند که مشخص شود کدامیک از آنها با ملاقات شدن درترتیب و اقدام i ام، باعث افزایش تعداد یالهای حلقه هامیلتونی مورد نظر می شوند.

در شکل ۷ مثالی از نحوه عملکرد این عملگرها نشان داده شده است. به عنوان مثال دو آتاماتای LA_1 و LA_6 از جمعیت تشکیل شده قبل به صورت تصادفی انتخاب می شوند. سپس با انتخاب تصادفی دو محل a_2 وa_3 مجموعه جابجایی $\{\alpha_2,\alpha_3\}$ حاصل می شود و با جابجایی راس های موجود روی اقدام های متناظر در فاصله جابجایی، دو کروموزوم جدید حاصل می شود.

ج) عملگر جهش: برای انجام دادن این عملگر می توان از روشهای مختلف که برای کار با جایگشت ها مناسب هستند استفاده کرد زیرا جایگشت تولید شده توسط آنها همیشه معتبر می باشند. چهار عملگر جهشSwap،Insertion،Inversionو Scramble برای الگوریتم تکاملی جدید مورد استفاده واقع شده اند.

Order Based (Swap): در این روش، دو اقدام (ژن) از یک آتاماتا (کروموزوم) به صورت تصادفی انتخاب شده و راس های متعلق به این اقدامها با یکدیگر جابجا می شوند. شکل های ۶ و ۸ شبه کد و مثالی از این عملگر را نشان می دهد.

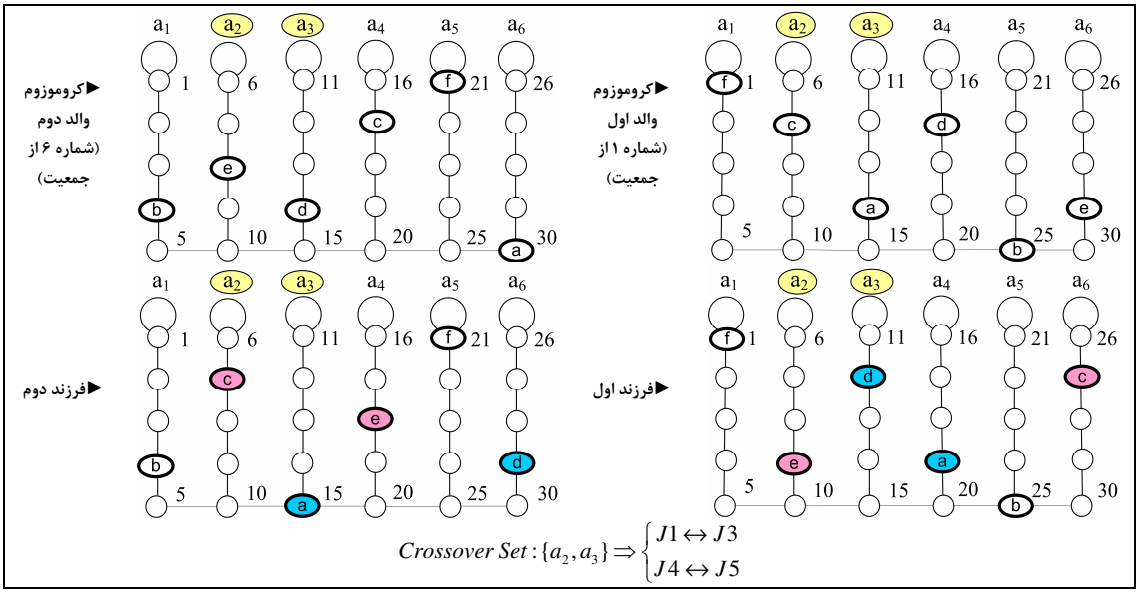
SubList Based: دو ژن از کروموزوم را بطور تصادفی انتخاب کرده و ترتیب ژنها در بین این دو ژن را معکوس می کنیم.

Insertion: یک ژن یا یک بلوک از ژنها را در کروموزوم بطور تصادفی انتخاب کرده و آنها را در یک نقطه تصادفی دیگر درج می کنیم و **Scramble**: یک بلوک از ژنها را بطور تصادفی انتخاب کرده و آنها را بصورت تصادفی دوباره مرتب می کنیم.

```
Procedure Mutation ( LA )
Begin
    i = Random *n;
    j = Random *n;
    Swap( LA.Action( i ).Object , LA.Action( j ).Object );
End Procedure
```

شکل ۶- شبه کد عملگر جهش

[9] Crossover Set

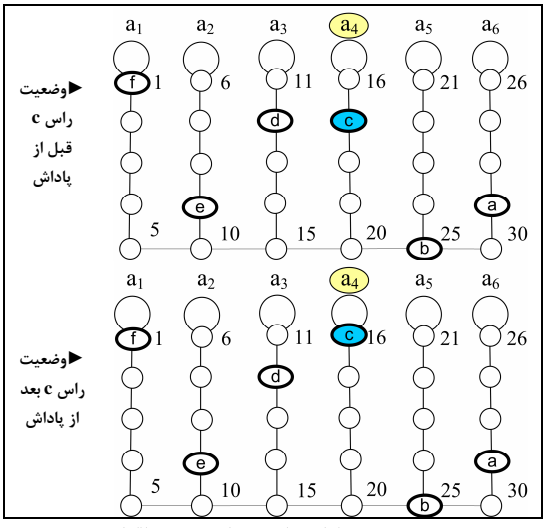


شکل ۷- نحوه انجام عملگر جابجایی Smart Exchange Crossover

```
Procedure Reward( LA, u )
Begin
    If (LA.Action(u).State-1) mod N <> 0 then
        Dec (LA.Action(u).State);
    End if
End Procedure
```

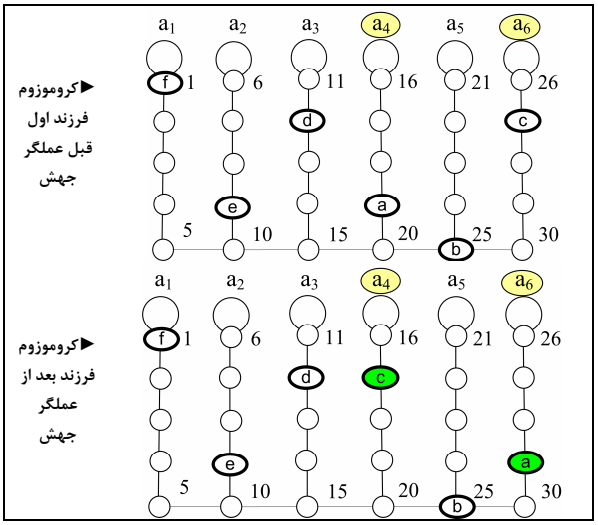
شکل ۹- عملگر پاداش اقدام u از آتاماتای LA با اتصالات مشابه آتاماتای ستلین

به عنوان مثال با توجه به شکل ۱۰ در آتاماتای با اتصالات مشابه آتاماتای ستلین اگر راس c در مجموعه یکی از وضعیت های {16,17,18,19,20} قرار داشته باشد و تعداد یالهایی که راس c با آنها به راس بعد از خود روی اقدام بعدی و راس قبلی خود روی اقدام قبلی متصل است برابر ۲ باشد و یا بطور ساده تر راس c با یالهایی به راس بعد و قبل از خودش وصل باشد، به اقدامی که راس c روی آن قرار دارد، پاداش داده می شود یعنی راس روی اقدام به سمت وضعیت های داخلی تر و با توجه به شکل ۱۰، بالاتر این اقدام حرکت می کند. اگر راس c در وضعیت داخلی (وضعیت شماره 16) قرار داشته باشد و اقدام مربوط به آن پاداش بگیرد در همان وضعیت باقی می ماند.



شکل ۱۰- نحوه پاداش دادن به راس c روی اقدام a_4

همچنین اگر تعداد یالهایی که راس روی اقدام u با آنها به راس بعد از خود روی اقدام بعدی و راس قبل خود روی اقدام قبلی متصل است کمتر از ۲ باشد و یا بطور ساده تر راس روی اقدام u به یکی یا



شکل ۸- نحوه اعمال عملگر جهش Order Based (Swap)

د) عملگر جریمه و پاداش[10]: از آنجا که هر کروموزوم به صورت یک آتاماتای یادگیر نشان داده شده است در هر یک از آتاماتاها پس از بررسی میزان برازندگی محل قرار گیری راس متعلق به یک اقدام آتاماتا (ژن کروموزوم) که به صورت تصادفی انتخاب می شود، آن اقدام (ژن) پاداش یا جریمه می شود. در اثر پاداش دادن یا جریمه کردن یک اقدام، وضعیت راس متعلق به آن اقدام در مجموعه وضعیت های اقدام، تغییر می کند. اگر یک راس در وضعیت مرزی یک اقدام قرار داشته باشد، جریمه شدن اقدام آن باعث تغییر وضعیت آن راس از روی این اقدام به روی یک اقدام دیگر می شود و در نتیجه باعث ایجاد جایگشت جدیدی می گردد. نرخ این عملگر باید پایین باشد زیرا این عملگر، یک عملگر جستجوی تصادفی است و اگر با نرخ بالا اعمال شود باعث کاهش در کارایی الگوریتم می شود. عملگر جریمه و پاداش با توجه به نوع آتاماتای یادگیر متفاوت خواهد بود. شکل ۹ شبه کد عملگر پاداش اقدام u از آتاماتای LA با اتصالات مشابه آتاماتای ستلین را نشان می دهد.

[10] Penalty and Reward

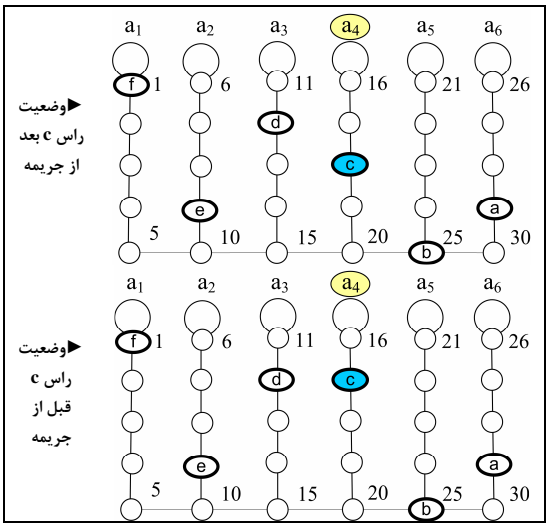
هیچکدام از راس های بعد و قبل از خودش وصل باشد، در این صورت مسیر ایجاد شده در این قسمت مناسب نبوده و اقدام u که راس روی آن قرار دارد، جریمه می شود. شبه کد عملگر جریمه اقدام u از آتاماتای LA با اتصالات مشابه آتاماتای ستلین در شکل ۱۱ نشان داده شده است.

```
Procedure Penalize( LA , u )
Begin
   If (LA.Action(u).State) Mod N <> 0  Then
      Inc(LA.Action(u).State);
   Else
      MaxEdgeCount=0 ;
    For U = 1 To |V_G| Do
      Create new Automata LA′ from LA by
      swapping Objects on u and U;
      If Edge_Count(Specified by LA') > MaxEdgeCount Then
            MaxEdgeCount =  Edge_Count (Specified by LA');
            BestNode = U;
       End If
     End For
      LA.Action(BestNode).State=
      LA.Action(BestNode).ActionNumber*N;
     Swap(LA.Action(u).State, LA.Action (BestNode).State);
   End Else
End Procedure
```

شکل ۱۱- شبه کد عملگر جریمه اقدام u از آتاماتای LA با اتصالات مشابه آتاماتای ستلین

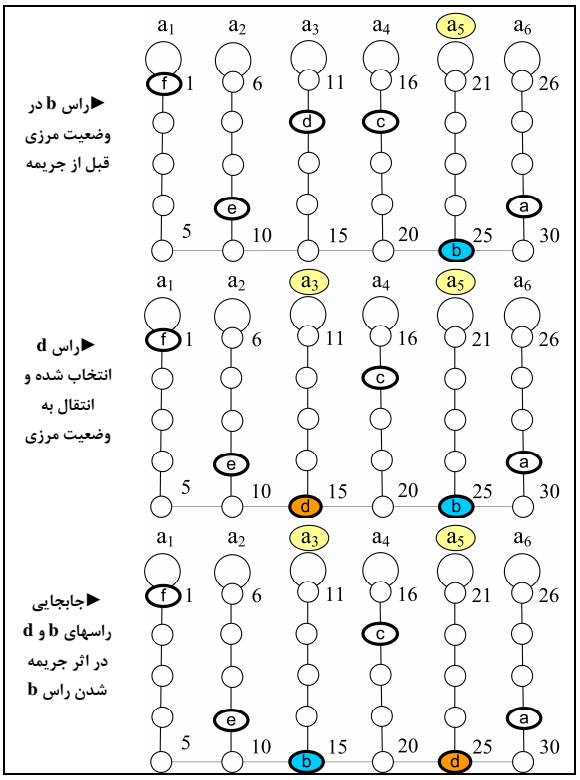
نحوه حرکت چنین راسی برای دو حالت مختلف در زیر آمده است.

الف) راس مورد نظر در وضعیتی غیر از وضعیت مرزی اقدام قرار داشته باشد: جریمه نمودن این اقدام سبب کم اهمیت و ناپایدار شدن آن می شود. برای مثال نحوه حرکت یک راس مانند c در چنین وضعیتی هنگام جریمه شدن اقدام مربوط به آن، در شکل ۱۲ نشان داده شده است.



شکل ۱۲- نحوه جریمه کردن راسی واقع در وضعیت غیر مرزی

ب) راس مورد نظر مانند b در شکل ۱۳ در وضعیت مرزی اقدام قرار داشته باشد: در این حالت راس دیگری مانند d از مجموعه راس های مستقر روی دیگر اقدامها را پیدا می کنیم بطوریکه با ایجاد یک آتاماتای موقت و در نتیجه یک جایگشت جدید، اگر در این جایگشت جای راس b و راس پیدا شده d عوض شوند، مسیر فعلی ترمیم گردد و بین راس b و راس های قبل و بعد از خودش و بین d و راس های قبل و بعد خود، یالهایی موجود باشد. در اینصورت اگر راس پیدا شده در وضعیت مرزی اقدام قرار داشته باشد جای دو راس عوض می شود و در غیر اینصورت مانند شکل، ابتدا راس پیدا شده به وضعیت مرزی اقدام خود منتقل و سپس جابجایی صورت می پذیرد.



شکل ۱۳- نحوه جریمه شدن یک راس موجود در وضعیت مرزی

در ادامه ساختار و حلقه اصلی الگوریتم تکاملی جدید برای حل مسئله جستجوی مدار هامیلتونی در گراف، در شکل ۱۴ ارائه شده است.

```
Function HCP_Solver(Edges of Graph) : Sequence_of_Vertices_in_Hamiltonian_Cycle
Begin
    P = Size of Population;
    Create the Initial Population LA1 … LAP;
    Copy Initial Population to OldPop;
    OldPop.EvaluateFitness();
    While( Not Terminate Condition ) Do
        For i = 1 To P-1 Do
            Parent1=Select (OldPop); Parent2=Select(OldPop);
            If (Random < CrossoverRate) Then
                Crossover ( Parent1, Parent2 );//Crossover Operator
                Creates Child1 & Child2
            End If
            If (Random < MutationRate) Then
                Mutation ( Child1 ); Mutation ( Child2 );
                Child1.EvaluateFitness(); Child2.EvaluateFitness();
                NewPop.LAi =child1; NewPop.LAi+1=Child2;  i=i+2;
            End If
        End For
        For i = 1 To P Do
            u = Random(1 to Vertex Count of Graph) ;
            If (Edge_Countu(NewPop.LAi) = 2) Then
                Reward(NewPop.LAi , u );
            Else If (Edge_Countu(NewPop.LAi) =0 OR 1)
                Penalize(NewPop.LAi , u );
            End If
        End For
        NewPop.EvaluateFitness();
        Copy NewPop to OldPop with Applying Elitism;
    End While
End Function
//Edge_Countu(NewPop.LAi)=Sum of Edges Count exists between Vertices in Action u and u+1 and Vertices in Action u and u-1 inside i'th LA in the New population;
```

شکل ۱۴- الگوریتم تکاملی جدید برای جستجوی مدار هامیلتونی در گراف

۹- نتایج آزمایشها

۹-۱- مسائل مورد آزمایش

برای حل مسئله جستجوی مدار هـامیلتونی در گـراف مـسایل مختلفی با چگالی یال متفاوت مورد آزمایش قرار گرفته اند. تعدادی از این مسئله ها برگرفته از کتابخانه TSP[11] هستند. تعدادی نیـز بـا توجه به رهنمودهای موجود در مقالات قبلی تولید شده اند. در مورد این مسئله منظور از چگالی یالهـا، تعـداد یالهـای موجـود در گـراف فعلـی نـسبت بـه گـراف کامـل شـامل تمـام یالهـای ممکـن مـی باشد[۷،۲۷]. برای مقایـسه الگوریتمهـا، ۸ نمونـه مـسئله دارای ۵۰، ۸۰، ۱۰۰، ۱۲۰، ۱۵۰، ۲۰۰، ۲۵۰و ۳۰۰ راس با چگالیهای مختلـف تولید شده اند. در مورد ارتباط چگالی یالهـای گـراف بـا هـامیلتونی بودن آن، نظریه معروف در تئوری گرافها توسط posa [۲۸] مطـرح شده است. نظریه به این صورت اسـت کـه تعـداد یـال $\frac{1}{2}n\ \log(n)$ یک مقدار آستانه برای هامیلتونی بودن گراف است که n تعداد گـره های گراف است. به این معنی که تقریبا تمام گرافهـای دارای مقـدار بیشتر از این مقدار آستانه، هامیلتونی هستند و تقریبا تمام گرافهای دارای تعداد یالهای کمتر از ایـن مقـدار نیـز هـامیلتونی نیـستند. از دیگر خصوصیات گـراف ماننـد متـصل بـودن آن نیـز معمـولا دارای چنین پدیده آستانه ای مشابه شناخته می شوند.

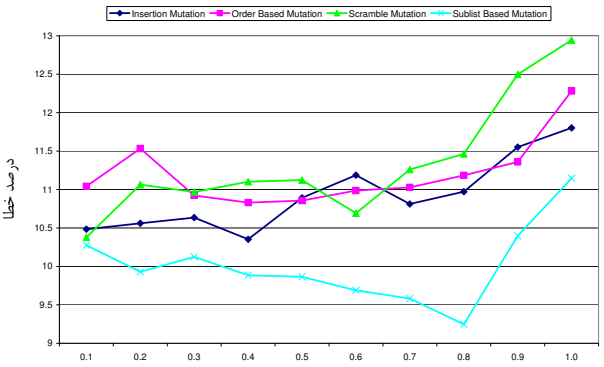
۹-۲- مقایسه نتایج الگـوریتم هـای مختلـف حـل مـسئله جستجوی مدار هامیلتونی در گراف

برای حل مسئله جستجوی مدارهامیلتونی در گراف دو دسته از آزمایشات انجام شده اند. دسته اول آزمایشات مربوط به بحث تنظیم پارامترهای الگوریتم تکاملی جدیـد بـوده اسـت. در ایـن آزمایـشات شش مسئله بکار رفته اند که گرافهای شامل ۵۰، ۱۰۰، ۱۵۰، ۲۰۰، ۲۵۰ و ۳۰۰ راس هـستند و بـا تعـداد یـال $\frac{1}{2}n\ \log(n)$ بـصورت تصادفی تولید شده اند. از بررسی های انجام شـده چنـین اسـتنتاج شده است که با توجـه بـه شـکل ۱۵ عملگـر جهـش Sublist Based همیشه بهتر از دیگر عملگرها بوده و ۰،۷ تا ۰،۸ نیـز مناسـب تـرین نرخ اجرای این عملگر می باشد. شکل ۱۶ نیز نشان می دهد که سه عملگر جابجایی Partially Mapped، Cycle وReverse Ordered بـا نـرخ بالای ۰،۵ عملکرد بهتری نسبت به دیگر عملگرها، نشان می دهنـد. شکل ۱۷ نتـایج حاصـل از اجـرای الگـوریتم تکـاملی جدیـد بـرای عملگرهای جهش و جابجایی مختلـف در تعامـل بـا یکـدیگر را روی شش مسئله انتخابی نشان می دهد. همچنین عمگـر انتخـاب رتبـه بندی نیز با توجه به شکلهای ۱۸ و ۱۹ عملکـرد بهتـری نـسبت بـه چرخ رولت دارد. شکل ۱۸ نشان می دهد که عملگر جهـش Sublist Based در کنار عملگر انتخاب رتبه بندی و با انواع اتصالات آتاماتـای ستلین و کرینیسکی و ترکیب آنهـا(All)، بهتـرین عملکـرد را دارنـد. عبارت ALL در ایـن شـکل حـاکی از ایـن مـسئله اسـت کـه انـواع مختلفی از اتصالات آتاماتا در جمعیت ژنتیکی بصورت تصادفی بکـار رفته است. شکل ۱۹ نیز نشان می دهد که عملگر جابجـایی Cycle و Reverse Ordered در کنـار عملگـر انتخـاب رتبـه بنـدی و بـا انـواع اتصالات آتاماتای ستلین و کرینیسکی و ترکیـب آنهـا(All)، بهتـرین عملکرد را دارند. در اجـرای الگوریتمهـا بـرای مقایـسه آنهـا از ایـن بهترین مقادیر استفاده شده است. دسته دوم از آزمایشات مربوط بـه مقایسه الگوریتمهای مختلف است. در شکل ۲۰، درصد خطا در پیدا کـردن حلقـه هـامیلتونی بـرای ۸ نمونـه مـسئله دارای تعـداد یال $\frac{1}{2}n\ \log(n)$ (n برابر بـا تعـداد راسـها) بـا اجـرای الگوریتمهـای مختلف برای ۵۰۰ تکرار، مورد مقایسه قرار گرفتـه اسـت. در شـکل ۲۱ و ۲۲ الگوریتمهـای مختلـف، روی گرافهـای تولیـد شـده دارای چگالی یال ۰،۱۵ و۰،۱۷ برای ۵۰۰ تکرار، مورد مقایسه قـرار گرفتـه اند. همچنان که دیده می شود نتایج بدست آمده از الگوریتم تکاملی جدید، حاکی از آن است که در اکثر موارد این الگوریتم قـادر اسـت بدون خطا یا با خطای بسیار کم، مدار هامیلتونی موجود در گراف را پیـدا کنـد. جـدول ۲ تعـداد تکـرار هـا و زمـان مـورد نیـاز بـرای الگوریتمهای مختلف جهت پیـدا کـردن مـدار هـامیلتونی در داخـل گراف هایی دارای ۱۰۰ راس را نشان می دهد. بـرای ایـن آزمـایش، چگـالی گرافهـای دارای ۱۰۰ راس از مقـدار ۰،۰۷ تـا ۰،۳۵ تغییـر کرده است و چنانچه مشاهده می شود برای چگالی هـای زیـر ۰،۲۵ بجز الگوریتم ژنتیکی و الگـوریتم تکـاملی جدیـد، دیگـر الگوریتمهـا توانایی پیدا کردن مدار هامیلتونی را در زمان معقول، ندارنـد. درایـه هایی از جدول که دارای علامت * می باشند حاکی از همگرا نشدن الگوریتم ها هستند. شرایط اجرای این آزمایش به این صـورت بـوده است که تعداد افراد جمعیت ۵۰، عمق آتاماتـا ۵، عملگـر جابجـایی Reverse Ordered با نرخ ۰،۸، عملگر جهـش SublistBased بـا نرخ ۰،۸ ، عملگر انتخاب از نوع رتبه بندی و نوع اتـصالات آتاماتـای بکار رفته در کروموزومهای جمعیت از نوع ستلین بوده است. جـدول ۳ نیز مثالهایی از زمان اجرا و تعداد تکـرار الگـوریتم ترکیبـی بـرای گرافهای مختلف با چگالی یال متفاوت را نشان می دهد.

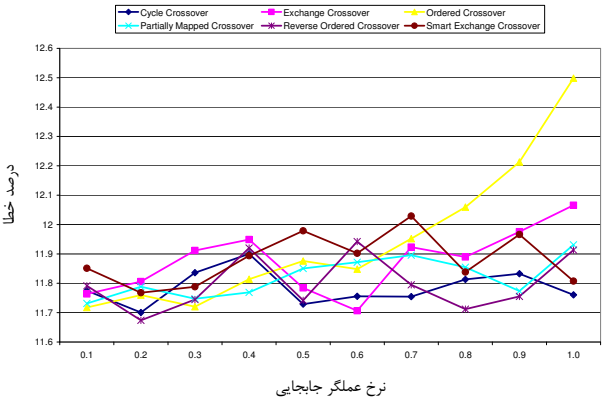
۱۰- نتیجه گیری و پیشنهادات

در این مقاله، مـساله جـستجوی مـدار هـامیلتونی در گرافهـای بدون جهت توسط الگوریتمهای حریصانه، کاهشی تکراری، آتاماتـای یادگیر و الگوریتم ژنتیک و یک الگوریتم تکاملی جدید مورد بررسـی قرار گرفت. الگوریتم تکاملی جدید از دو روش الگوریتم های ژنتیکی و آتاماتاهای یادگیر بطور همزمان بـرای جـستجو در فـضای حـالات جواب مسئله، استفاده می نماید. در این مقاله نشان داده شد کـه بـا اسـتفاده همزمـان آتاماتـای یـادگیر و الگـوریتم ژنتیـک در فراینـد جستجو، امکان همگرایی الگوریتم برای پیدا کردن مـدار هـامیلتونی افزایش پیدا می کند و در مدت زمانی معقول ممکن است در حالیکه با استفاده از دیگر الگوریتمها چنین کاری امکان پذیر نیـست. نتـایج آزمایش ها، برتری نتایج حاصل از روش تکاملی جدید را نـسبت بـه روش‌های دیگر نشان می دهد.

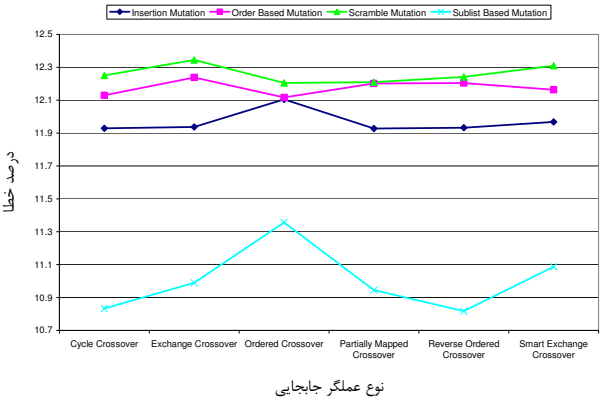
[11] TSP Library



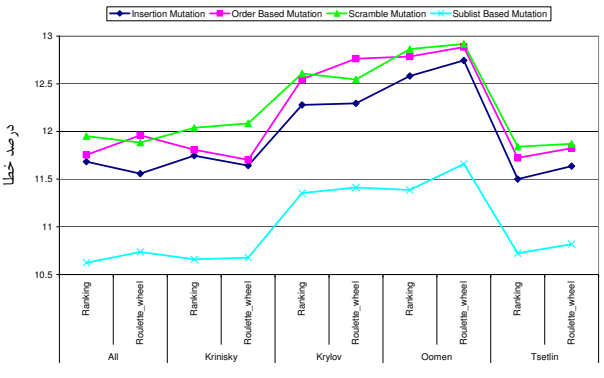
شکل ۱۵-میانگین درصد خطای حاصل از عملگرهای جهش با نرخهای مختلف



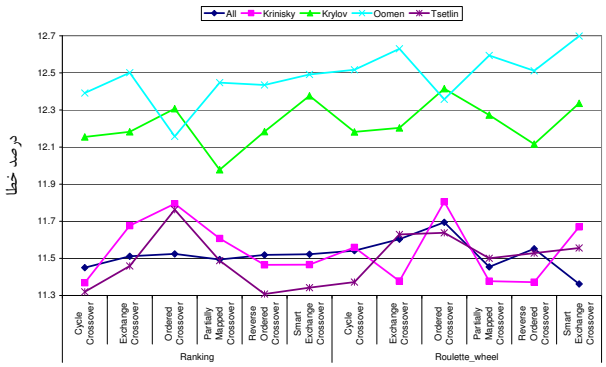
شکل ۱۶-میانگین درصد خطای حاصل ازعملگرهای جابجایی با نرخهای مختلف



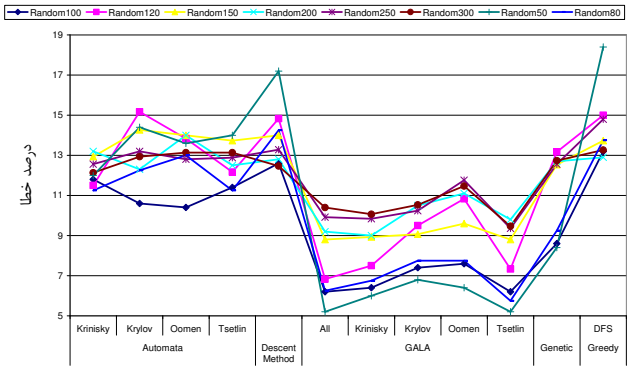
شکل ۱۷- میانگین درصد خطای حاصل از اجرای عملگرهای جابجایی و جهش مختلف در کنار یکدیگر



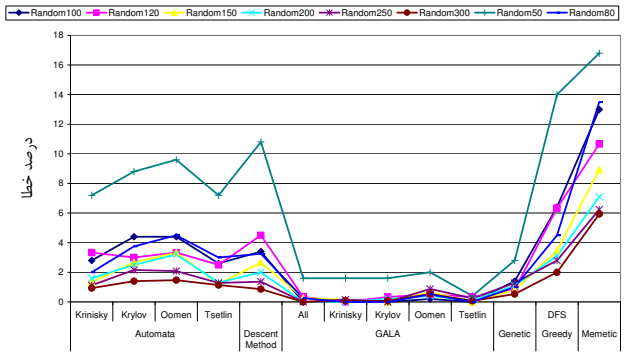
شکل ۱۸- میانگین درصد خطای حاصل از عملگرهای انتخاب و جهش مختلف با اتصالات متفاوت آتاماتای یادگیر



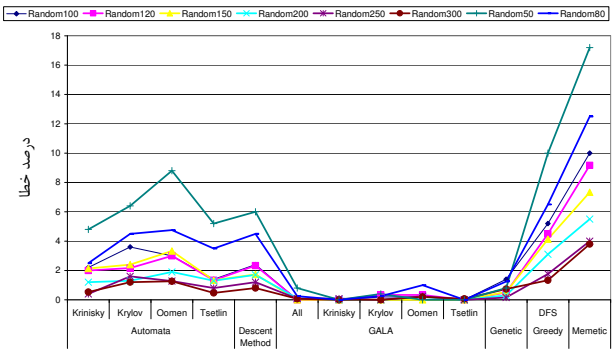
شکل ۱۹-میانگین درصد خطای حاصل از عملگرهای انتخاب و جابجایی مختلف با اتصالات متفاوت آتاماتا



شکل ۲۰- میانگین درصد خطا برای مسائل دارای تعداد یال $1/2(n\log(n))$



شکل ۲۱-میانگین درصد خطا برای مسائل دارای چگالی یال ۰,۱۵



شکل ۲۲-میانگین درصد خطا برای مسائل دارای چگالی یال ۰,۱۷

جدول ۲- تعداد تکرار ها و زمان مورد نیاز برای الگوریتمهای مختلف جهت پیدا کردن مدار هامیلتونی در گراف هایی دارای ۱۰۰ و با چگالی یال متفاوت

چگالی یالها	تعداد یال	Greedy (DFS)		Descent Method		Memetic Algorithm		Genetic Algorithm		Learning Automata		GALA	
		تعداد تکرار	زمان (ms)	تعداد تکرار	زمان (ms)	تعداد تکرار	زمان (ms)	تعداد تکرار	زمان (ms)	تعداد تکرار	زمان (ms)	تعداد تکرار	زمان (ms)
۰,۰۷	۳۴۷	*	*	*	*	*	*	۲۷۷۲۸	۱۸۴۹۲۲	*	*	۶۶۷۳	۸۶۱۵۶
۰,۰۸	۳۹۶	*	*	*	*	*	*	۱۳۰۲۵	۸۶۳۱۲	*	*	۵۴۳۹	۶۹۷۳۴
۰,۰۹	۴۴۶	*	*	*	*	*	*	۱۷۱۱	۱۱۱۴۱	*	*	۶۷۴	۹۸۹۱
۰,۱	۴۹۵	*	*	*	*	*	*	۱۰۴۰	۶۹۶۹	*	*	۲۸۰	۴۵۰۰
۰,۱۱	۵۴۴	*	*	*	*	*	*	۷۵۷	۵۴۰۶	*	*	۳۳۴	۴۹۵۳
۰,۱۲	۵۹۴	*	*	*	*	*	*	۸۱۲	۵۶۲۵	*	*	۳۵۷	۵۲۸۱
۰,۱۳	۶۴۴	*	*	*	*	*	*	۶۳۰	۴۴۲۱	*	*	۲۲۶	۳۶۷۲
۰,۱۴	۶۹۳	*	*	*	*	*	*	۴۰۶	۲۸۹۱	*	*	۱۴۹	۲۴۳۸
۰,۱۵	۷۴۲	*	*	*	*	*	*	۳۹۷	۲۸۱۳	*	*	۱۶۱	۲۵۷۸
۰,۱۶	۷۹۲	*	*	*	*	*	*	۳۲۳	۲۴۸۵	*	*	۱۳۸	۲۲۳۵
۰,۱۷	۸۴۲	*	*	*	*	*	*	۱۶۹	۱۲۶۶	*	*	۷۷	۱۳۹۰
۰,۱۸	۸۹۱	*	*	*	*	*	*	۲۲۹	۱۷۱۸	*	*	۶۵	۱۲۱۹
۰,۱۹	۹۴۰	*	*	*	*	*	*	۲۷۰	۱۹۵۳	*	*	۶۳	۱۱۴۰
۰,۲	۹۹۰	*	*	*	*	*	*	۱۵۹	۱۳۲۹	*	*	۶۱	۱۱۸۷
۰,۲۵	۱۲۳۸	*	*	*	*	*	*	۹۷	۸۶۰	۱۴۸	۲۳۵	۳۵	۸۱۲
۰,۳	۱۴۸۵	۷	۲۵۰	۱	۲۳۵	*	*	۷۵	۷۵۰	۱۰۹	۹۴	۲۰	۵۹۴
۰,۳۵	۱۷۳۲	۴۵	۲۸۲	۱	۱۲۵	۱۱۸	۱۸۷	۴۶	۵۳۱	۱۴	۹۴	۱۹	۵۴۷

جدول۳-مثالهایی از زمان اجرا وتعداد تکرار الگوریتم تکاملی برای گرافهای مختلف

نام مسئله	چگالی ۰,۱۳		چگالی ۰,۱۵		چگالی ۰,۱۷	
	تعداد تکرار	زمان اجرا (ms)	تعداد تکرار	زمان اجرا (ms)	تعداد تکرار	زمان اجرا (ms)
Random100	۳۵۴	۴۷۶۶	۲۳۵	۳۲۹۷	۱۷۸	۲۵۱۶
Random120	۲۲۶	۴۰۳۲	۲۱۹	۳۸۹۱	۱۰۳	۱۹۲۲
Random150	۱۶۴	۴۰۰۰	۷۸	۲۰۴۷	۸۲	۲۱۷۲
Random200	۲۸۹	۹۵۶۳	۱۲۸	۴۵۹۳	۱۰۹	۴۰۴۷
Random250	۳۴۷	۱۴۸۹۱	۴۴۳	۱۸۹۵۳	۲۸۲	۱۳۵۰۰
Random300	۱۳۱۱	۶۸۵۴۷	۲۴۱	۱۴۲۱۹	۱۰۰	۶۵۴۷
Random50	۶۷۲	۴۳۱۳	۳۴۳	۲۴۲۲	۶۱	۶۱۰
Random80	۳۱۹	۳۴۸۴	۱۹۶	۲۲۱۹	۱۴۲	۱۶۵۶

۱۱- منابع و مراجع

[1] R. Santana and E. P. de León, "A hybrid genetic algorithm for a Hamiltonian path problem", In Proceedings of the First Symposium on Artificial Intelligence (CIMAF-97), pp. 126-132, Habana, Cuba, March 1997.

[2] E. P. de León, R. Santana, and A. Ochoa., "A genetic algorithm for a Hamiltonian path problem: Mutation - crossover interaction.", In Proceedings of the 13th ISPE/IEE International Conference on CAD/CAM Robotics & Factories of the Future'97, pp. 1001-1006, Universidad Tecnológica de Pereira, Colombia, December 15- 17, 1997.

[3] E. P. de León, A. Ochoa, and R. Santana., "A genetic algorithm for a Hamiltonian path problem", In Proceedings of the International Conference on Industrial and Engineering Applications of AI and Expert Systems, Atlanta, USA, 1997.

[4] E. P. de León, A. Ochoa, and R. Santana., "A genetic algorithm for a Hamiltonian path problem", Technical Report ICIMAF 96-01, CENIA 95-01, Institute of Cybernetics, Mathematics and Physics, Havana, Cuba, September 1996.

[5] J. Aleš, B. Mohar, T. Pisanski, "Heuristic search for Hamilton cycles in cubic graphs", Discrete Mathematics, Elsevier, Vol. 307, pp. 303–309, 2007.

[6] M. DeLeon, "A Study of Sufficient Conditions for Hamiltonian Cycles", Department of Mathematics and Computer Science, Seton Hall University.

[7] I.A. Wagner, A.M. Bruckstein, "Hamiltonian(t)-an ant-inspired heuristic for recognizingHamiltonian graphs", Proceedings of the 1999 Congress on Evolutionary Computation, CEC 99, Vol. 2, pp. -1469, 1999.

[8] D. Rautenbach, I. Stella, "Note On the maximum number of cycles in a Hamiltonian graph", Discrete Mathematics, Elsevier, Vol. 304, pp. 101–107, 2005

[9] K. A. De Jong, and W.M. Spears, "Using genetic algorithms to solve NP-complete problems", Proceeding of Third International Conference on Genetic Algorithms and their Applications, pp. 124-132, Kaufmann, 1989.

[10] E. Ando , M. Yamashita, T. Nakata, Y. Matsunaga, "The statistical longest path problem and its application to delay analysis of logical circuits", Proceedings of the 8th ACM/IEEE international workshop on Timing issues in the specification and synthesis of digital systems, pp. 134 – 139, Monterey, California, USA, 2002.

[11] http://en.wikipedia.org/wiki/Hamiltonian_path

[12] L. Posa, "Hamiltonian circuits in random graphs", Discrete Mathematics, Vol. 14, pp. 359-364, 1976.

[13] T.I. Fenner and A.M. Frieze, "On the existence of hamiltonian cycles in a class of random graphs", Discrete Mathematics, Vol. 45, pp 301-305, 1983.

[14] F. Rubin, "A Search Procedure for Hamilton Paths and Circuits'", Journal of the ACM, Vol. 21, No. 4, October 1974.

[15] I.B. SHIELDS, "Hamilton Cycle Heuristics in Hard Graphs", PHD Thesis, Computer Science, North Carolina State University, 2004.

[16] R. Uehara, and Y. Uno, "Efficient Algorithms for the Longest Path Problem", Lecture Notes in Computer Science, Algorithms and Computation, Vol. 3341, pp. 871-883, Springer Berlin / Heidelberg, December 2004.

[17] A. M. Dean, C. J. Knickerbocker, P. F. Lock, and M. Sheard, "A survey of graphs Hamiltonian-connected from a vertex", In Graph theory, combinatorics and applications, Vol. 1, pp. 297–313, 1991.

[18] D.E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison Wesley, Reading MA, 1989.

[19] T. Starkweather, S. McDaniel, C. Whitley, K. Mathias, D. Whitley, "A comparison of genetic sequencing operators", In proceedings of the forth International Conference on Genetic Algorithms, San Diego, CA, pp. 69–76, Morgan Kaufmann, San Francisco, 1991.

[20] J.H. Holland, "Adaptation in Natural and Artificial Systems", Ann Arbor, University of Michigan Press, 1975.

[21] K.S. Narendra, and M.A.L. Thathachar, "Learning Automata: An Introduction", Prentice-hall, Englewood cliffs, 1989.

[22] B. J. Oommen, and D. C. Y. Ma, "Deterministic Learning Automata Solution to the Keyboard Optimization Problem", IEEE Transaction on Computers, Vol. 37, No. 1 ,pp. 2-3, 1988.

[23] M. RezapourSaleh, and M. R.Meybodi, "A Hybrid Algorithm for Solving Graph Isomorphism Problem", Proceedings of the Second International Conference on Information and Knowledge Technology (IKT2005), Tehran, Iran, May 24-26, 2005.

[24] K. Asghari, A. Safari Mamaghani, and M. R. Meybodi, "An Evolutionary Approach for Query Optimization Problem in Database", Proceedings of International Joint Conferences on Computers, Information and System Sciences, and Engineering (CISSE2007), University of Bridgeport, England, December 2007.

[25] A. Safari Mamghani, K. Asghari, M.R. Meybodi, and F.Mahmoodi, "A New Method based on Genetic Algorithm for minimizing Join operations Cost in Data Base", Proceedings of 13th Annual CSI Computer Conference of Iran, Kish Island, Iran, March 9-11, 2008.

[26] B. Zarei, K.Asghari, and M.R. Meybodi, "A Hybrid Method based on Clustering for Solving Large Traveling salesman Problem", Proceedings of 13th Annual CSI Computer Conference of Iran, Kish Island, Iran, March 9-11, 2008.

[27] T. Bohman, A.M. Frieze and R. Martin, "How many random edges make a dense graph Hamil-tonian?", Random Structures and Algorithms, Vol. 22 , pp. 33-42, 2003.

[28] B. Bollobas, "Graph Theory - An Introductory Course", Springer-Verlag, pp. 139-142, New York, 1979.