

## یک روش ترکیبی برای حل مساله مرتب سازی ترتیبی

باقر زارعی

محمد رضا میبیدی

دانشکده مهندسی کامپیوتر، دانشگاه آزاد اسلامی واحد  
شبستر، ایران

دانشکده برق، مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه  
صنعتی امیر کبیر، تهران، ایران

Zarei\_Bager@yahoo.com

MMeybodi@aut.ac.ir

چکیده - یکی از مسائل بسیار مهم در تئوری گراف ها، مساله مرتب سازی ترتیبی می باشد. آتوماتاهای یادگیر و الگوریتم های ژنتیکی هر دو از ابزارهای جستجو می باشند که برای حل بسیاری از مسائل  $NP$ -Complete بکار برده می شوند. در این مقاله یک الگوریتم ترکیبی برای حل مساله مرتب سازی ترتیبی پیشنهاد شده است. این الگوریتم از دو روش الگوریتم های ژنتیکی و آتوماتاهای یادگیر بطور همزمان برای جستجو در فضای حالت استفاده می نماید. نشان داده شده است که با استفاده همزمان از آتوماتای یادگیر و الگوریتم ژنتیک در فرایند جستجو، سرعت رسیدن به جواب افزایش چشمگیری پیدا می کند و همچنین از بدام افتادن الگوریتم در حداقل های محلی جلوگیری می نماید. نتایج آزمایش ها، برتری الگوریتم ترکیبی را نسبت به الگوریتم ژنتیکی و آتوماتاهای یادگیر نشان می دهد.

کلمات کلیدی: مساله مرتب سازی ترتیبی، آتوماتای یادگیر، الگوریتم ژنتیک

### ۱- مقدمه

حلشان به طور نمایی افزایش می یابد. این مسائل، مسائل بهینه سازی ترکیبی هستند، که زمان حل آنها به صورت تابعی غیر چند جمله ای است. مساله مرتب سازی ترتیبی یکی از آنها می باشد که حل مساله به معنای پیدا کردن بهترین تور، در مقایسه با تورهای شناخته شده قبلی نمی باشد بلکه همچنین باید ثابت کرد که توری با هزینه کمتر از تور پیدا شده نیز وجود ندارد.

آتوماتاهای یادگیر و الگوریتم های ژنتیکی، هر دو ابزار جستجوی عمومی می باشند که برای حل بسیاری از مسائل  $NP$ -Complete بکار برده شده است. در این مقاله یک الگوریتم ترکیبی برای حل مساله مرتب سازی ترتیبی پیشنهاد شده است. این الگوریتم از دو روش الگوریتم های ژنتیکی و آتوماتاهای یادگیر بطور همزمان برای جستجو در فضای حالت استفاده می نماید. نشان داده شده است که با استفاده همزمان از آتوماتای یادگیر و الگوریتم ژنتیک در فرایند جستجو، سرعت رسیدن به جواب افزایش چشمگیری پیدا می کند و همچنین از بدام افتادن الگوریتم در حداقل

گراف ها ابزارهای قدرتمندی هستند که به طور گسترده در کاربردهای متعددی مورد استفاده قرار می گیرند. یکی از مسائل بسیار مهم در تئوری گراف ها، مساله مرتب سازی ترتیبی می باشد. بسیاری از کاربردهای عمومی از جمله طراحی حلقه های شبکه های Sonet، کابل های برق، مسیر هواپیما ها، مسیریابی وسائط نقلیه و ... را می توان با مساله مرتب سازی ترتیبی مدل کرد.

مساله مرتب سازی ترتیبی، نسخه ای از مساله فروشنده دوره گرد نامتقارن می باشد که یکسری محدودیت های اولویتی باید روی رئوس رعایت شود. مساله در Escudero (۱۹۸۸) معرفی شد. فرض کنید که یک گراف کامل داریم که هر یال  $(u, v) \in E$ ، هزینه نامنفی  $C(u, v)$  را دارد. هدف یافتن سیکل همیلتنی با حداقل هزینه می باشد بطوریکه محدودیت های اولویتی در بین رئوس ارضاء شود.

بعضی از مسائل وجود دارند که با افزایش بعد آنها، زمان

های محلی جلوگیری می نماید.

## ۲- الگوریتم های ژنتیک

الگوریتم های ژنتیکی که بر مبنای ایده تکامل در طبیعت عمل می نماید، بر روی جمعیتی از راه حل های بالقوه به جستجوی راه حل نهایی می پردازد. در هر نسل، بهترین های آن نسل انتخاب می شوند، و پس از زاد و ولد، مجموعه جدیدی از فرزندان را تولید می کنند. در این فرایند افراد مناسبتر با احتمال بیشتری در نسل های بعدی باقی خواهند ماند.

در آغاز الگوریتم، تعدادی از افراد<sup>۱</sup> (جمعیت اولیه) به صورت تصادفی ساخته شده و تابع هدف برای تک تک آنها ارزیابی می شود. اگر شرط رسیدن به جواب برقرار نباشد ( به جواب بهینه نرسیده باشیم)، نسل بعدی با انتخاب والدین براساس میزان برازندگی شان تولید می شود و فرزندان با احتمال ثابتی دچار جهش می شوند. سپس میزان برازندگی فرزندان جدید محاسبه شده و جمعیت جدید، از جایگزینی فرزندان با والدین ایجاد می شود و این فرآیند تا برقرار شدن شرط خاتمه تکرار می شود.

## ۳- اتوماتاهای یادگیر

یادگیری در اتوماتاهای یادگیر، انتخاب یک اقدام<sup>۲</sup> بهینه از میان یک مجموعه از اقدام های مجاز اتوماتا می باشد. این اقدام روی یک محیط تصادفی اعمال می شود و محیط به این اقدام اتوماتا بوسیله یک پاسخ تصادفی از مجموعه پاسخ های مجاز جواب می دهد. پاسخ محیط بصورت آماری به اقدام اتوماتا وابسته است. واژه محیط شامل اجتماع تمام شرایط خارجی و تاثیرات آنها روی عملکرد اتوماتا می باشد.

برای یک گراف با اندازه  $n$ ،  $n!$  جایگشت مختلف از رئوس وجود دارد و در صورتیکه از اتوماتاهای یادگیر برای حل کردن مساله مرتب سازی ترتیبی استفاده شود، اتوماتا باید  $n!$  اقدام داشته باشد که تعداد زیاد اقدام ها سرعت همگرایی اتوماتا را کاهش می دهد. به همین جهت اتوماتای مهاجرت اشیاء<sup>۳</sup> توسط اومن<sup>۴</sup> و ما<sup>۵</sup> پیشنهاد شده است.

## ۴- الگوریتم جستجوی ترکیبی برای حل مساله مرتب سازی ترتیبی

با ترکیب الگوریتم ژنتیک و اتوماتای یادگیر و تلفیق مفاهیم ژن، کروموزوم، اقدام و عمق، سابقه تاریخی تکامل راه حل مساله، به شکل کارا استخراج شده و در روند جستجو مورد استفاده قرار می گیرد. خاصیت مهم الگوریتم ترکیبی، مقاومت آن در مقابل تغییرات سطحی جواب هاست، به عبارتی دیگر تعادلی انعطاف پذیر بین کارایی الگوریتم ژنتیک و پایداری اتوماتای یادگیر در الگوریتم ترکیبی وجود دارد. خود ترمیمی، تولید مثل، جریحه و پاداش (هدایت) از ویژگیهای الگوریتم ترکیبی است. در ادامه پارامترهای اصلی این الگوریتم توضیح داده شده است.

### ژن و کروموزوم:

در الگوریتم پیشنهادی برخلاف الگوریتم های ژنتیک کلاسیک، از کدگذاری دودویی برای کروموزوم ها استفاده نمی شود. هر کروموزوم توسط یک اتوماتای یادگیر از نوع مهاجرت اشیاء نشان داده می شود. بطوریکه هر کدام از ژنهای کروموزوم به یکی از اقدامهای اتوماتا نسبت داده می شود و در یک عمق مشخصی از آن اقدام قرار می گیرد.

در این اتوماتا  $\alpha = \{\alpha_1, \dots, \alpha_k\}$  مجموعه اقدام های مجاز برای اتوماتای یادگیر است. این اتوماتا  $k$  اقدام دارد (تعداد اقدام های این اتوماتا با تعداد راس های گراف برابر است). اگر راس  $u$  از گراف در اقدام  $m$  قرار گرفته باشد، در اینصورت راس  $u$  در ترتیب ملاقات کردن شهرها،  $m$ امین شهر می باشد.

$\phi = \{\phi_1, \phi_2, \dots, \phi_{KN}\}$  مجموعه وضعیت ها و  $N$  عمق حافظه برای اتوماتا می باشد. مجموعه وضعیت های این اتوماتا به  $K$  زیر مجموعه  $\{\phi_1, \phi_2, \dots, \phi_N\}$  و  $\{\phi_{N+1}, \phi_{N+2}, \dots, \phi_{2N}\}$  و  $\dots$  و  $\{\phi_{(K-1)N+1}, \phi_{(K-1)N+2}, \dots, \phi_{KN}\}$  افراز می شود و راس های گراف بر اساس این که در کدام وضعیت قرار داشته باشند دسته بندی می گردند. اگر گره  $u$  از گراف در مجموعه وضعیت های  $\{\phi_{(j-1)N+1}, \phi_{(j-1)N+2}, \dots, \phi_{jN}\}$  قرار داشته باشد در اینصورت راس  $u$  در ترتیب ملاقات کردن شهرها،  $j$ امین شهر می باشد. در مجموعه وضعیت های اقدام  $j$ ، به وضعیت  $\phi_{(j-1)N+1}$  وضعیت داخلی و به

1 Individuals

2 Action

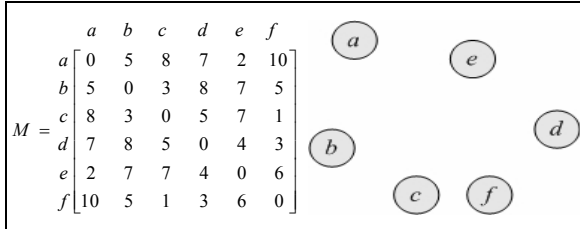
3 Object Migrating Automata

4 Oommen

5 Ma

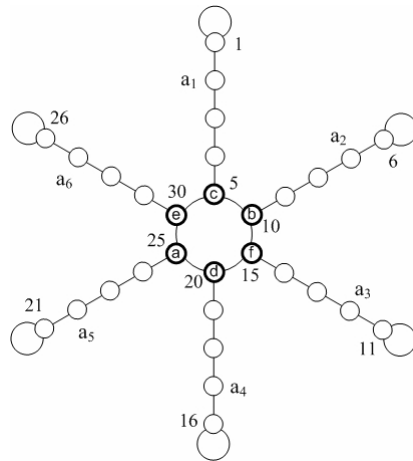
وضعیت  $\phi_{jN}$ ، وضعیت مرزی گفته می شود.

به عنوان مثال گراف کامل شکل ۱ را که شامل ۶ راس می باشد در نظر بگیرید.



شکل ۱- گراف کامل با ۶ راس

جایگشت  $\langle c, b, f, d, a, e \rangle$  از گراف شکل ۱ را در نظر بگیرید. این جایگشت توسط یک آتوماتای یادگیر با اتصالات مشابه آتوماتای ستلین در شکل ۲ نشان داده شده است. این آتوماتا دارای ۶ اقدام  $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6\}$  (به تعداد راسهای گراف) و عمق ۵ می باشد. مجموعه وضعیت های  $\{1, 6, 11, 16, 21, 26\}$  وضعیت های  $\{5, 10, 15, 20, 25, 30\}$  وضعیت های مرزی آتوماتا هستند. در ابتدا هر یک از راسهای گراف در وضعیت مرزی اقدام مربوطه قرار دارند.



شکل ۲- نمایش جایگشت  $\langle c, b, f, d, a, e \rangle$  به وسیله آتوماتای یادگیر با اتصالات مشابه آتوماتای ستلین

### جمعیت اولیه:

با فرض اینکه تعداد اعضای جمعیت n باشد، جمعیت اولیه با ایجاد n جایگشت تصادفی که محدودیت های اولویتی در بین رؤس را ارضاء می کنند، تولید می شود.

به عنوان مثال جمعیت اولیه برای گراف شکل ۱ با فرض  $n=6$  و این محدودیت که رؤس b و d قبل از راس a ملاقات

شوند، ۶ جایگشت تصادفی  $\langle b, d, e, a, f, c \rangle$ ،  $\langle d, e, f, b, c, a \rangle$ ،  $\langle e, f, b, d, a, c \rangle$ ،  $\langle c, f, b, e, d, a \rangle$ ،  $\langle b, d, c, a, e, f \rangle$  و  $\langle b, e, d, f, c, a \rangle$  که محدودیت فوق را ارضاء می کنند، می باشد. جمعیت اولیه حاصل از گراف شکل ۱ در شکل ۳ نشان داده شده است. در ابتدا هرگره در وضعیت مرزی اقدام خود قرار دارد.

### تابع برازندگی:

در الگوریتم های ژنتیک تابع برازندگی، شاخص زنده ماندن کروموزوم ها است. لذا برازندگی یک آتوماتا در مساله مرتب سازی ترتیبی به صورت زیر تعریف می شود.

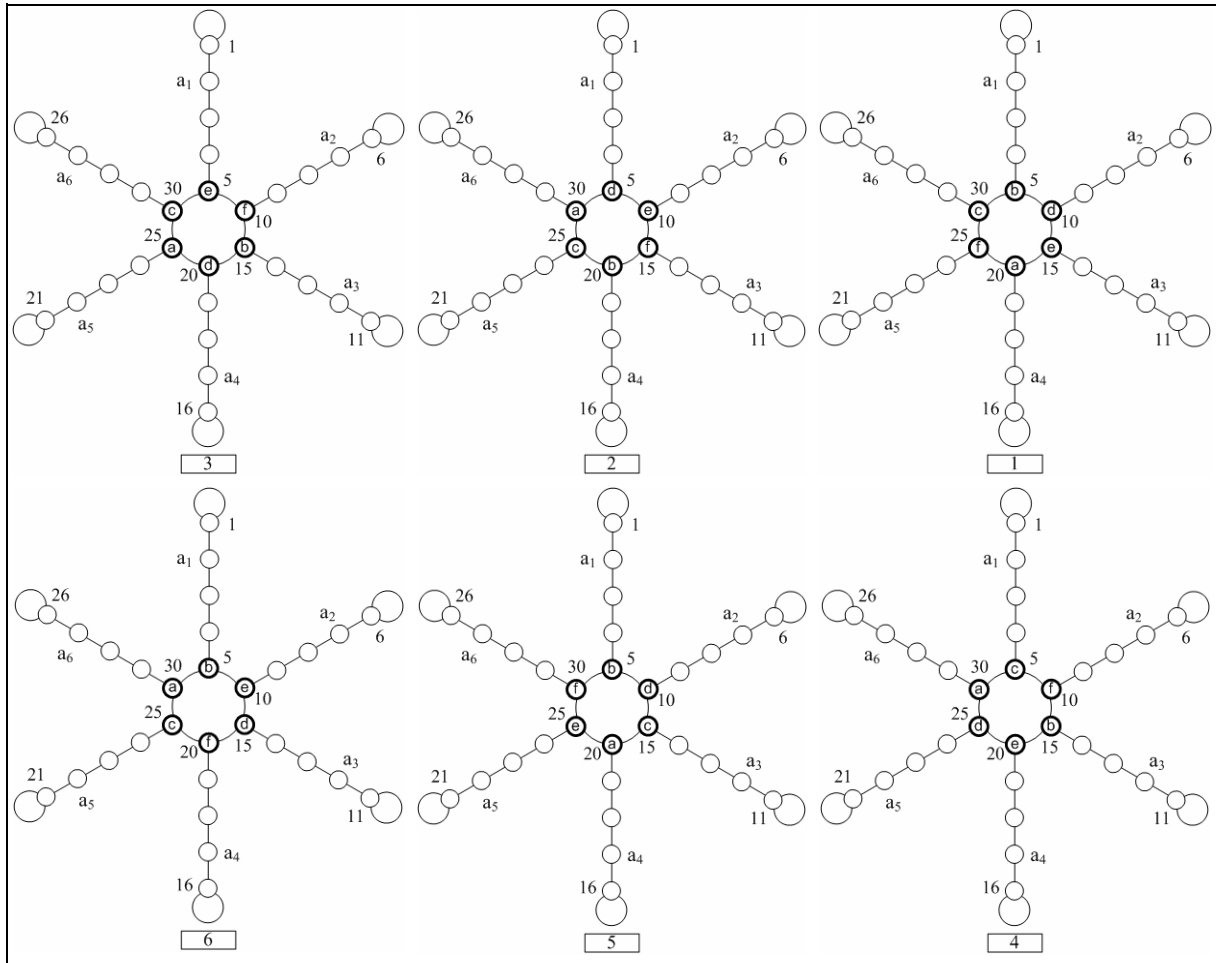
$$f(LA_i) = 1 / \text{Lenght of Specified Tour by } LA_i$$

### عملگرها:

از آنجائیکه در الگوریتم ترکیبی، هر کروموزوم به صورت یک آتوماتای یادگیر نمایش داده می شود، عملگرهای جابجایی و جهش مشابه عملگرهای سنتی ژنتیک نیستند.

الف) عملگر انتخاب<sup>۶</sup>: برای انتخاب آتوماتاهای یادگیر (کروموزوم ها) برای عملگرهای جهش و ترکیب می توان از یکی از روشهای رتبه بندی، سازوکار چرخ رولت و یا Tournament استفاده کرد.

6 Fitness Function  
7 Selection Operator



شکل ۳- جمعیت اولیه برای گراف ۱

#### Procedure Crossover ( $LA_1, LA_2$ )

##### Begin

Generate two random numbers  $r_1$  and  $r_2$  between 1 to  $n$

$r_1 = \text{Random} * n$ ;  $r_2 = \text{Random} * n$ ;

$r_1 = \text{Min}(r_1, r_2)$ ;  $r_2 = \text{Max}(r_1, r_2)$

**for**  $i = r_1$  **to**  $r_2$  **do**

**if** ( $J_i(LA_1) < J_i(LA_2)$ ) **then**

$j = \text{Action of } LA_2 \text{ where}$

$LA_2.\text{Object}(LA_2.\text{Action}(j)) = LA_1.\text{Object}(LA_1.\text{Action}(i))$ ;

$\text{Swap}(LA_2.\text{Object}(LA_2.\text{Action}(i)), LA_2.\text{Object}(LA_2.\text{Action}(j)))$ ;

**end if**

**else**

$j = \text{Action of } LA_1 \text{ where}$

$LA_1.\text{Object}(LA_1.\text{Action}(j)) = LA_2.\text{Object}(LA_2.\text{Action}(i))$ ;

$\text{Swap}(LA_1.\text{Object}(LA_1.\text{Action}(i)), LA_1.\text{Object}(LA_1.\text{Action}(j)))$ ;

**end else**

**end for**

##### End Procedure

//  $J_i(LA_k) = (\text{Weight of Edge } (i-1, i) \text{ in } LA_k + \text{Weight of Edge } (i, i+1) \text{ in } LA_k) / 2 - (\text{Length of Specified Tour By } LA_k / n)$ ;

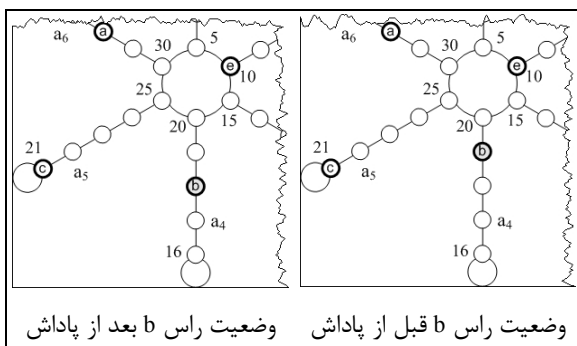
شکل ۴- شبه کد عملگر جابجایی (New Crossover)

به عنوان مثال فرض کنید که آتوماتاهای  $LA_2$  و  $LA_5$  از جمعیت تشکیل شده قبل به عنوان والد انتخاب شوند. با انتخاب تصادفی دو محل  $a_2$  و  $a_3$ ، مجموعه جابجایی  $\{a_2, a_3\}$  حاصل می شود و در نهایت مطابق شکل ۵ با

ب) عملگر ترکیب یا جابجایی<sup>۸</sup> برای انجام دادن این عملگر می توان از یکی از روشهای Partially Mapped Crossover، Ordered Crossover، Cycle Crossover و New Crossover استفاده کرد. در اینجا فقط روش پیشنهادی یعنی روش New Crossover توضیح داده می شود. در این روش دو کروموزوم والد انتخاب شده و به صورت تصادفی دو ژن  $i$  و  $j$  در یکی از دو کروموزوم والد انتخاب می شوند. سپس همین دو ژن در کروموزوم والد دیگر نیز انتخاب می شوند. مجموعه ژنهای با شماره های بین  $i$  و  $j$  را مجموعه جابجایی می نامیم. سپس ژن های هم شماره در دو مجموعه جابجایی با یکدیگر جابجا می شوند. با این عمل دو کروموزوم جدید حاصل می شوند که اصطلاحاً فرزندان آتوماتای والد خوانده می شوند. در شکل ۴ شبه کد این عملگر نشان داده شده است.

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 اتوماتای یادگیر متفاوت می باشد.

به عنوان مثال در اتوماتای با اتصالات مشابه اتوماتای ستلین، اگر راس  $b$  در مجموعه وضعیت های  $\{16, 17, 18, 19, 20\}$  قرار داشته باشد و میانگین هزینه یالهای ورودی و خروجی به راس  $b$  (هزینه یال ورودی به راس  $b$  + هزینه یال خروجی از راس  $b$  تقسیم بر ۲) از مقدار آستانه (مقدار آستانه بصورت تطبیقی مشخص می گردد و مقدار آن در هر لحظه برابر است با نسبت هزینه کل تور به تعداد راسها) کوچکتر باشد به این راس پاداش داده می شود و به سمت وضعیت های داخلی تر این اقدام حرکت می کند. اگر راس  $b$  در داخلی ترین وضعیت (وضعیت شماره 16) قرار داشته باشد و پاداش بگیرد در همان وضعیت باقی می ماند. نحوه حرکت چنین راسی در شکل ۶ نشان داده شده است.



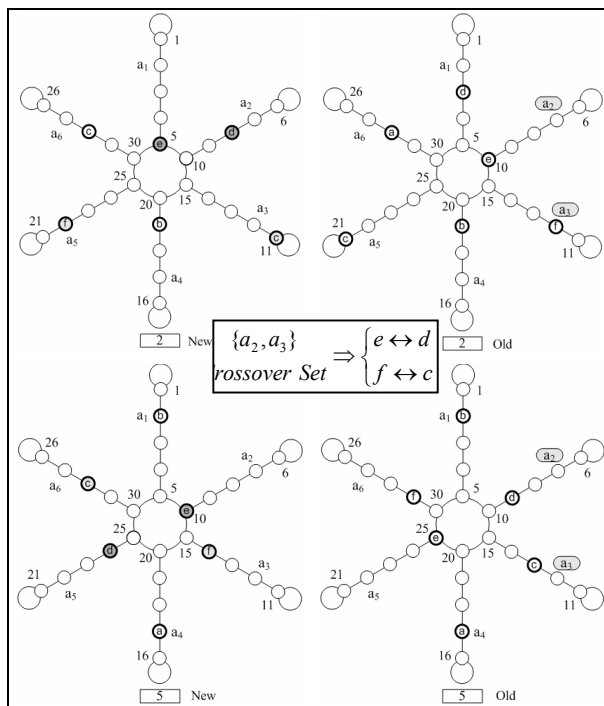
شکل ۶- نحوه پاداش دادن به راس  $b$

اگر میزان برازندگی یک راس از مقدار آستانه بزرگتر باشد در اینصورت تور برقرار شده مناسب نبوده و این راس جریمه می شود. در شکل ۷ شبه کد عملگر جریمه اقدام  $u$  از اتوماتای LA با اتصالات مشابه اتوماتای ستلین نشان داده شده است. نحوه حرکت چنین راسی برای دو حالت مختلف در زیر آمده است.

(الف) راس در وضعیتی غیر از وضعیت مرزی قرار داشته باشد: جریمه نمودن این راس سبب کم اهمیت شدن این راس شده و راس به سمت وضعیت های مرزی حرکت می کند.

(ب) راس در وضعیت مرزی قرار داشته باشد: در این حالت راسی از گراف را پیدا می کنیم بطوریکه اگر در جایگشت مربوطه جای دو راس عوض شوند بیشترین کاهش در هزینه

جایجایی اقدام های متناظر در فاصله جایجایی، دو کروموزوم جدید حاصل می شود.



شکل ۵- نحوه انجام عملگر جایجایی (New Crossover)

(پ) عملگر جهش<sup>۹</sup>: برای انجام دادن این عملگر می توان از یکی از روشهای Insertion Mutation, Swap Mutation و Inversion Mutation با جایگشت ها مناسب هستند استفاده کرد. به عنوان مثال در روش Swap Mutation، دو اقدام (ژن) از یک اتوماتا (کروموزم) به صورت تصادفی انتخاب شده و جایجا می شوند.

(ت) عملگر جریمه و پاداش<sup>۱۰</sup>: از آنجائیکه هر کروموزوم به صورت یک اتوماتای یادگیر نشان داده شده است، در هر یک از اتوماتاها پس از بررسی میزان برازندگی یک ژن (راس یا اقدام) که به صورت تصادفی انتخاب می شود، آن ژن پاداش یا جریمه می شود. در اثر پاداش دادن یا جریمه کردن یک ژن، وضعیت ژن در مجموعه وضعیت های اقدام مربوطه، تغییر می کند. اگر ژنی در وضعیت مرزی یک اقدام قرار داشته باشد، جریمه شدن آن باعث تغییر اقدام آن و در نتیجه باعث ایجاد جایگشت جدیدی می شود. نرخ این عملگر باید پایین باشد زیرا این عملگر، یک عملگر

9 Mutation Operator

10 Penalty and Reward

افزایش نیابد.

در شکل ۹ شبه کد الگوریتم ترکیبی برای حل مساله مرتب سازی ترتیبی آورده شده است.

```

Function SOP_Solver(G) : SOP_Tour
Begin
  n = Size of Population; // n = |VG|
  Create the initial population LA1 ... LAn;
  EvalFitness();
  while( All (Length of Specified Tour By LAi > Constant-Value) ) do
    NewLA1 = NewLA2 = LA with minimum Value of Tour-Lenght;
    for i = 2 to n do
      Select LA1; Select LA2 ;
      if (Random > 1 - CrossoverRate) then
        Crossover ( LA1, LA2 );
      if (Random > 1 - MutationRate) then
        Mutation ( LA1 ); Mutation ( LA2 );
      NewLAi+1 = LA1;
      NewLAi+2 = LA2 ;
      i=i+2;
    end for
    for i = 0 to n do
      LAi = NewLAi;
      u = Random *n;
      if ( Ju( LAi ) < threshold Threshold(LAi ) ) then
        Reward(LAi , u );
      else
        Penalize(LAi , u );
      end for
      EvalFitness();
    end while
    ModificationProcess(); //for observe precedence constraints
End Function
//Threshold(LAi) = Lenght( Specified Tour by LAi ) / |VG|;
//Ju(LAi) = (lenght of edge (u-1,u) in LAi + lenght of edge (u,u+1) in LAi) / 2;

```

شکل ۹- شبه کد الگوریتم ترکیبی برای حل مساله مرتب سازی ترتیبی

## ۵- نتایج آزمایش ها

در این بخش نتایج آزمایشی الگوریتم های حل مساله مرتب سازی ترتیبی که براساس اتوماتای یادگیر، الگوریتم ژنتیک و الگوریتم ترکیبی پیاده سازی شده اند، نشان داده شده است. این نتایج بهبود قابل توجه الگوریتم ترکیبی را نسبت به روشهای مبتنی بر اتوماتای یادگیر و الگوریتم ژنتیک نشان می دهد. در آزمایش های انجام گرفته اندازه گراف ها (گراف ها از TSPLIB انتخاب شده اند) از ۲۲ تا ۲۸۰ راس و تعداد تکرارها از ۵۰ تا ۵۰۰ تکرار در نظر گرفته شده است. در الگوریتم ترکیبی و اتوماتای یادگیر عمق های ۱، ۴، ۷، ۱۰ و ۱۵ آزمایش شده اند. در الگوریتم ترکیبی و الگوریتم ژنتیکی روش Swap Mutation با نرخ ۲۵٪ و روش رتبه بندی برای انتخاب کروموزمها استفاده شده است و همچنین سائز جمعیت برابر با تعداد نودهای گراف در نظر گرفته شده است.

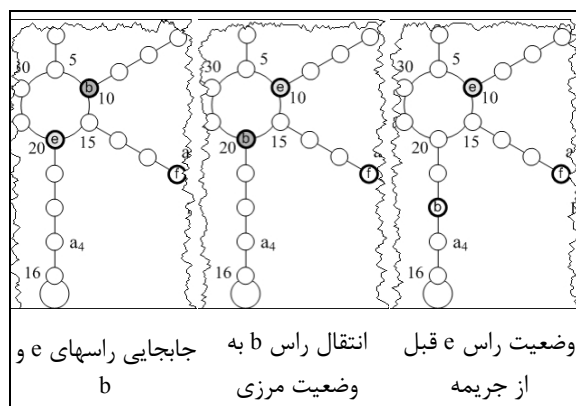
تور حاصل گردد. در اینصورت اگر راس پیدا شده در وضعیت مرزی قرار داشته باشد جای دو راس عوض می شود و در غیر اینصورت ابتدا راس مشخص شده به وضعیت مرزی اقدام خود منتقل و سپس جابجایی صورت می پذیرد. نحوه حرکت چنین راسی در شکل ۸ نشان داده شده است.

```

Procedure Penalize( LA, u )
  repeat
    for u = 1 to n do
      if (LA.State(U)) mod N <> 0 then
        Inc(LA.State(U));
      end for
    until at least one node appears in the boundary state
    bestTourLenght = ∞;
    for U = 1 to n do
      Create permutation LA' from LA by swapping u and U
      if Lenght(Specified Tour by LA') < bestTourLenght then
        bestTourLenght = Lenght(Specified Tour by LA');
        bestNode = U;
      end if
    end for
    LA.State(bestNode) = LA.Action(bestNode)*N;
    LA.State(u) = LA.Action(u)*N;
    Swap(LA.State(u),LA.State(bestNode));
  End Procedure

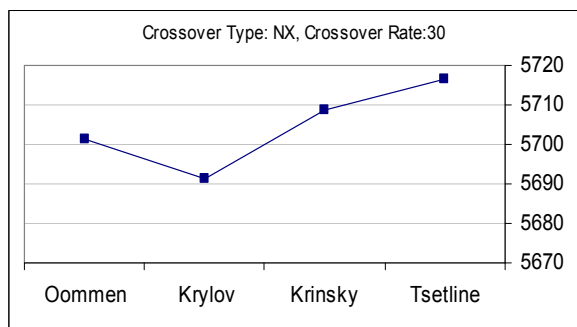
```

شکل ۷- شبه کد عملگر جریمه اقدام u از اتوماتای LA با اتصالات مشابه اتوماتای ستلین

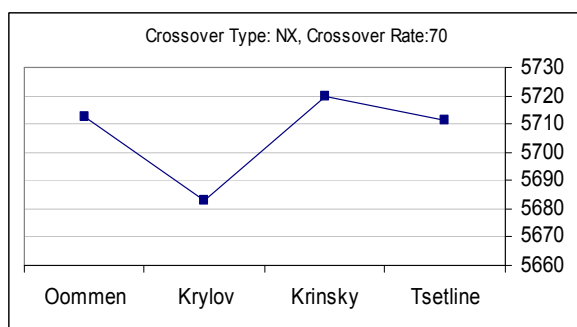


شکل ۸- نحوه جریمه کردن راسی که در وضعیت مرزی قرار دارد

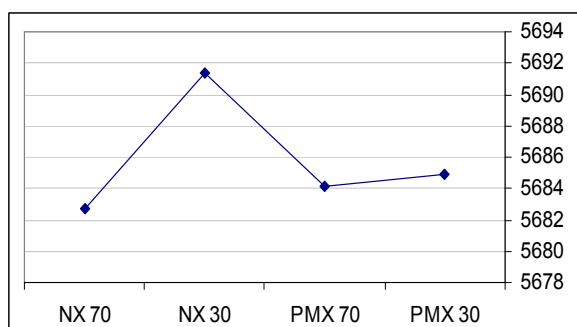
با توجه به اینکه جایگشت های تولید شده بعد از اعمال عملگرهای ژنتیکی (ترکیب، جهش، پاداش و جریمه) ممکن است محدودیت های اولویتی در بین رئوس را ارضاء نکنند، بنابراین این جایگشت ها باید بررسی شده و در صورت لزوم اصلاح شوند. بدلیل اینکه اصلاح جایگشت های تولید شده برای ارضاء محدودیت های اولویتی بعد از هر عملگر ژنتیکی هزینه بر می باشد، بنابراین فقط در انتهای الگوریتم ترکیبی تور بدست آمده برای ارضاء محدودیت های اولویتی اصلاح می شود. فرآیند اصلاح زمان چند جمله ای دارد و باید طوری اعمال شود که طول (هزینه) تور، تا حد ممکن



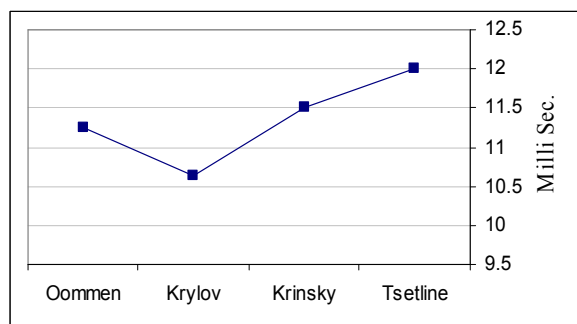
نمودار ۴- میانگین طول (هزینه) تور بدست آمده از الگوریتم ترکیبی با روش ترکیب New Crossover و نرخ ترکیب ۳۰



نمودار ۵- میانگین طول (هزینه) تور بدست آمده از الگوریتم ترکیبی با روش ترکیب New Crossover و نرخ ترکیب ۷۰

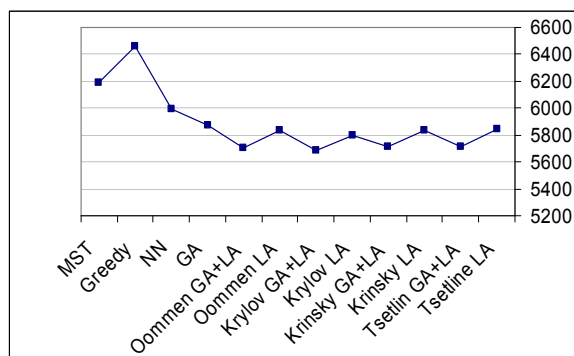


نمودار ۶- میانگین طول (هزینه) تور بدست آمده توسط الگوریتم ترکیبی مبتنی بر اتوماتای کرایلو با اعمال روشها و نرخ های ترکیب مختلف

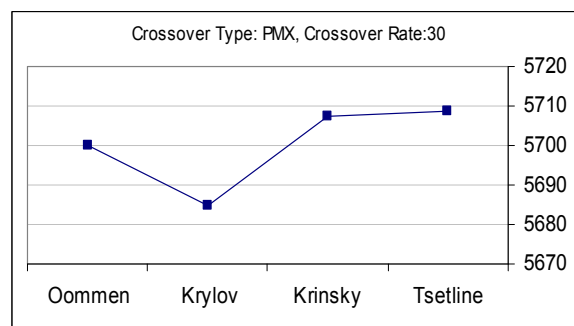


نمودار ۷- میانگین زمان لازم برای الگوریتم ترکیبی مبتنی بر اتوماتاهای ستلین، کرینسکی، کرایلو و اومن

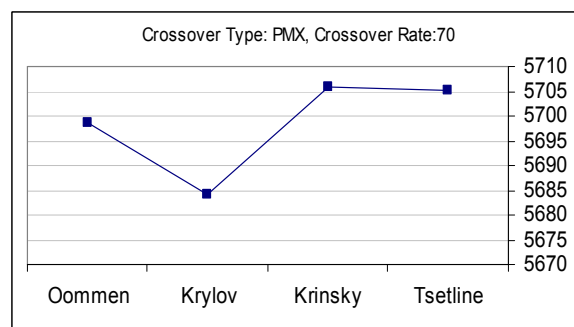
در نمودارهای ۱ الی ۷ مقایسه الگوریتم ترکیبی با سایر الگوریتم های حل مساله مرتب سازی ترتیبی بطور خلاصه آورده شده است. همانطور که از نتایج معلوم است، الگوریتم ترکیبی مبتنی بر اتوماتای کرایلو با روش ترکیب New Crossover و نرخ ترکیب ۷۰ بهتر از بقیه الگوریتم ها و سایر روشها و نرخ های ترکیب هم از لحاظ زمان اجرا و هم از لحاظ طول تور بدست آمده عمل می کند.



نمودار ۱- میانگین طول (هزینه) تور بدست آمده از الگوریتمهای مختلف



نمودار ۲- میانگین طول (هزینه) تور بدست آمده از الگوریتم ترکیبی با روش ترکیب Partially Mapped Crossover و نرخ ترکیب ۳۰



نمودار ۳- میانگین طول (هزینه) تور بدست آمده از الگوریتم ترکیبی با روش ترکیب Partially Mapped Crossover و نرخ ترکیب ۷۰

## ۶- نتیجه گیری و پیشنهادها

گراف ها، بویژه گراف های برجسب دار، ابزار های قدرتمند و پراستفاده ای هستند که به طور گسترده در کاربردهای کامپیوتر مورد استفاده قرار می گیرند. یکی از مسائل بسیار مهم در تئوری گراف ها، پیدا کردن تور مرتب سازی ترتیبی می باشد. محققان بیش از دو دهه بر روی این مساله کار کرده اند، ولی با توجه به این حقیقت که هنوز الگوریتمی از درجه چند جمله ای برای حل این مساله وجود ندارد، پژوهش ها در این زمینه همچنان ادامه دارد. با استفاده از روشهای جستجوی مناسب و ترکیب آنها، می توان الگوریتم های بهینه برای این مساله پیدا نمود. همچنین با خوشه بندی گره های گراف و اجرای الگوریتم ترکیبی بر روی هر خوشه بطور مستقل، می توان به نتایج بهتری رسید و همچنین استفاده از الگوریتم ژنتیک چند جمعیتی می تواند نتایج را بهبود دهد.

## مراجع

- 3th Annual International Computer Society of Iran Computer Conference CSICC-98, 1999, Tehran, Iran, 417-428.
- [10] M. R. Meybodi, and H. Beigy, "New Class of Learning Automata Based Scheme for Adaptation of Backpropagation Algorithm Parameters", Proceedings of EUFIT-98, 7-10 Sep. 1998, Aachen, Germany, 339-344.
- [11] Sanjeev Arora, "Nearly Linear Time Approximation Schemes for Euclidean TSP and other Geometric Problems", January 1997, (added to TSPBIB on May 2, 1997).
- [12] P. Merz, and B. Freisleben, "Genetic Local Search for the TSP: New Results", in Proceedings of the 1997 IEEE.
- [13] E. Cantu-Paz, "A Survey of Parallel Genetic Algorithms", IlliGAL Reprint No. 97003, May 1997.
- [14] L.F. Escudero, and M.T. Ortuno, "On Due-Date Based Valid Cuts for the Sequential Ordering Problem", Vol. 5, No. 1, 1997, 159-165.
- [15] B. Freisleben, and P. Merz, "A Genetic Local Search Algorithm for Solving Symmetric and Asymmetric Traveling Salesman Problems", appeared in Proceedings of the 1996 IEEE International Conference on Evolutionary Computation, 1996, Nagoya, Japan, 616-621.
- [16] B. Freisleben, and P. Merz, "New Genetic Local Search Operators for the Traveling Salesman Problem", in Proceedings of the 4th Conference on Parallel Problem Solving from Nature - PPSN IV, (H.-M. Voigt, W. Ebeling, I. Rechenberg, H.-P. Schwefel, eds.), Vol. 1141 of Lecture Notes in Computer Science, 1996, 890-899.
- [17] M. Jünger, G. Reinelt, and G. Rinaldi, "The Traveling Salesman Problem", in Handbooks in Operations Research and Management Science, Vol. 7 (M.O. Ball, T. Magnanti, C.L. Monma, and G. Nemhauser, eds), Elsevier Science B.V., 1995, 225-330.
- [18] P. Moscato, and M.G. Norman, "An Analysis of the Performance of Traveling Salesman Heuristics on Infinite-Size Fractal Instances in the Euclidean Plane", Oct. 1994.
- [19] M. Grötschel, and O. Holland, "Solution of Large-Scale Symmetric Traveling Salesman Problems", Mathematical Programming 51, 1991, 141-202.
- [20] M. Padberg, and G. Rinaldi, "A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems", SIAM Review 33, 1991, 60-100.
- [21] B. J. Oommen, R. S. Valiveti, and J. R. Zgierski, "An Adaptive Learning Solution to the Keyboard Optimization Problem", IEEE Transaction On Systems, Man, And Cybernetics, Vol. 21, No. 6, 1991, 1608-1618.
- [22] D. E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Reading, MA, Addison-Wesley, 1989.
- [23] K. S. Narendra, and M. A. L. Thathachar, "Learning Automata: An Introduction", Prentice-hall, Englewood cliffs, 1989.
- [24] 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, 1988, 2-3.
- [25] A. A. Hashim, S. Amir, and P. Mars, "Application of Learning Automata to Data Compression", in Adaptive and Learning Systems, K. S. Narendra, Editor, New York, Plenum Press, 1986, 229-234.
- [26] P. Mars, K. S. Narendra, and M. Chrystall, "Learning Automata Control of Computer Communication Networks", Proceedings of Third Yale Workshop on Application of Adaptive Systems Theory, 1983, Yale University.
- [27] M. R. Meybodi, and S. Lakshmivarhan, "A Learning Approach to Priority Assignment in a Two Class M/M/1 Queuing System with Unknown Parameters", Proceedings of Third Yale Workshop on Applications of Adaptive System Theory, 1983, Yale University, 106-109.
- [28] F. Buseti, "Genetic Algorithm Overview".
- [29] <http://www.tsp.gatech.edu>.
- [1] میبیدی، محمد رضا و بیگی، حمید. "حل مساله تناظر گراف توسط اتوماتاهای یادگیر". دانشکده مهندسی کامپیوتر. دانشگاه صنعتی امیرکبیر. تهران. ایران. ۱۳۷۹.
- [۲] میبیدی، محمد رضا و رضاپور میرصالح، مهدی. "یک روش ترکیبی (GA+LA) برای حل مساله تناظر گراف". دانشکده مهندسی کامپیوتر. دانشگاه صنعتی امیرکبیر. تهران. ایران. ۱۳۸۲.
- [3] Bager Zarei, M. R. Meybodi, and Mortaza Abbaszadeh, "A Hybrid Method for Solving Traveling Salesman Problem", Proceedings of the 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007), IEEE Computer Society, 11-13 July 2007, Melbourne, Australia, 394-399.
- [4] D. S. Johnson, and L. A. McGeoch, "Experimental Analysis of Heuristics for the STSP", in the Traveling Salesman Problem and its Variations, G. Gutin and A. Punnen, Editors, Kluwer Academic Publishers, 2002, Boston, 369-443.
- [5] D. S. Johnson, G. Gutin, L. A. McGeoch, A. Yeo, W. Zhang, and A. Zverovich, "Experimental Analysis of Heuristics for the ATSP", in the Traveling Salesman Problem and its Variations, G. Gutin and A. Punnen, Editors, Kluwer Academic Publishers, 2002, Boston, 445-487.
- [6] D. S. Johnson, "A Theoretician's Guide to the Experimental Analysis of Algorithms", to appear in Proceedings of the 5th and 6th DIMACS Implementation Challenges, M. Goldwasser, D. S. Johnson, and C. C. McGeoch, Editors, American Mathematical Society, Providence, 2002.
- [7] J. Cirasella, D.S. Johnson, L.A. McGeoch, and W. Zhang, "The Asymmetric Traveling Salesman Problem: Algorithms, Instance Generators, and Tests", in Algorithm Engineering and Experimentation, Third International Workshop, ALNEX 2001, Lecture Notes in Computer Science, Vol. 2153, Springer, 2001, Berlin, 32-59.
- [8] K. Bryant, "Genetic Algorithms and the Traveling Salesman Problem", Thesis, 2000, Harvey Mudd College, Dept. of Mathematics.
- [9] H. Beigy, and M. R. Meybodi, "Optimization of Topology of Neural Networks Using Learning Automata", Proceedings of