مجله کامپیوتر و روباتیک ۱ (۱۳۸۷) ۳۷-۲۵

Computer & Robotics

بهینه‌سازی اجرای پرس‌وجوها در پایگاه داده‌های رابطه‌ای با الگوریتم تکاملی ترکیبی

کیوان اصغری[۱]، علی صفری‌ممقانی[۱]، فریبرز محمودی[۱،*]، محمدرضا میبدی[۲]

[۱] دانشکده مهندسی برق، رایانه و فن‌آوری اطلاعات دانشگاه آزاد اسلامی قزوین
[۲] دانشکده مهندسی کامپیوتر دانشگاه صنعتی امیر کبیر

دریافت مهر۱۳۸۶، تجدید نظر بهمن۱۳۸۶، پذیرش فروردین ۱۳۸۷

چکیده

بهینه‌سازی پرس‌وجوهای پایگاه‌داده‌ای، یکی از مسائل تحقیقاتی مشکل است. تکنیک‌های جستجوی جامع مانند: برنامه‌نویسی پویا برای پرس‌وجوهای با تعداد روابط کم، مناسب هستند ولی با افزایش تعداد روابط موجود در پرس‌وجو، بدلیل نیاز به مصرف زیاد حافظه و پردازش، استفاده از این روش‌ها مناسب نخواهند بود، بنابراین مجبوریم از روش‌های تصادفی و تکاملی استفاده کنیم. استفاده از روش‌های تکاملی بدلیل کارایی و قدرتمندی آنها، تبدیل به ناحیه تحقیقاتی مناسبی در زمینه بهینه‌سازی پرس‌وجو گردیده است. در این مقاله یک الگوریتم تکاملی ترکیبی برای حل مساله بهینه‌سازی ترتیب اجرای عملگرهای پیوند در پرس‌وجوهای پایگاه داده‌ای پیشنهاد شده است. این الگوریتم از دو روش الگوریتم‌های ژنتیکی و آتاماتاهای یادگیر بطور همزمان برای جستجو در فضای حالات مسئله استفاده می‌نماید. در این مقاله، نشان داده شده است که با استفاده همزمان از آتاماتاهای یادگیر و الگوریتم‌های ژنتیکی در فرایند جستجو، سرعت رسیدن به جواب افزایش پیدا کرده و از بدام افتادن الگوریتم در مینیمم‌های محلی جلوگیری می‌شود. نتایج آزمایش‌ها، برتری الگوریتم ترکیبی را نسبت به روش‌های مبتنی بر الگوریتم ژنتیکی و آتاماتای یادگیر نشان می‌دهد.

واژه‌های کلیدی: بهینه‌سازی پرس‌وجو، عملگر پیوند، آتاماتای یادگیر، الگوریتم ژنتیکی

۱. مقدمه

مدل داده‌ای رابطه‌ای توسط کاد [۱]، معرفی شده است و در سالهای اخیر، سیستم‌های پایگاه‌داده رابطه‌ای بعنوان استانداردی در انواع کاربردهای علمی و تجاری، شناخته می‌شوند. کار بر روی عملگر پیوند، بدلیل هزینه‌های ارزیابی بالای این عملگر، هدف اولیه بهینه‌سازهای پرس‌وجوی رابطه‌ای است. اگر پرس‌جوها در حالت محاوره‌ای باشند، شامل تعدادکمی رابطه خواهد بود که بهینه‌سازی این عبارات را می‌توان بوسیله یک جستجوی جامع انجام داد.

اما در صورتی که تعداد روابط بیش از ۵ یا ۶ رابطه باشد، تکنیک‌های جستجوی جامع، از لحاظ حافظه و زمان پرهزینه خواهند بود. پرس‌وجوهای با تعداد پیوند زیاد در سیستم‌های جدید مانند سیستم‌های مدیریت پایگاه داده استنتاجی[۱]، سیستم‌های خبره[۲]، سیستم‌های مدیریت پایگاه دادهای مهندسی

* نویسنده مکاتبه‌کننده: mahmoudi@itrc.ac.ir, Tel (281)3675782

1 Deductive
2 System Expert

(CAD/CAM)، سیستم‌های پشتیبانی تصمیم[3]، داده‌کاوی[4] و سیستم‌های مدیریت پایگاه‌داده‌های علمی و ... دیده می‌شوند. به هر حال سیستم‌های مدیریت پایگاه‌داده[5]، نیازمند بکارگیری تکنیک‌های بهینه‌سازی پرس‌وجو با هزینه کم برای مقابله با چنین پرس‌وجوهای پیچیده‌ای هستند. دسته‌ای از الگوریتم‌های جستجوی ترتیب مناسب برای اجرای عملگرهای پیوند، الگوریتم‌های قطعی می‌باشند که کل فضای حالات را بصورت کامل جستجو می‌کنند و بعضاً با بکارگیری روشهای مکاشفه‌ای[6]، این فضا را کاهش می‌دهند[۷]. یکی از این الگوریتم‌ها، روش برنامه‌نویسی پویاست[7] که اولین بار برای بهینه‌سازی ترتیب پیوند در System-R توسط سلینگر و همکارانش مطرح گردید[۹ ،۲]. مهمترین ایراد الگوریتم این است که با افزایش تعداد روابط موجود در پرس‌وجو، نیازمند مصرف زیاد حافظه و پردازنده می‌باشد. از دیگر الگوریتم‌های قطعی می‌توان به الگوریتم حداقل قابلیت انتخاب[8] [۷]، الگوریتم KBZ[۱۰] و الگوریتم AB [۱۱] اشاره کرد. جهت نشان دادن ناتوانی و ضعف الگوریتم‌های قطعی در مقابله با پرس‌وجوهای بزرگ، الگوریتم‌های دیگری به نام الگوریتم‌های تصادفی معرفی شده‌اند. الگوریتم‌های مطرح شده در این زمینه، الگوریتم بهبود مکرر[9] [۵، ۶، ۱۲]، الگوریتم نرم کردن شبیه‌سازی شده[10] [۵، ۱۲، ۱۳، ۱۴]، بهینه‌سازی دو مرحله‌ای[11][۱۲]، نرم کردن شبیه‌سازی شده گردشی[12][۸] و نمونه‌گیری تصادفی[13][۱۵] می‌باشد.

با توجه به طبیعت الگوریتم‌های تکاملی و اینکه در اکثر مواقع مقاوم و کاراتر می‌باشند و با درنظر گرفتن کارهای صورت گرفته در این زمینه، مناسب‌ترین گزینه برای حل این مسئله، استفاده از الگوریتم‌های تکاملی می‌باشد. اولین کار انجام شده بر روی مسئله بهینه‌سازی ترتیب پیوندها با استفاده از الگوریتم ژنتیکی، توسط بنت و همکارانش ارایه گردید[۳]. در حالت کلی الگوریتم بکار رفته توسط آنها در مقایسه با الگوریتم برنامه‌نویسی پویای بکار رفته برای System-R، در اکثر مواقع دارای هزینه کمتری می‌باشد. از ویژگی‌های دیگر این الگوریتم، قابلیت بکارگیری آن در معماری موازی می‌باشد. از دیگر کارهای صورت گرفته مبتنی بر الگوریتم ژنتیکی، روش ارایه شده توسط استین‌بران و همکارانش بوده است[۷] که روشهای کدگذاری و عملگرهای ژنتیکی متفاوتی را بکار گرفته‌اند. نمونه دیگر از الگوریتم‌های تکاملی بکار گرفته شده برای مسئله بهینه‌سازی پیوندها، روش برنامه‌نویسی ژنتیکی می‌باشد که توسط استیلگر و اسپیلیوپولو [۱۶] مطرح شده است. بهینه‌ساز ژنتیکی CGO نیز توسط مولرو و همکارانش در[۱۷] ارایه شده است. در این مقاله یک الگوریتم تکاملی ترکیبی برای حل مساله بهینه‌سازی ترتیب اجرای عملگرهای پیوند در پرس‌وجوهای پایگاه داده‌ای پیشنهاد کرده‌ایم. این الگوریتم از دو روش الگوریتم‌های ژنتیکی و آتاماتاهای یادگیر بطور همزمان برای جستجو در فضای حالات مسئله استفاده می‌نماید. نشان داده شده است که با استفاده همزمان از آتاماتاهای یادگیر و الگوریتم‌های ژنتیکی در فرایند جستجو، سرعت رسیدن به جواب افزایش پیدا کرده و از بدام افتادن الگوریتم در مینیمم‌های محلی جلوگیری می‌شود. نتایج آزمایش‌ها، برتری الگوریتم ترکیبی را نسبت به روش‌های مبتنی بر الگوریتم ژنتیکی و آتاماتای یادگیر نشان می‌دهد.

ادامه مقاله بدین صورت سازماندهی شده است که بخش دوم، مساله ترتیب عملگرهای پیوند در پرس‌وجوهای پیوندی را تعریف می‌کند. توضیح مختصری درباره آتاماتاهای یادگیر و الگوریتم‌های ژنتیکی در بخش ۳ آمده است. در بخش۴، الگوریتم ترکیبی پیشنهادی در این مقاله شرح داده می‌شود و در بخش ۵، نتایج آزمایش‌ها ارایه می‌گردد. بخش ۶ نیز شامل نتیجه‌گیری می‌باشد.

۲.تعریف مسئله

بهینه‌سازی پرس‌وجو، فعالیتی است که طی آن، یک طرح کارا برای اجرای پرس‌وجو (qep)[14]، تولید می‌گردد و یکی از مراحل اساسی در پردازش پرس‌وجو می‌باشد. در این مرحله، سیستم مدیریت پایگاه داده از بین تعدادی طرح اجرا، بهترین طرح را بر می‌گزینند. بگونه‌ای که اجرای پرس‌و-

3 Decision Support system- DSS
4 Data Mining
5 Database Management System-DBMS
6 Heuristic
7 Dynamic Programming
8 Minimum Selectivity
9 Iterative Improvement -II
10 Simulated Annealing
11 Two –Phase OPtimization
12 Toured Simulated Annealing
13 Random Sampeling

14 Query Execution Plan

جوی داده شده توسط کاربر با استفاده از این طرح دارای کمترین هزینه، بویژه هزینه عملیات ورودی/ خروجی[15] می-باشد. ورودی بهینه ساز، نمایش داخلی پرس‌وجوی Q می‌باشد که توسط کاربر به سیستم مدیریت پایگاه‌داده، وارد شده است. هدف کلی از بهینه‌سازی پرس‌وجو، انتخاب کاراترین طرح اجرایی برای دستیابی به داده مناسب و پاسخ به پرس‌وجوی داده شده می‌باشد. به بیان دیگر، در صورتی که مجموعه همه طرح‌های اجرایی اختصاص داده شده برای پاسخ به پرس وجوی Q را با S نشان دهیم، هر عضو qep متعلق به مجموعه S دارای هزینه (cost(qep می‌باشد (این هزینه شامل زمان پردازشی و زمان ورودی/خروجی می‌باشد). هدف هر الگوریتم بهینه‌سازی یافتن عضوی مانند qep0 متعلق به مجموعه S می‌باشد، به نحوی که[۳]:

$$cost(qep_0) = min_{qep \in S} cost(qep)$$

طرح اجرایی برای پاسخ به یک پرس‌وجو، دنباله‌ای از عملگرهای جبری رابطه‌ای می‌باشد که بر روی روابط پایگاه داده اعمال می‌شود و جواب لازم برای آن پرس‌وجو را تولید می‌کند. از بین عملگرهای رابطه‌ای موجود، پردازش و بهینه سازی عملگر پیوند[16] که بوسیله نماد ∞ نمایش داده می‌شود، جزو مشکل‌ترین عملگرها می‌باشد. اساسا، عملگر پیوند دو رابطه را بعنوان ورودی می‌گیرد و تاپل‌های آنها را یک به یک بر اساس معیار مشخصی، ترکیب کرده و یک رابطه جدید را به عنوان خروجی تولید می‌نماید. از آنجا که عملگر پیوند دارای خاصیت انجمنی[17] جابجایی[18] می‌باشد، تعداد طرح‌های اجرایی موجود برای پاسخ‌دهی به یک پرس‌وجو با افزایش تعداد پیوندهای بین روابط، به صورت نمایی رشد می‌کند. علاوه بر این موارد، معمولاً یک سیستم مدیریت پایگاه داده، انواع روش‌های پیاده‌سازی عملگر پیوند را برای پردازش پیوندها و انواع اندیس‌ها را برای دسترسی به روابط پشتیبانی می‌کند. بطوریکه این موارد، گزینه‌های لازم برای پاسخ‌دهی به یک پرس‌وجو را هر چه بیشتر می‌کند. گرچه تمامی طرح‌های اجرایی موجود برای پاسخ به یک پرس‌وجوی مشخص، دارای خروجی یکسان می‌باشند اما از آنجا که کاردینالیتی روابط میانی ایجاد شده یکسان نیستند، طرح‌های اجرایی بوجود آمده دارای هزینه متفاوتی خواهند بود. بنابراین، انتخاب ترتیب مناسب برای اجرای عمل پیوند در هزینه کلی، تاثیرگذار است. مسئله بهینه سازی پرس‌وجو که اغلب از آن بعنوان مسئله انتخاب ترتیب مناسب برای اجرای عملگرهای پیوند[19] یاد می‌شود، یک مسئله NP-hard می‌باشد[۴].

۳.آتاماتاهای یادگیر و الگوریتم‌های ژنتیکی

یادگیری در آتاماتاهای یادگیر، انتخاب یک اقدام بهینه از میان یک مجموعه از اقدام‌های مجاز آتاماتا می‌باشد. این اقدام روی یک محیط تصادفی اعمال می‌شود و محیط به این اقدام آتاماتا بوسیله یک پاسخ تصادفی از مجموعه پاسخ‌های مجاز، جواب می‌دهد. پاسخ محیط بصورت آماری به اقدام آتاماتا وابسته است. اصطلاح محیط شامل اجتماع تمام شرایط خارجی و تاثیرات آنها روی عملکرد آتاماتا می‌باشد. برای اطلاعات بیشتر درباره آتاماتاهای یادگیر می‌توان به [۱۸] مراجعه نمود. آتاماتاهای یادگیر دارای کاربردهای فراوانی می‌باشد. بعضی از این کاربردها عبارتند از: مسیریابی در شبکه‌های ارتباطی [۲۰]، فشرده سازی تصاویر [۲۱]، شناسایی الگو[۲۲]، برنامه‌ریزی فرآیندها در یک شبکه کامپیوتری[۲۳]، تئوری صف [۲۴]، کنترل دسترسی در شبکه‌های انتقال ناهمزمان[۲۴]، کمک به آموزش شبکه‌های عصبی [۲۵]، دسته‌بندی و افراز اشیاء [۲۶] و پیدا کردن ساختار بهینه برای شبکه‌های عصبی [۲۷]. برای یک پرس‌وجو با n عملگر پیوند، $n!$ طرح اجرایی مختلف وجود دارد و در صورتیکه از آتاماتاهای یادگیر برای پیدا کردن طرح اجرایی بهینه استفاده شود آتاماتا باید $n!$ اقدام داشته باشد که تعداد زیاد اقدام‌ها، سرعت همگرایی آتاماتا را کم می‌کند به همین جهت، آتاماتای مهاجرت اشیاء[20] توسط اومن[21] و ما[22] پیشنهاد شده است[۱۸]. الگوریتم‌های ژنتیکی که برمبنا ی ایده تکامل در طبیعت عمل می‌کنند، برروی جمعیتی از راه‌حلهای بالقوه به جستجوی راه‌حل نهایی می‌پردازند. در هر نسل، بهترین‌های آن نسل انتخاب می‌شوند و پس از زاد و ولد، مجموعهٔ

15 Input/Output- I/O
16 Join
17 Associative
18 Commutative

19 Join Ordering Problem
20 Object Migrating Automata (OMA)
21 Oommen
22 Ma

جدیدی از فرزندان را تولید می‌کنند. در این فرایند افراد مناسبتر با احتمال بیشتری در نسلهای بعد باقی خواهند ماند[۱۹]. برای اطلاعات بیشتر درباره الگوریتمهای ژنتیکی می‌توان به [۲۸و۲۹] مراجعه نمود.

۴.الگوریتم ترکیبی پیشنهادی برای حل مسئله ترتیب عملگرهای پیوند

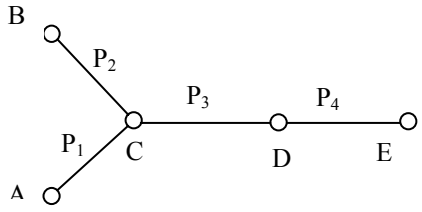
با ترکیب الگوریتم ژنتیکی و آتاماتای یادگیر و تلفیق مفاهیم ژن، کروموزوم، اقدام و عمق، سابقه تاریخی تکامل راه حل مساله، به شکل کارا استخراج شده و در روند جستجو مورد استفاده قرار می‌گیرد. خاصیت مهم الگوریتم ترکیبی، مقاومت آن در مقابل تغییرات سطحی جواب‌هاست، به عبارتی دیگر تعادلی انعطاف پذیر بین کارایی الگوریتم ژنتیکی و پایداری آتاماتای یادگیر در الگوریتم ترکیبی وجود دارد. خودترمیمی، تولید مثل، جریمه 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 الگوریتم‌های ژنتیکی کلاسیک، از کدگذاری دودویی یا نمایش جایگشت طبیعی برای کروموزوم‌ها استفاده نمی‌شود. هر کروموزوم توسط یک آتاماتای یادگیر از نوع مهاجرت اشیا نشان داده می‌شود. بطوریکه هر کدام از ژنها در کروموزوم به یکی از اقدامهای آتاماتا نسبت داده می‌شود و در یک عمق مشخصی از آن اقدام قرار می‌گیرد. در این آتاماتا $\underline{\alpha} = \{\alpha_1, \alpha_2, \dots, \alpha_k\}$ مجموعه اقدام‌های مجاز برای آتاماتای یادگیر است. این آتاماتا k اقدام دارد (تعداد اقدام‌های این آتاماتا با تعداد اعمال پیوند برابر است). اگر پیوند شماره u از پرس‌وجوی داده شده در اقدام m قرار گرفته باشد، در اینصورت پیوند u، m امین پیوندی از پرس‌وجو خواهد بود که اجرا می‌شود. $\underline{\phi} = \{\phi_1, \phi_2, \dots, \phi_{KN}\}$ مجموعه وضعیت‌ها و N عمق حافظه برای آتاماتا می‌باشد. مجموعه وضعیت‌های این آتاماتا به k زیر مجموعه $\{\phi_1, \phi_2, \dots, \phi_N\}$ و $\{\phi_{N+1}, \phi_{N+2}, \dots, \phi_{2N}\}$ و ... و $\{\phi_{(K-1)N+1}, \phi_{(K-1)N+2}, \dots, \phi_{KN}\}$ افراز می‌شود و عملگرهای پیوند بر اساس این که در کدام وضعیت قرار داشته باشند دسته‌بندی می‌گردند. اگر پیوند شماره u از پرس‌وجو در مجموعه وضعیت‌های $\{\phi_{(j-1)N+1}, \phi_{(j-1)N+2}, \dots, \phi_{jN}\}$ قرار داشته باشد در اینصورت پیوند u امین پیوندی خواهد بود که اجرا می‌شود. در مجموعه وضعیت‌های اقدام j به وضعیت $\phi_{(j-1)N+1}$ وضعیت داخلی و به وضعیت ϕ_{jN}، وضعیت مرزی گفته می‌شود. گره‌ای که در وضعیت $\phi_{(j-1)N+1}$ قرار دارد گره با اهمیت بیشتر و گره‌ای در وضعیت ϕ_{jN} گره با اهمیت کمتر نامیده می‌شود. در اثر پاداش دادن یا جریمه کردن یک اقدام، وضعیت پیوند وابسته به آن اقدام، تغییر می‌کند که بعد از ایجاد شدن چند نسل از آتاماتاها توسط الگوریتم ژنتیکی می‌توان به یک جایگشت بهینه رسید که همان بهترین راه حل مسئله است. اگر پیوندی در وضعیت مرزی یک اقدام قرار داشته باشد، جریمه شدن آن باعث تغییر اقدامی که پیوند به آن وابسته است، می‌شود و در نتیجه باعث ایجاد جایگشت جدیدی می‌گردد. حال پرس‌وجوی زیر را در نظر بگیرید:

(A∞C) and (B∞C) and (C∞D) and (D∞E)

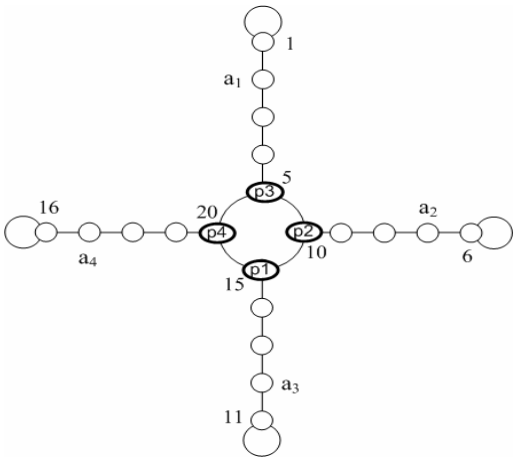
هر عمل پیوند دارای یک شرط برای انجام پیوند می‌باشد که جهت سادگی نمایش حذف شده است و مشخص می‌کند که کدام تاپلها از روابط پیوند یافته در نتیجه ظاهر می‌شوند. پرس وجوی بالا می‌تواند به صورت گراف شکل ۱ نمایش داده شود. حروف بزرگ را برای نشان دادن رابطه‌ها و pi را برای نمایش عملگرهای پیوند بکار می‌بریم.



شکل۱. گراف پرس‌وجو

مجموعه عملگرهای پیوند p1,p2,p3,p4 را در نظر می‌گیریم تا جایگشتی از ترتیب اجرای عملیات پیوند را با آتاماتای مهاجرت اشیاء مبتنی بر اتصالات آتاماتای ستلین نشان دهیم. این آتاماتای یادگیر دارای ۴ اقدام{a1,a2,a3,a4} (به تعداد پیوندهای پرس‌وجو) و عمق ۵ می‌باشد. مجموعه وضعیت‌های {۱،۶،۱۱،۱۶،۲۱،۲۶}

وضعیتهای داخلی و مجموعه وضعیتهای {۵،۱۰،۱۵،۲۰،۲۵،۳۰} وضعیتهای مرزی آتاماتای یادگیر هستند. در ابتدا هر یک از پیوندهای پرس‌وجو در وضعیت مرزی اقدام مربوطه قرار دارند. در الگوریتم ترکیبی هر ژن از کروموزوم معادل یک اقدام آتاماتا می‌باشد و لذا می‌توان در ادامه این دو واژه را به جای یکدیگر بکار برد. این آتاماتای یادگیر(کروموزوم) دارای ۴ اقدام (ژن) می‌باشد و هر اقدام دارای ۵ وضعیت داخلی می‌باشد. فرض کنید در اولین جایگشت ابتدا پیوند ۳ و سپس پیوند ۲ و پیوند ۱ و در انتها پیوند ۴ ترتیب اجرای عملگرهای پیوند ما باشند. نحوه نمایش این ترتیب اجرا با آتاماتای مهاجرت اشیای مبتنی بر اتصالات آتاماتای ستلین[۲۳] بصورت شکل ۲ است. در ابتدا هر یک از پیوندها در وضعیت مرزی اقدام مربوطه قرار دارند.

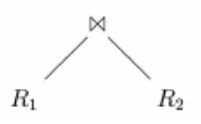


شکل۲. نمایش جایگشت پیوندهای (p3,p2,p1,p4) توسط آتاماتای یادگیر مبتنی بر اتصالات آتاماتای ستلین

تابع برازندگی: در الگوریتم.های ژنتیکی، تابع برازندگی شاخص زنده ماندن کروموزوم‌ها است. هدف جستجوی ترتیب بهینه پیوندهای پرس‌وجو، یافتن جایگشتی از عملگرهای پیوند می‌باشد که کل هزینه اجرای پرس‌وجو در این جایگشت کمینه باشد. نکته‌ای که در محاسبه تابع برازندگی مهم است، تعداد رجوعات به دیسک می‌باشد. لذا می‌توانیم تابع برازندگی F را برای یک طرح اجرایی qep بصورت زیر تعریف نماییم:

F (qep)=1/تعداد رجوعات به دیسک

برای محاسبه تعداد رجوعات به دیسک (هزینه) یک طرح اجرایی، هر طرح اجرایی را متناسب با یک درخت پردازشی در نظر می‌گیریم. هزینه هر گره بصورت بازگشتی (پایین به بالا و از راست به چپ) با جمع هزینه‌های بدست آمده از دو گره فرزند و هزینه لازم جهت پیوند آنها برای بدست آوردن نتیجه نهایی، محاسبه می‌شود[۷]. برای نمونه، پیوند $R_1 \infty R_2$ را در نظر می‌گیریم:



در این صورت هزینه ارزیابی برابر با مقدار زیر خواهد بود:

$$C_{total} = C(R_1) + C(R_2) + C(R_1 \infty R_2)$$

که در آن $C(R_k)$ هزینه بدست آوردن گره فرزند می‌باشد و از رابطه زیر قابل محاسبه است:

$$C(R_k) := \begin{cases} 0 & \text{اگر Rk یک رابطه پایه‌ای در پرس‌وجو باشد} \\ C(R_i \infty R_j) & \text{اگر } R_k = R_i \infty R_j \text{ یک نتیجه میانی باشد} \end{cases}$$

بعنوان یک حالت خاص اگر R_1 و R_2 هر دو رابطه پایه‌ای باشند، هزینه C_{total} برابر با $C(R_1 \infty R_2)$ خواهد بود و باتوجه به اینکه نوع پیاده‌سازی بکار رفته برای عمل پیوند در پرس‌وجوها را از نوع حلقه‌های تودر تو[۲۴] در نظر گرفته‌ایم، هزینه محاسبه عمل پیوند در این نوع پیاده‌سازی برابر با $C(R_1 \infty R_2) = b_{R_1} + b_{R_2}$ است که b_{R_k} تعداد بلاکهای رابطه R_k می‌باشد.

عملگرها: با توجه به اینکه در الگوریتم ترکیبی هر کروموزوم بصورت یک آتاماتای یادگیر نمایش داده می‌شود عملگرهای جابجایی و جهش با مشابه این عملگرها در الگوریتمهای ژنتیکی متفاوت می‌باشند.

عملگر انتخاب[۲۵]: عملگر انتخاب بکار رفته برای این الگوریتم، از نوع چرخ رولت[۲۶] می‌باشد.

عملگر ترکیب یا جابجایی[۲۷]: عملگرهای جابجایی پیاده‌سازی شده برای الگوریتم ترکیبی عبارتند از:

Ordered: این روش ترتیب نسبی ژنها در کروموزوم را تا

23 Tsetlin

24 Nested Loop
25 Selection
26 roulette wheel
27 Crossover

حد ممکن حفظ می‌کند. با در نظر گرفتن جایگشت‌های والد اول و والد دوم، دو جایگشت فرزند به وسیله انتخاب دو نقطه برش در جایگشت‌های والدین و کپی کردن المان‌های بین دو نقطه برش در فرزندان، و پر کردن باقیمانده جایگشت‌های فرزندان با المان‌های استفاده نشده از والد دیگر با شروع از نقطه برش دوم به بعد، ایجاد می‌شوند.

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

Partially Mapped: در این نوع عملگر جابجایی، دو جایگشت فرزند به وسیله انتخاب دو نقطه برش در جایگشت‌های والدین و کپی کردن المان‌های بین دو نقطه برش در فرزندان و پر کردن باقیمانده جایگشت‌های فرزندان با المان‌های متناظر آنها از والدین، ایجاد می‌شوند. توجه کنید که اگر یکی از المانهای خارج از ناحیه جابجایی فرزندی با یکی از المان‌های ناحیه جابجایی خود یکسان باشد آن المان به عنوان حفره در نظر گرفته می‌شود 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 می‌کنیم. سپس از والد اول مقدار ژنی را که در موقعیت یکسانی با ژن پیدا شده قرار دارد را به عنوان مقدار حفره در نظر می‌گیریم.

Cycle: در این عملگر جابجایی، دو جایگشت فرزند بوسیله شکل دادن یک سیکل در بین والدین ایجاد می‌شوند. به عنوان مثال برای ایجاد فرزند اول، با شروع از اولین المان والد ۱، آن را به اولین ژن در والد دوم نگاشت می‌کنیم. 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 می‌یابد. المان‌هایی از سیکل را که در والد ۱ هستند در فرزند اول قرار می‌دهیم. سپس مکان‌های خالی فرزند اول را با المان‌های متناظر در والد ۲ پر می‌کنیم. فرزند دوم نیز به همین ترتیب ایجاد می‌شود.

Exchange: در این عملگر جابجایی، دو کروموزوم والد انتخاب می‌شوند و به صورت تصادفی دو ژن i و j در یکی از دو کروموزوم والد انتخاب می‌شوند. سپس همین دو ژن در کروموزوم دیگر انتخاب می‌شوند. مجموعه ژنهای با شماره‌های بین i و j را مجموعه جابجایی می‌نامیم. سپس ژن‌های هم شماره در دو مجموعه جابجایی با یکدیگر جابجا می‌شوند (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 ازمجموعه جابجایی دوم و ...). با این عمل دو کروموزوم جدید حاصل می‌شوند که اصطلاحاً فرزندان دو آتاماتای والد خوانده می‌شوند.

Smart Exchange: این عملگر جابجایی دقیقا مانند روش Exchange است با این تفاوت که مثلا ژن شماره i از مجموعه جابجایی اول با ژن شماره i از مجموعه جابجایی دوم زمانی جابجا می‌شود که هزینه اجرای پیوند مشخص شده با ژن i در مجموعه جابجایی اول کمتر از مجموعه دوم باشد. در نهایت هر کروموزوم مجموعه ژنهای i تا j خود را در صورتی که هزینه شان بیشتر از ژنهای کروموزوم دیگر باشد از او کپی می‌کند. شبه کد این عملگر به عنوان مثال در شکل۳ نشان داده شده است.

از آنجا که در این الگوریتم از n کروموزوم (آتاماتا) استفاده می‌شود و هر آتاماتا دارای مشخصه‌های اختصاصی مربوط به خود (وضعیت، اقدام و شئ متناظر هر اقدام) می‌باشد، جهت خوانایی بیشتر شبه کد، این مشخصه‌ها را با پیشوند نام آتاماتا و جداساز نقطه نشان می‌دهیم. مثلا برای نشان دادن وضعیت پیوند u از آتاماتای i از نمایش LAi.State(u) استفاده شده است. در الگوریتم مذکور، $cost_i(LA1)$، هزینه (تعداد رجوعات به دیسک) برای پیوند موجود در اقدام i ام آتاماتای یادگیر ۱ می‌باشد. به عنوان مثال دو آتاماتا از جمعیت تشکیل شده به صورت تصادفی انتخاب می‌شوند. سپس با انتخاب تصادفی دو محل a2 وa3 مجموعه جابجایی{a2,a3} حاصل می‌شود. در نهایت مطابق شکل ۵ با جابجایی اقدام‌های متناظر در فاصله جابجایی، دو کروموزوم جدید حاصل می‌شود عملگر جهش[۲۸]: برای انجام دادن این عملگر از روشهای مختلفی که برای کار با جایگشت‌ها مناسب هستند استفاده شده است. روش‌های پیاده‌سازی شده برای عملگر جهش عبارتند از:

28 Mutation

```
Procedure Crossover ( LA1, LA2 )
 Generate two random numbers r1 and r2 between 1 to n
 r1 = Random *n; r2 = Random *n;
 r1 = Min(r1, r2 ), r2 = Max(r1, r2 )
 For i = r1 to r2 do
  If (cost_i( LA1) < cost_i( LA2) ) then
    j = Action of LA2 where
    LA2.Object(LA2.Action( j )) =LA1.Object(LA1.Action( i ));
    Swap(LA2.Object(LA2.Action(i)),LA2.Object(LA2.Action(j)));
  Else
    j = Action of LA1 where
    LA1.Object(LA1.Action( j ))=LA2.Object(LA2.Action( i ));
    Swap(LA1.Object(LA1.Action(i)),LA1.Object(LA1.Action(j)));
  Endif
 End For
End Crossover
```

شکل ۳: شبه کد عملگر جابجایی

Order Based (Swap): ایـن عملگـر، دو اقـدام (ژن) از یـک آتاماتـا (کرومـوزوم) را بـه صـورت تـصادفی انتخـاب نمـوده و جابجا می‌کند.

SubList Based: دو ژن از کرومـــوزوم را بطـــور تـــصادفی انتخاب کرده و ترتیب آنها را معکوس می‌کند.

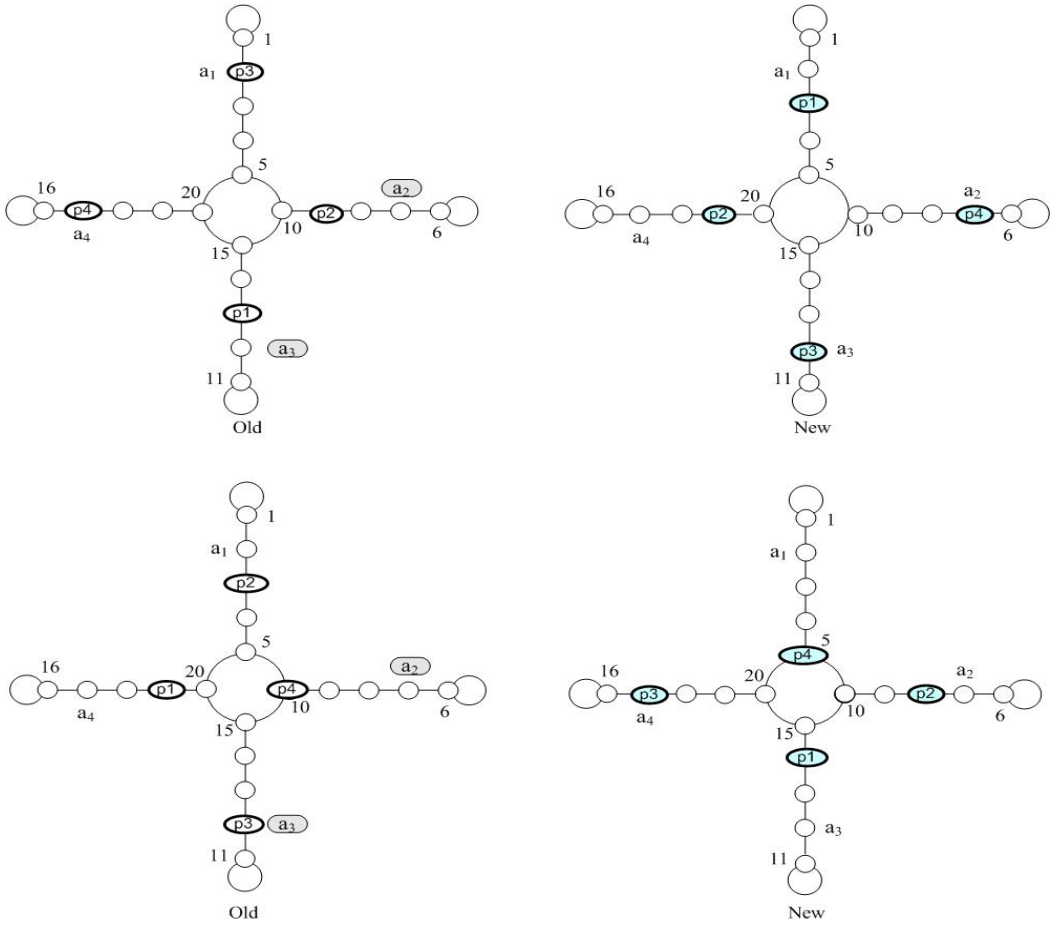
Insertion: ژن یـــا بلـــوکی از ژنهـــا را بطـــور تـــصادفی در کرومـوزوم انتخـاب کـرده و آنـرا در نقطـه تـصادفی دیگـری درج می‌کند.

Scramble: یـک بلـوک از ژنهـا را بطـور تـصادفی انتخـاب کرده و آنها را بصورت تصادفی دوباره مرتب می‌کند.

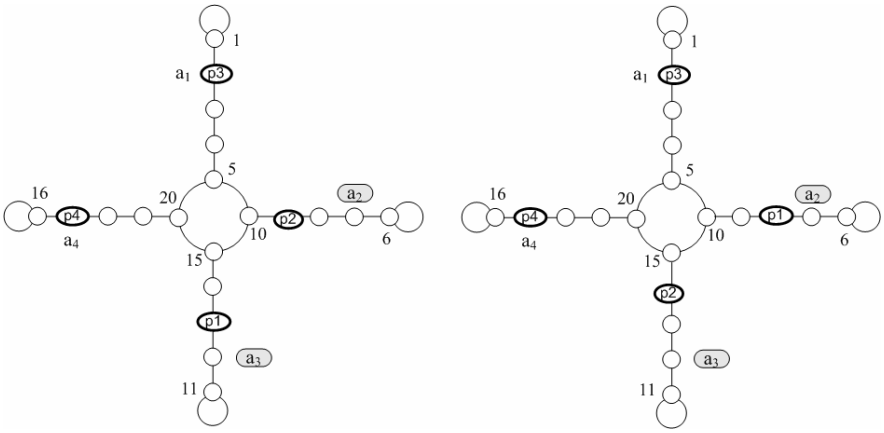
بـه عنـوان نمونـه شـکل ۴ شـبه کـد عملگـر جهـش Swap Mutation و شـــکل ۶ نیـــز مثـــالی از ایـــن عملگـــر را نـــشان می‌دهد.

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

شکل ۴: شبه کد عملگر جهش



شکل۵. نحوه انجام عملگر جابجایی

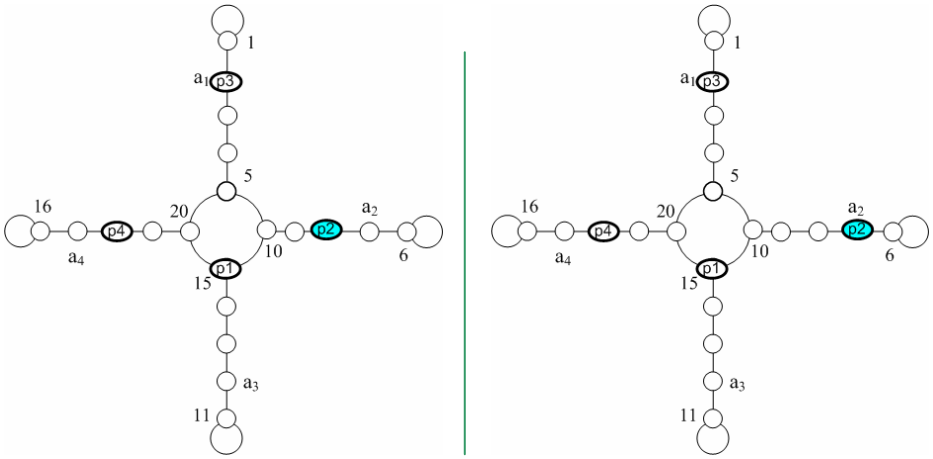


شکل۶: نحوه انجام عملگر جهش

عملگر جریمه و پاداش: در هر یک از کروموزوم‌ها پس از بررسی میزان برازندگی یک ژن که به صورت تصادفی انتخاب شده است، به آن ژن پاداش یا جریمه داده می‌شود. در اثر پاداش دادن یا جریمه کردن یک ژن، عمق ژن تغییر می‌کند. شکل ۷ شبه کد عملگر پاداش را نشان می‌دهد.

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

شکل ۷. شبه کد عملگر پاداش



شکل۸. نحوه پاداش پیوند p2

به عنوان مثال در آتاماتای با اتصال‌های مشابه آتاماتای ستلین اگر پیوند p2 در مجموعه وضعیت‌های {۶،۷،۸،۹،۱۰} قرار داشته باشد و هزینه برای پیوند p2 موجود در اقدام دوم از متوسط هزینه‌های پیوندهای موجود در کروموزوم، کمتر باشد، به این پیوند پاداش داده می‌شود و اهمیت پیوند افزایش می‌یابد و به سمت وضعیت‌های داخلی‌تر این اقدام حرکت می‌کند. اگر پیوند p2 در وضعیت داخلی۶ قرار داشته و پاداش بگیرد، در همان وضعیت باقی می‌ماند. این مورد در شکل ۸ نشان داده شده است. همچنین اگر هزینه پیوندی از متوسط هزینه پیوندهای موجود در کروموزوم، بیشتر باشد، در اینصورت محل قرارگیری این پیوند مناسب نبوده و جریمه می‌شود شبه کد عملگر جریمه در شکل ۹ نشان داده شده است.

نحوه حرکت چنین پیوندی برای دو حالت مختلف در زیر آمده است. الف) پیوند در وضعیتی غیر از وضعیت مرزی قرار داشته باشد: جریمه نمودن این راس سبب کم اهمیت شدن این راس می‌شود. نحوه حرکت چنین راسی در شکل ۱۰ نشان داده شده است.

ب) پیوند در وضعیت مرزی قرار دارد. در این حالت پیوندی از پرس‌وجو را پیدا می‌کنیم بطوریکه اگر در طرح اجرایی مربوطه جای دو راس عوض شوند بیشترین کاهش در مقدار هزینه حاصل گردد در اینصورت اگر پیوند پیدا شده در وضعیت مرزی قرار داشته باشد جای دو پیوند عوض می‌شود و در غیر اینصورت ابتدا پیوند مشخص شده به وضعیت مرزی اقدام خود منتقل و سپس جابجایی صورت می‌پذیرد. نحوه حرکت چنین پیوندی در شکل ۱۱ نشان داده شده است. در شکل ۱۲ نیز شبه‌کد الگوریتم ترکیبی تعیین ترتیب پیوندها در پرس‌وجو نشان داده شده است.

۵.نتایج آزمایش‌ها

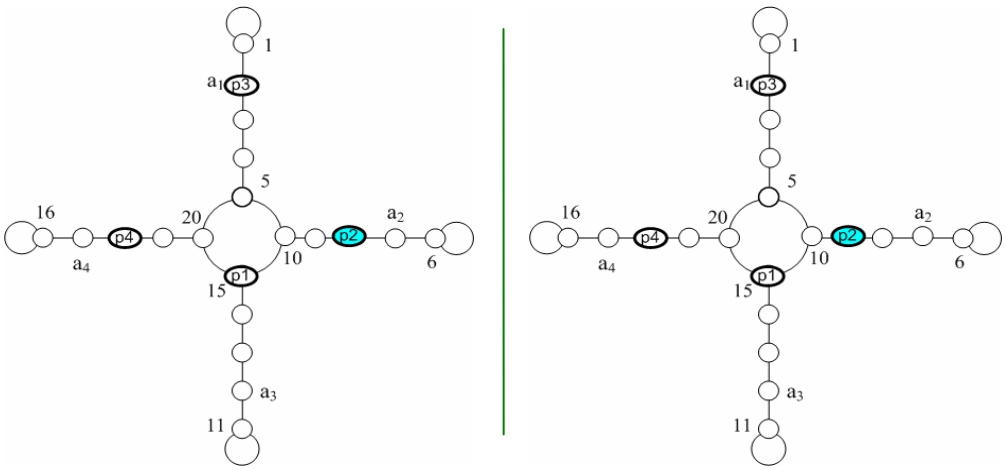
در این بخش نتایج آزمایش‌های انجام شده توسط الگوریتم‌های مبتنی بر آتاماتاهای یادگیر(LA)، الگوریتم ژنتیک (GA) و الگوریتم ترکیبی (GALA) ارایه می‌گردد. قبل از مقایسه نتایج الگوریتم‌ها، به تنظیم پارامترهای الگوریتم ترکیبی خواهیم پرداخت.

این کار در شکلهای ۱۳ تا ۱۷ انجام گرفته است. شکل ۱۳ نتایج حاصل از اجرای الگوریتم ترکیبی با استفاده از عملگرهای جابجایی مختلف با نرخ‌های متفاوت را نشان می‌دهد. شکل ۱۴ نیز نتایج حاصل از اجرای الگوریتم ترکیبی با استفاده از عملگرهای جهش مختلف با نرخ‌های متفاوت را نمایش داده است.

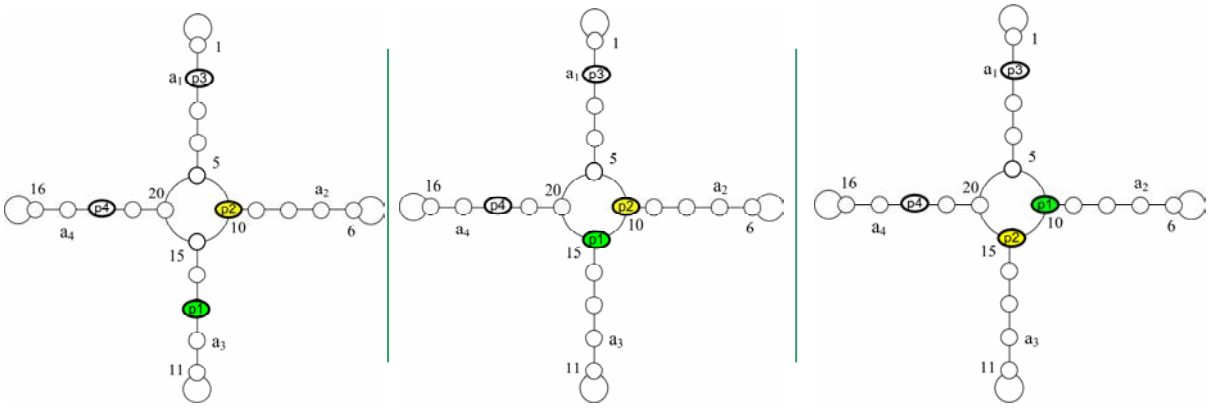
شکل ۱۵ عملگرهای جابجایی و جهش مختلف در تعامل با یکدیگر را در اجرای الگوریتم ترکیبی نشان می‌دهد. شکل ۱۶ حاکی از مقایسه هزینه بدست آمده از الگوریتم ترکیبی برای تعداد نسلهای مختلف می‌باشد. شکل ۱۷ نشان دهنده مقایسه هزینه بدست آمده از الگوریتم ترکیبی برای تعداد کروموزومهای مختلف تشکیل دهنده جمعیت است. بنابراین با توجه به نتایج این نمودارها در انجام آزمایش‌هایمان پارامترهای الگوریتم را بصورت زیر قرار داده‌ایم:

```
Procedure Penalize( LA, u )
  repeat
    For U = 1 to n do
      If (LA.State(U)) mod N <> 0 then
        Inc(LA.State(U));
      End If
    End for
  Until at least one join appears in the boundary state
  bestcost = ∞ ;
  for U = 1 to n do
     Create QEP  LA′ from LA by swapping u and U
     If  cost_i( LA′) < bestError then
         bestcost = cost_i( LA′);
         bestjoin = U;
     End If
  End for
  LA.State(bestjoin) = LA.Action( bestjoin)*N;
  LA.State(u) = LA.Action(u)*N;
  Swap(LA.State(u),LA.State(bestjoin));
End Penalize
```

شکل۹. شبه کد عملگر جریمه



شکل۱۰. نحوه جریمه کردن یک راس که در وضعیتی غیر از وضعیت مرزی قرار داشته باشد.



شکل۱۱. نحوه جریمه کردن یک راس که در وضعیت مرزی قرار داشته باشد

```
Function JoinOrdering(Query)
  //n=Number of Joins
  Create the initial population LA1 … LAn;
  EvalFitness();
  While ( Not (StopCondition)) do
    NewLA1=NewLA2= LA with minimum Value of Cost;
    For i = 2 to n do
      Select LA1 ; Select LA2 ;
      If ( Random > 0.9 ) then
        Crossover ( LA1, LA2 );
      End If
      If (Random > 0.6 ) then
        Mutation ( LA1 ); Mutation ( LA2 );
      End If
      NewLAi+1 = LA1; NewLAi+2 = LA2 ;
      i=i+2;
    End For
    For i = 0 to n do
      LAi = NewLAi;
      u = Random *n;
      If (costu( LAi) <MeanCost ) then Reward(LAi , u );
      Else Penalize(LAi , u );
      End If
    End For
    EvalFitness();
  End While
End JoinOrdering
```

شکل۱۲. شبه کد الگوریتم ترکیبی

اندازه جمعیت:۷۰، تعداد تکرار الگوریتم:۵۰۰، نوع عملگر جابجایی و نرخ آن:Ordered با نرخ ۰,۸ و نوع عملگر جهش و نرخ آن:Sublist Based با نرخ ۰,۷.

پس از تنظیم پارامترها به مقایسه نتایج الگوریتم‌ها می‌پردازیم. شکل ۱۸ نتایج حاصل از اجرای الگوریتم ترکیبی مبتنی بر اتصالات کرایلو[29] را در مقایسه با آتاماتای یادگیر و الگوریتم ژنتیکی نشان می‌دهد. در تمام نمودارها محور عمودی نشان دهنده متوسط هزینه طرح‌های اجرایی بدست آمده توسط هر یک از الگوریتم‌ها می‌باشد همچنین در شکلهای ۱۸ تا ۲۲ محور افقی نشان دهنده تعداد پیوندهای موجود در پرس‌وجوی داده شده است. نتایج آزمایشات نشان می‌دهند که الگوریتم ترکیبی نسبت به الگوریتم ژنتیکی نتایج مناسبتری را می‌دهد و هزینه طرح‌های اجرای بدست آمده با استفاده از الگوریتم ترکیبی برای تعداد پیوندهای مشخص شده کمتر از الگوریتم ژنتیکی می‌باشد. اما نتایج الگوریتم ترکیبی با اتصالات ستلین نسبت به آتاماتای یادگیر بهبود مناسبی نداشته است و در اکثر مواقع نتایج این دو الگوریتم به هم نزدیک بوده و در مواقعی نیز الگوریتم ترکیبی عملکرد بهتری دارد. شکلهای ۱۹ و ۲۰ بترتیب نتایج الگوریتم ترکیبی مبتنی بر اتصالات آتاماتای کرینیسکی[30] و اومن را در مقایسه با الگوریتم ژنتیکی و آتاماتای یادگیر نشان می‌دهد. در شکل ۲۱ نتایج الگوریتم ترکیبی با اتصالات مختلف آتاماتا و با عمقهای متفاوت برای پرس‌و جویی شامل ۸۰ عملگر پیوند مورد مقایسه قرار گرفته است. آزمایشات، نشان دهنده برتری الگوریتم ترکیبی نسبت به دیگر روشها می‌باشد و می‌توان گفت که هزینه طرح‌های اجرای بدست آمده از طریق الگوریتم ترکیبی مبتنی بر آتاماتای کرایلو، کرینیسکی و اومن در همه موارد، کمتر از دیگر الگوریتمها بوده است. مقایسه نتایج الگوریتمهای ترکیبی دارای اتصالات مختلف آتاماتا با یکدیگر نیز حاکی از برتری الگوریتم ترکیبی مبتنی بر اتصالات آتاماتای کرینیسکی نسبت به الگوریتمهای ترکیبی با سایر اتصالات دارد که این مورد در شکل ۲۲ نشان داده شده است. بنابراین می‌توان گفت که از بین الگوریتمهای ارایه شده، الگوریتم ترکیبی مبتنی بر اتصالات کرینیسکی مناسب‌ترین

29 Krylov

30 Krinisky

روش برای حل مسئله ترتیب پیوندها در پرس‌وجوهای پایگاه داده می‌باشد.

۶.نتیجه‌گیری

در این مقاله یک الگوریتم ترکیبی برای حل مساله ترتیب عملگرهای پیوند در پرس‌وجوهای پایگاه داده‌ای پیشنهاد گردید. این الگوریتم از دو روش الگوریتم‌های ژنتیکی و آتاماتاهای یادگیر بطور همزمان برای جستجوی در فضای حالت استفاده می‌نماید. در این مقاله، نشان داده شد که با استفاده همزمان آتاماتای یادگیر و الگوریتم ژنتیک در فرایند جستجو، سرعت رسیدن به جواب به شکل مطلوبی بهبود می‌یابد. نتایج آزمایش‌ها، برتری این روش را نسبت به روش-های مبتنی بر الگوریتمهای ژنتیکی و آتاماتاهای یادگیر نشان می‌دهد.

مراجع

[1] E.F. Codd, A relational model of data for large shared data banks, CACM 13 (6) (1970) 377–387.

[2] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price, Access path selection in a relational database management system, In Proc. Of the ACM SIGMOD Conf. on management of Data, Boston, USA (1979) 23–34.

[3] K. Bennet, M. C. Ferris, Y. E. Ioannidis, A genetic algorithm for database query optimization, In Proc. Of the Fourth Intl. Conf. on Genetic Algorithms, San Diego, USA (1991) 400–407.

[4] T. Ibaraki and T. Kameda, Optimal nesting for computing N-relational joins, ACM Trans. on Database Systems, 9 (3) (1984) 482–502.

[5] A. Swami, A. Gupta, Optimization of large join queries, In Proc. Of the ACM SIGMOD Conf. on Management of Data, Chicago, USA (1988) 8–17.

[6] A.Swami, Optimization of large join queries: Combining heuristics and combinational techniques, In Proc. Of the ACM SIGMOD Conf. on Management of Data, Portland, OR, USA (1989) 367–376.

[7] M. Steinbrunn, G. Moerkotte, A. Kemper, Heuristic and randomized optimization for the join ordering problem, VLDB Journal: Very Large Data Bases 6 (3) (1997) 191–208.

[8] R. Lanzelotte, P. Valduriez, M. Zait, On the effectiveness of optimization search strategies for parallel execution spaces, In Proc. Of the Conf. on Very Large Data Bases (VLDB), Dublin, Ireland (1993) 493–504.

[9] M. M. Astrahan et al, System R: A relational approach to data management., ACM Trans. on Database Systems, 1 (2) (1976) 97–137.

[10] R. Krishnamurthy, H. Boral, C. Zaniolo, Optimization of nonrecursive queries, In Proc. of the Conf. on Very Large Data Bases (VLDB), Kyoto, Japan (1986) 128–137.

[30]

[11] A. Swami, B. Iyer, A polynomial time algorithm for optimizing join queries, In Proc. IEEE Conf. on Data Engineering, Vienna, Austria (1993) 345–354.

[12] Y. E. Ioannidis, Y. C. Kang, Randomized algorithms for optimizing large join queries, In Proc. Of the ACM SIGMOD Conf. on Management of Data, Atlantic City, USA (1990) 312–321.

[13] Y. Ioannidis, E. Wong, Query optimization by simulated annealing, In Proc. Of ACM SIGMOD Conf. on the Management of Data, San Francisco, CA (1987) 9–22.

[14] S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi, Optimization by simulated annealing, Science, 220 (4598) (1983) 671–680.

[15] C. Galindo-Legaria, A. Pellenkoft, M. Kersten, Fast, randomized join-order selection why use transformations, In Proc. Of the 20th Int. Conf. on Very Large Data Bases (VLDB), Santiago, Chile (1994) 85–95.

[16] M. Stillger, M. Spiliopoulou, Genetic programming in database query optimization, In Proc. Of the First Annual Conf. on Genetic Programming, USA (1996) 388–393.

[17] V. Muntes-Mulero, J. Aguilar-Saborit, C. Zuzarte, J.-L. Larriba-Pey, Cgo: a sound genetic optimizer for cyclic query graphs, In Proc. Of ICCS 2006,UK,2006, pp.156-163.

[18] H. Beigy, M. R. Meybodi, Randomized Las Vegas Algorithm for Graph Isomorphism, Proceedings of Third International Conference on Intelligent Data Engineering and Automated Learning,Manchester,UK (2004) 12–14.

[19] Y. Wang, K. Fan Genetic-Basic Search for Error-Correcting Graph Isomorphism, IEEE Transaction on Systems, Man. And Cybernetics-Par`t B: Cybernetics 27 (4) (1997).

[20] P. Mars, K. S. Narendra, M. Chrystall, Learning Automata Control of Computer Communication Networks, Proc. Of Third Yale workshop on Application of Adaptive Systems Theory. Yale University, 1983.

[21] A. Hashim, S. ,Amir, P. Mars, Application of Learning Automata to Data Compression, In Adaptive and Learning Systems, K. S. Narendra (Ed), New York: Plenum Press,1986, pp. 229–234.

[22] M. A. L. Thathachar, P. S. Sastry, Learning Optimal Discriminant Functions Through a Cooperative Game of Automata, IEEE Trans, Syst., Man and Cybern 27 (4) (1997) 588-597.

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

[24] M. R. Meybodi, S. Lakshmivarhan, A Learning Approach to Priority Assignment in a Two Class M/M/1 Queuing System with Unknown Parameters, Pproc. Of Third Yale Workshop on Applications of Adaptive System Theory, Yale University,1983, pp. 106–109.

[25] M. R. Meybodi, H. Beigy, New Class of Learning Automata Based Scheme for Adaptation of Backpropagation Algorithm Parameters, Proc. Of EUFIT-98, Achen, Germany (1998) 339–344.

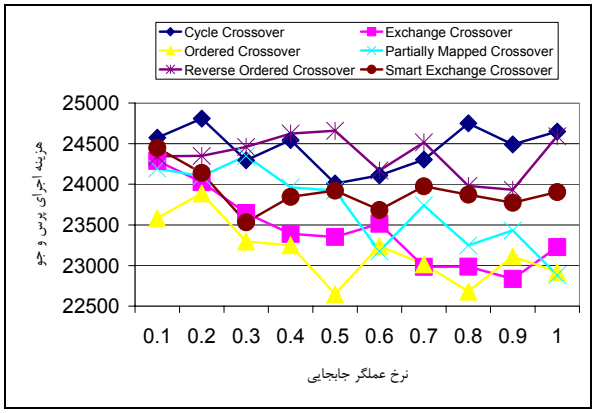
[26] B. J. Oommen, D. C. Y. Ma, Deterministic Learning Automata Solution to the Keyboard Optimization Problem, IEEE Trans. On Computers 37 (1) (1988) 2–3.

[27] H. Beigy, M. R. Meybodi, Optimization of Topology of neural Networks Using Learning Automata, Proc. Of 3th Annual Int. Computer Society of Iran Computer Conf. CSICC-98, Tehran, Iran (1999) 417–428.

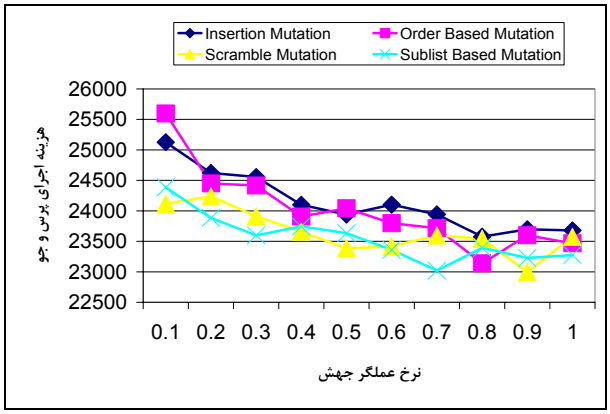
[28] B. Falkenhainer, K.D. Forbus, D. Gentner, The Structure-mapping Engine: Algorithms and Examples, Artificial Intelligence (41) (1989) 1–63.

[29] E. Cantu-Paz, A Survey of Parallel Gentic Algorithms, IlliGAL Report (97003) (1997).

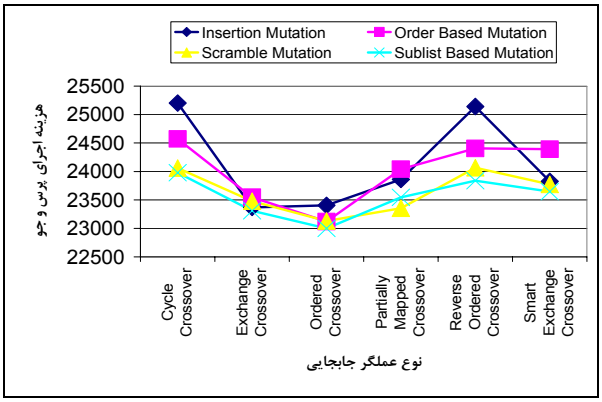
ضمائم



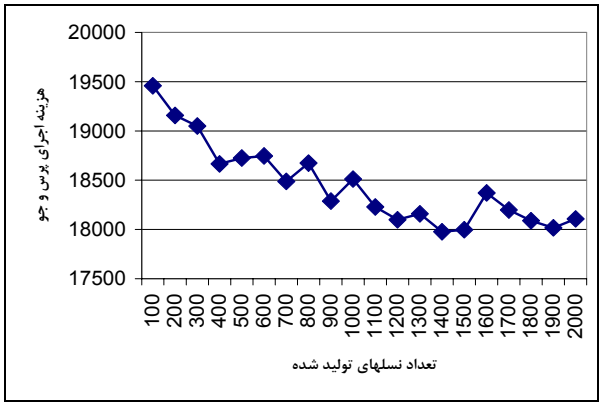
شکل ۱۳. مقایسه هزینه بدست آمده از الگوریتم ترکیبی برای عملگرهای جابجایی مختلف با نرخهای متفاوت.



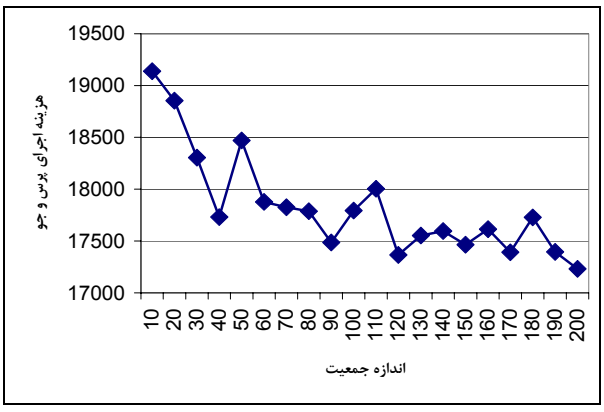
شکل ۱۴. مقایسه هزینه بدست آمده از الگوریتم ترکیبی برای عملگرهای جهش مختلف با نرخهای متفاوت.



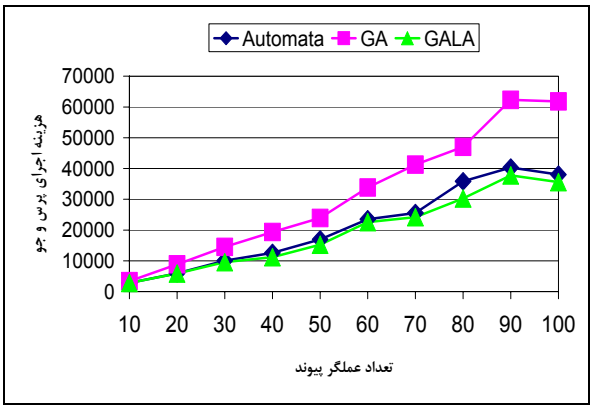
شکل ۱۵. مقایسه هزینه بدست آمده از الگوریتم ترکیبی برای عملگرهای جهش و جابجایی مختلف در تعامل با یکدیگر.



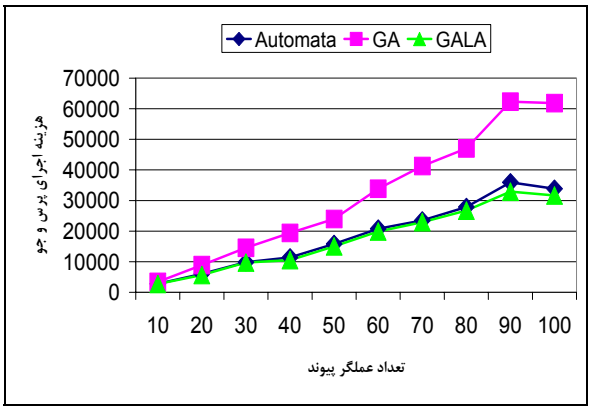
شکل۱۶ . مقایسه هزینه بدست آمده از الگوریتم ترکیبی برای تعداد نسلهای مختلف.



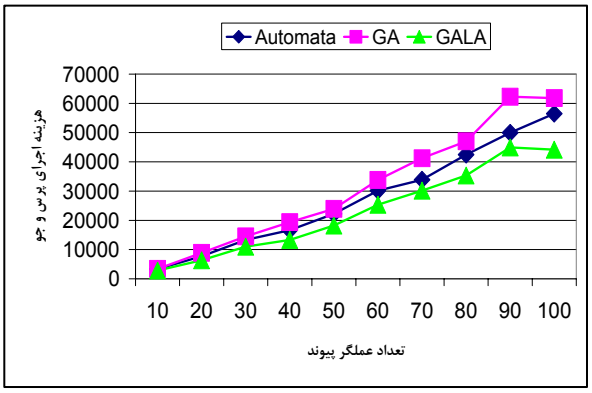
شکل ۱۷. مقایسه هزینه بدست آمده از الگوریتم ترکیبی برای تعداد جمعیت مختلف.



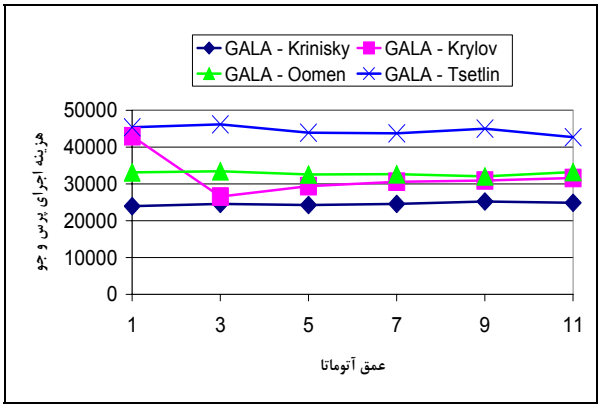
شکل ۱۸. مقایسه هزینه بدست آمده از الگوریتمهای ترکیبی و آتاماتای یادگیر مبتنی بر اتصالات کرایلو و الگوریتم ژنتیکی.



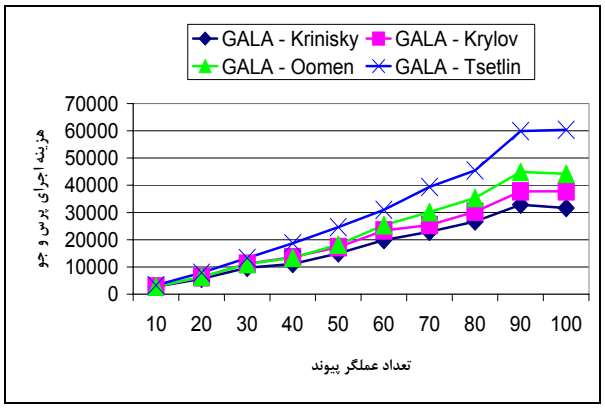
شکل ۱۹. مقایسه هزینه بدست آمده از الگوریتمهای ترکیبی و آتاماتای یادگیر مبتنی بر اتصالات کرینیسکی و الگوریتم ژنتیکی.



شکل ۲۰. مقایسه هزینه بدست آمده از الگوریتمهای ترکیبی و آتاماتای یادگیر مبتنی بر اتصالات اومن و الگوریتم ژنتیکی.



شکل ۲۱. مقایسه هزینه بدست آمده از الگوریتمهای ترکیبی برای آتاماتاهای یادگیر مبتنی بر اتصالات مختلف و عمقهای متفاوت.



شکل ۲۲. مقایسه هزینه بدست آمده از الگوریتمهای ترکیبی برای آتاماتاهای یادگیر مبتنی بر اتصالات مختلف.