سیزدهمین کنفرانس ملی انجمن کامپیوتر ایران
جزیره کیش، خلیج فارس، ایران ۱۹ الی ۲۱ اسفند ۱۳۸۶
انجمن کامپیوتر ایران
Computer Society of Iran
دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر
پوستر

رویکرد نوین مبتنی بر الگوریتم ژنتیک در کمینه کردن هزینه اجرای عملگرهای پیوند در پایگاه داده

علی صفری ممقانی
گروه مهندسی کامپیوتر،
دانشگاه آزاد اسلامی،
بناب، ایران
safari_m_61@yahoo.com

کیوان اصغری
گروه مهندسی کامپیوتر،
دانشگاه آزاد اسلامی،
خامنه، ایران
kayvan.asghari@yahoo.com

محمد رضا میبدی
دانشکده مهندسی کامپیوتر،
دانشگاه صنعتی امیرکبیر،
تهران، ایران
mmeybodi@aut.ac.ir

فریبرز محمودی
دانشکده مهندسی کامپیوتر،
دانشگاه آزاد اسلامی،
قزوین، ایران
mahmoudi@itrc.ac.ir

چکیده: انتخاب یک ترتیب مناسب برای عملگر پیوند در پرس و جوهای پایگاه داده ای یک مسئله NP-Hard است. استفاده از تکنیکهای جستجوی جامع برای این مسئله مناسب نیست. در این مقاله یک الگوریتم ژنتیکی طراحی گردیده است که کروموزومهای بکار رفته در آن بصورت آتاماتای یادگیر مهاجرت اشیاء می باشند. نشان داده شده است که استفاده از این الگوریتم ژنتیکی در بهبود جواب ها بسیار مؤثر می باشد و علاوه بر افزایش سرعت الگوریتم در رسیدن به جوابهای مناسب، از بدام افتادن آن در بهینه های محلی جلوگیری می‌کند.

واژه های کلیدی: پرس‌وجو، عملگر پیوند، آتاماتای یادگیر مهاجرت اشیاء، الگوریتم‌های ژنتیک

۱- مقدمه

با توجه به هزینه‌های ارزیابی بالای عملگر پیوند، کار بر روی این عملگر هدف اولیه بهینه‌سازهای پرس‌وجوی های رابطه‌ای است. اگر پرس‌جوها در یک حالت محاوره‌ای باشند، شامل تعدادکمی رابطه خواهد بود که بهینه‌سازی این عبارات، می‌تواند بوسیله یک جستجوی جامع انجام گیرد. در صورتی که تعداد روابط بیش از ۵ یا ۶ رابطه باشد، تکنیک‌های جستجوی جامع پرهزینه خواهد بود. پرس‌وجوهای با تعداد پیوند زیاد در سیستم‌های جدید مانند سیستم‌های مدیریت پایگاه داده های استنتاجی و سیستم‌های خبره، سیستم‌های مدیریت پایگاه دادهای مهندسی و سیستم‌های پشتیبان تصمیم، داده‌کاوی و سیستم‌های مدیریت پایگاه‌داده‌های علمی و ... دیده می‌شوند.

گروهی 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 این فضا را کاهش می‌دهند[۴]. یکی از این الگوریتم‌ها، روش برنامه-نویسی پویاست که اولین بار برای بهینه‌سازی ترتیب پیوند در System-R توسط سلینگر و همکارانش مطرح گردید[۱]. مهمترین ایراد الگوریتم این بود که با افزایش تعداد روابط موجود در پرس‌وجو، نیازمند مصرف زیاد حافظه و پردازنده می‌باشد. از الگوریتم‌های قطعی دیگر می‌توان به الگوریتم حداقل قابلیت انتخاب [۴] ، الگوریتم KBZ[۵] و الگوریتم AB [۶] اشاره کرد. جهت نشان دادن ناتوانی و ضعف الگوریتم‌های قطعی در مقابله با پرس‌وجوهای بزرگ، الگوریتم‌های تصادفی معرفی شده‌اند. الگوریتم‌های مطرح شده در این زمینه، الگوریتم بهبود مکرر، الگوریتم نرم کردن شبیه‌سازی شده ، بهینه‌سازی دو مرحله‌ای نمونه‌گیری تصادفی می باشد[۸، ۷].

اولین کار انجام شده بر روی مسئله بهینه‌سازی ترتیب پیوندها با استفاده از الگوریتم ژنتیکی، توسط بنت و همکارانش ارایه گردید[۲]. 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. یکی از دیگر کارهای صورت گرفته مبتنی بر الگوریتم ژنتیکی توسط استین بران و همکارانش بوده است[۴] که روشهای کدگذاری و تکنیک‌های متفاوتی را در این زمینه بکار گرفته اند. نمونه دیگر از الگوریتم‌های تکاملی بکار گرفته شده برای مسئله بهینه‌سازی پیوندها، روش برنامه‌نویسی ژنتیکی می‌باشد که توسط استیلگر و اسپیلیوپولو[۹] مطرح شده است و بهینه‌ساز CGO، که توسط مولرو همکارانش در[۱۰] ارایه شده است.

در این مقاله یک الگوریتم ترکیبی برای حل مساله بهینه‌سازی ترتیب عملگرهای پیوند در پرس و جوهای پایگاه داده‌ای پیشنهاد شده است. این الگوریتم از دو روش الگوریتم‌های ژنتیکی و آتاماتاهای یادگیر بطور همزمان برای جستجوی در فضای حالات استفاده می‌نماید. نشان داده شده است که با استفاده همزمان از آتاماتاهای یادگیر و الگوریتم‌های ژنتیکی در فرایند جستجو، سرعت رسیدن به جواب افزایش چشمگیری پیدا می نماید و همچنین از بدام افتادن الگوریتم در مینیمم های محلی جلوگیری می شود. نتایج آزمایش‌ها، برتری الگوریتم ترکیبی را نسبت به الگوریتم‌های مبتنی بر الگوریتم‌های ژنتیکی و یا آتاماتاهای یادگیر نشان می‌دهد. ادامه مقاله بدین صورت سازماندهی شده است: بخش دوم مساله ترتیب عملگرهای پیوند در پرس و جوهای پیوندی را تعریف می‌کند. توضیح مختصری درباره آتاماتاهای یادگیر و الگوریتم

های ژنتیک در بخش ۳ آمده است. در بخش ۴، الگوریتم ترکیبی پیشنهادی در این مقاله شرح داده می‌شود و در بخش ۵ نتایج آزمایش‌ها ارایه می‌گردد. بخش۶ نیز شامل نتیجه‌گیری می‌باشد.

۲- تعریف مسئله

بهینه‌سازی پرس‌وجو، فعالیتی است که طی آن، یک طرح کارا برای اجرای پرس‌وجو(qep) ، تولید می‌گردد و یکی از مراحل اساسی در پردازش پرس‌وجو می‌باشد که سیستم مدیریت پایگاه داده در این مرحله از بین تعدادی طرح اجرا، بهترین طرح را بر می‌گزینند، بگونه‌ای که اجرای پرس‌وجو داده شده توسط کاربر با استفاده از این طرح دارای کمترین هزینه، بویژه هزینه عملیات ورودی/خروجی می‌باشد. 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 [۲].

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

طرح اجرایی برای پاسخ به یک پرس‌وجو، دنباله‌ای از عملگرهای جبری رابطه‌ای می‌باشد که بر روی روابط پایگاه‌داده اعمال می‌شوند و جواب لازم برای آن پرس‌وجو را تولید می‌کنند. از بین عملگرهای رابطه‌ای موجود، پردازش و بهینه سازی عملگر پیوند 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 و جابجایی می‌باشد، تعداد طرح های اجرایی موجود برای پاسخ‌دهی به یک پرس‌وجو به صورت نمایی با تعداد پیوندهای بین آنها رشد می‌کند. علاوه بر این موارد، یک سیستم مدیریت پایگاه داده معمولا انواع روش‌های پیوند برای پردازش پیوندها و انواع اندیس‌ها را برای دسترسی به روابط پشتیبانی می‌کند، بطوریکه این موارد، گزینه‌های لازم برای جواب دهی به یک پرس‌وجو را هر چه بیشتر می‌کند. گرچه تمامی طرح‌های اجرایی موجود برای پاسخ به یک پرس‌وجوی مشخص، دارای خروجی یکسان می‌باشند ولی 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، می‌تواند در هزینه کلی، تاثیرگذار باشد. مسئله بهینه سازی پرس‌وجو که اغلب از آن بعنوان مسئله انتخاب ترتیب مناسب برای اجرای عملگرهای پیوند یاد می‌شود، یک مسئله NP-hard می‌باشد[۳].

۳- آتاماتاهای یادگیر و الگوریتمهای ژنتیکی

یادگیری در آتاماتاهای یادگیر، انتخاب یک اقدام بهینه از میان یک مجموعه از اقدام های مجاز آتاماتا می‌باشد. این اقدام روی یک محیط تصادفی اعمال می‌شود و محیط به این اقدام آتاماتا بوسیله یک پاسخ تصادفی از مجموعه پاسخ های مجاز جواب می دهد. 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[۱۱] مراجعه نمود. برای یک پرس و جو با اندازهn ، 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 بدلیل تعداد زیاد اقدام ها، سرعت همگرایی کاهش می یابد و به همین جهت، آتاماتای مهاجرت اشیاء توسط اومن و همکارانش پیشنهاد شده است[۱۱].

الگوریتم های ژنتیکی که برمبنای ایده تکامل در طبیعت عمل می نمایند، بر روی جمعیتی از راه حلهای بالقوه به جستجوی راه حل نهایی می پردازند. در هر نسل، بهترینهای آن نسل انتخاب می شوند، و پس از زاد و ولد، مجموعهٔ جدیدی از فرزندان را تولید می کنند.

۴- حل مسئله بهینه سازی پرس وجو با الگوریتم ترکیبی

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 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 آتاماتای یادگیر در الگوریتم ترکیبی وجود دارد. خودترمیمی، تولید مثل، جریمه و پاداش از ویژگی‌های الگوریتم ترکیبی است.درادامه پارامترهای اصلی این الگوریتم توضیح داده شده است.

ژن و کروموزوم: در الگوریتم پیشنهادی برخلاف الگوریتم های ژنتیکی کلاسیک، از کدگذاری دودویی برای کروموزوم ها استفاده نمی شود. 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 اقدام های مجاز برای آتاماتای یادگیر است. این آتاماتا k اقدام دارد (تعداد اقدام های این آتاماتا با تعداد اعمال پیوند برابر است). اگر پیوند شماره u از پرس و جوی داده شده در اقدام m قرار گرفته باشد، در اینصورت پیوند u ، m امین پیوندی از پرس و جو خواهد بود که اجرا می شود. $f = \{f_1, f_2, \dots, f_{KN}\}$ مجموعه وضعیت ها و N عمق

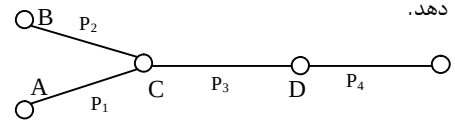
حافظه برای آتاماتا می باشد. مجموعه وضعیت‌های این آتاماتا به k زیر مجموعه $\{f_1, f_2, \dots, f_N\}$ و $\{f_{N+1}, f_{N+2}, \dots, f_{2N}\}$ و ... و $\{f_{(K-1)N+1}, f_{(K-1)N+2}, \dots, f_{KN}\}$ افراز می‌شود و عملگرهای پیوند بر اساس این که در کدام وضعیت قرار داشته باشند دسته بندی می گردند. اگر پیوند شماره u از پرس و جو در مجموعه وضعیت‌های $\{f_{(j-1)N+1}, f_{(j-1)N+2}, \dots, f_{jN}\}$ قرار داشته باشد در اینصورت پیوند u، j امین پیوندی خواهد بود که اجرا می‌شود. در مجموعه وضعیت‌های اقدام j ، به وضعیت $f_{(j-1)N+1}$ وضعیت داخلی و به وضعیت f_{jN}، وضعیت مرزی گفته می‌شود. گره ای که در وضعیت $f_{(j-1)N+1}$ قرار دارد گره با اهمیت بیشتر و گره ای در وضعیت f_{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 می توان 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 پرس و جوی زیر را در نظر بگیرید:

$$(A\infty C) \text{ AND } (B\infty C) \text{ AND } (C\infty D) \text{ AND } (D\infty E)$$

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 P_i را برای نمایش عملگرهای پیوند بکار می‌بریم.

مجموعه عملگرهای پیوند P_1,P_2,P_3,P_4 را در نظر می گیریم تا جایگشتی از ترتیب اجرای عملیات پیوند را با آتاماتای مهاجرت اشیاء مبتنی بر آتاماتای ستلین نشان دهیم. این آتاماتای یادگیر دارای ۴ اقدام $\{a_1,a_2,a_3,a_4\}$ (به تعداد پیوندهای پرس و جو) و عمق می باشد. مجموعه وضعیت های {۲۶، ۲۱، ۱۶، ۱۱، ۶، ۱}وضعیتهای داخلی و مجموعه وضعیتهای {۳۰، ۲۵، ۲۰، ۱۵، ۱۰، ۵}وضعیتهای مرزی آتاماتای یادگیر هستند.درابتدا هریک از پیوندهای پرس وجو در وضعیت مرزی اقدام مربوطه قرار دارند. این آتاماتای یادگیر (کروموزوم)، دارای ۴ اقدام (ژن) می باشد و هر اقدام دارای ۵ وضعیت داخلی می باشد. شکل 2 نحوه نمایش اجرای پیوندهای با ترتیب 3 ، 2 ، 1 و 4 را با آتوماتای مهاجرت اشیای مبتنی بر اتصالات آتوماتای ستلین نشان می دهد.



شکل۱: گراف پرس‌وجو

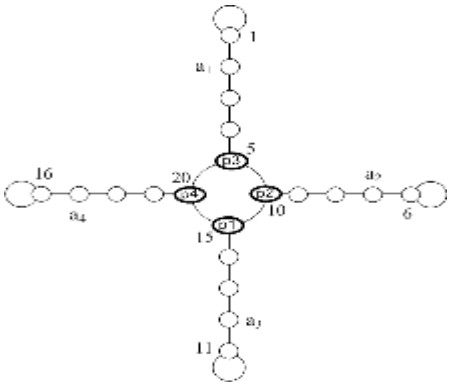
تابع برازندگی: در الگوریتم های ژنتیک تابع برازندگی، شاخص زنده ماندن کروموزوم ها است. هدف جستجوی ترتیب بهینه پیوندهای پرس و جو، یافتن جایگشتی از عملگرهای پیوند می‌باشد که کل هزینه اجرای پرس و جو در این جایگشت، کمینه باشد. نکته‌ای که در محاسبه تابع برازندگی مهم است، تعداد رجوعات به دیسک می‌باشد، بنابراین می‌توانیم تابع برازندگی F را برای یک طرح اجرایی qep بصورت مقابل تعریف نماییم: **تعداد رجوعات به دیسک**/1 = F(qep)

برای محاسبه تعداد رجوعات به دیسک (هزینه) یک طرح اجرایی، هر طرح اجرایی را متناسب با یک درخت پردازشی در نظر می گیریم. هزینه هر گره بصورت بازگشتی (پایین به بالا و از راست به چپ) بوسیله 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 R1 ∞ R2 را در نظرمی‌گیریم:

دراینصورت 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(RK) هزینه بدست آوردن گره فرزند می‌باشدو ازرابطه زیر قابل محاسبه است.

$$C(R_k) := \begin{cases} 0 & \text{اگر } R_k \text{ یک رابطه پایه ای در پرس و جو باشد} \\ C(R_i \infty R_j) & \text{اگر } R_k = R_i \infty R_j \text{ یک نتیجه میانی باشد} \end{cases}$$

بعنوان یک حالت خاص اگر R1 و R2، هر دو رابطه پایه‌ای باشند، هزینه C_{total} برابر با $C(R_1 \infty R_2)$ خواهد بود که با توجه به اینکه نوع پیاده سازی بکار رفته برای عمل پیوند در پرس و جوها از نوع حلقه های تو در تو می باشد، هزینه محاسبه عمل پیوند در این نوع پیاده سازی برابر است با $C(R_1 \infty R_2) = b_{R_1} + b_{R_2}$ که b_{R_1} تعداد بلاکهای رابطه R1 می باشد.



شکل۲: نمایش جایگشتی از پیوندهای (p_3,p_2,p_1,p_4)

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

عملگر انتخاب: عملگر انتخاب بکار رفته برای این الگوریتم، از نوع چرخ رولت می‌باشد.

عملگر ترکیب: شبه کد این عملگر در شکل ۳ 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 شیئ 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 پیوند u از آتاماتای i از نمایش $LA_i.State(u)$ استفاده شده است.

```
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

عملگر جهش: شبه کد عملگر جهش بصورت شکل ۴ است.

```
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

عملگر جریمه و پاداش: در هر یک از کروموزوم‌ها پس از بررسی میزان برازندگی یک ژن که به صورت تصادفی انتخاب می شود، آن ژن پاداش یا جریمه داده می شود. در اثر پاداش یا جریمه کردن یک ژن، عمق ژن تغییر می کند.شکل۵شبه کد عملگر پاداش را نشان می دهد.

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

شکل ۵ : شبه کد عملگر پاداش

در الگوریتم های بالا، $cost_i(LA1)$ ، هزینه (تعداد رجوعات به دیسک) برای پیوند موجود در اقدام i ام آتاماتای یادگیر ۱ می‌باشد.

به عنوان مثال در آتاماتای با اتصال های 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 می شود و اهمیت پیوند افزایش می یابد و به سمت وضعیت های داخلی‌تر این اقدام حرکت می‌کند. اگر پیوند p2 در وضعیت داخلی۶ قرار داشته باشد و پاداش بگیرد در همان وضعیت باقی می‌ماند.

اگر هزینه پیوندی از متوسط هزینه پیوندهای موجود در کروموزوم، بیشتر باشد در اینصورت محل قرارگیری این پیوند مناسب نبوده و جریمه می شود شبه کد عملگر جریمه در شکل۶ نشان داده شده است. نحوه حرکت چنین پیوندی برای دو حالت مختلف در ادامه آمده 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 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
```

شکل۶: شبه کد عملگر جریمه

الف) پیوند در وضعیتی غیر از وضعیت مرزی قرار داشته باشد: جریمه نمودن این راس سبب کم اهمیت شدن این راس می شود.

ب) پیوند در وضعیت مرزی قرار دارد. در این حالت پیوندی از پرس و جو را پیدا می کنیم بطوریکه اگر در طرح اجرایی مربوطه جای دو راس عوض شوند بیشترین کاهش در مقدار خطا حاصل گردد. در اینصورت اگر پیوند پیدا شده در وضعیت مرزی قرار داشته باشد، جای دو پیوند عوض می شود و در غیر اینصورت ابتدا پیوند مشخص شده به وضعیت مرزی اقدام خود منتقل و سپس جابجایی صورت می پذیرد.

در شکل ۷، شبه کد الگوریتم ترکیبی نشان داده شده است. در این شکلMeanCost، متوسط هزینه پیوندهای موجود در کروموزوم است.

```
Function JoinOrdering(Query)
 Create the initial population LA1 … LAn;  //n=Number of Joins
  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
```

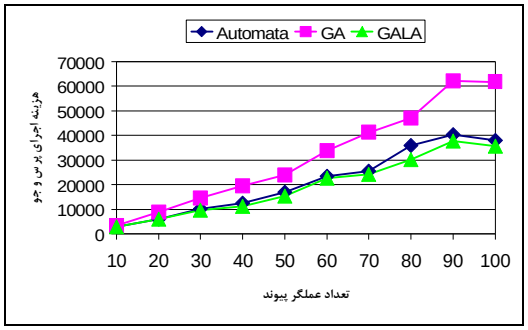
شکل ۷: شبه کد الگوریتم ترکیبی ترتیب پیوندها در پرس و جو

۵- نتایج آزمایش‌ها

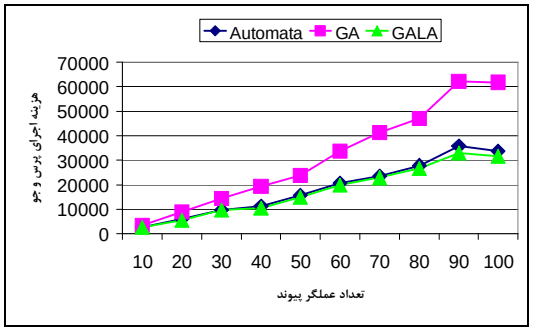
در این بخش، نتایج آزمایش‌های انجام شده توسط الگوریتم‌های مبتنی بر آتاماتاهای یادگیر، الگوریتم ژنتیک و الگوریتم پیشنهادی ارایه می گردد. شکل‌های ۸ و ۹ بترتیب نتایج الگوریتم پیشنهادی مبتنی بر اتصالات آتاماتای کرایلو، کرینیسکی را در مقایسه با نتایج الگوریتم ژنتیکی و آتاماتای یادگیر نشان می دهد. در این نمودارها محور عمودی نشان دهنده متوسط هزینه طرح های اجرایی بدست آمده توسط هر یک از الگوریتم‌ها و محور افقی نشان دهنده تعداد پیوندهای موجود در پرس و جوی داده شده می باشد. آزمایشات، حاکی از برتری الگوریتم پیشنهادی نسبت به دیگر الگوریتمها می باشد و می توان گفت که هزینه طرح های اجرای بدست آمده توسط الگوریتم پیشنهادی مبتنی بر آتاماتای کرینیسکی، کرایلو و حتی اومن در همه موارد، کمتر از دیگر الگوریتمها میباشد. مقایسه نتایج الگوریتمهای ترکیبی پیشنهادی حاکی از برتری الگوریتم ترکیبی پیشنهادی مبتنی بر اتصالات آتاماتای کرینیسکی در مقایسه با الگوریتمهای ترکیبی با سایر اتصالات دارد که این مورد در شکل ۱۰ نشان داده شده است.

۶- نتیجه گیری

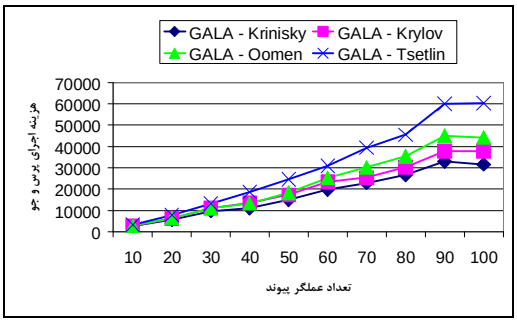
در این مقاله، یک الگوریتم ژنتیکی با ساختار جدید کروموزوم‌ها به صورت آتاماتای مهاجرت اشیا ارائه شده است. ساختار آتاماتای مهاجرت اشیا باعث بهبود جوابهای ایجاد شده می‌شود. نتایج آزمایشها برتری الگوریتم ترکیبی را نسبت به الگوریتم ژنتیکی و آتوماتای یادگیر نشان می دهد. با توجه به اینکه، یکی از مناسب‌ترین الگوریتم ها برای مسئله بهینه سازی پرس و جوهای پیوندی، الگوریتم ژنتیکی می باشد و به دلیل قابلیت غلبه الگوریتم ما به الگوریتم ژنتیکی، الگوریتم ارائه شده توسط ما یک الگوریتم مناسب برای این مسئله می باشد.



شکل۸: هزینه الگوریتم پیشنهادی مبتنی بر اتصالات کرایلو در مقایسه با الگوریتم آتاماتای یادگیر و الگوریتم ژنتیکی



شکل۹: هزینه الگوریتم پیشنهادی مبتنی بر اتصالات کرینسکی در مقایسه با آتاماتای یادگیر و الگوریتم ژنتیکی



شکل۱۰: مقایسه هزینه الگوریتم پیشنهادی مبتنی بر اتصالات ستلین و کرایلو و کرینسکی و اومن

۷- مراجع

[1] 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", Proc. Of the ACM SIGMOD Conf. on management of Data, pages 23-34, Boston, USA, 1979.

[2] K. Bennet, M. C. Ferris and Y. E. Ioannidis, "A Genetic Algorithm for Database Query Optimization", Proc. Of the Fourth Intl. Conf. on Genetic Algorithms, pp. 400-407, San Diego, USA, 1991.

[3] T. Ibaraki and T. Kameda, "Optimal Nesting for Computing N-Relational Joins", ACM Trans. on Database Systems, Vol. 9, No. 3, pp. 482-502, 1984.

[4] M. Steinbrunn, G. Moerkotte, and A. Kemper, "Heuristic and Randomized Optimization for the Join Ordering Problem", VLDB Journal: Very Large Data Bases, Vol. 6, No. 3, pp. 191-208, 1997.

[5] R. Krishnamurthy, H. Boral, and C. Zaniolo, "Optimization of Nonrecursive Queries", Proc. of the Conf. on Very Large Data Bases (VLDB), pp. 128-137, Kyoto, Japan, 1986.

[6] A. Swami and B. Iyer, "A Polynomial Time Algorithm for Optimizing Join Queries", Proc. IEEE Conf. on Data Engineering, pp. 345-354, Vienna, Austria, 1993.

[7] Y. E. Ioannidis and Y. C. Kang, "Randomized algorithms for optimizing large join queries", In Proc. Of the ACM SIGMOD Conf. on Management of Data, pp. 312-321, Atlantic City, USA, 1990.

[8] Y. Ioannidis and E. Wong, "Query Optimization by Simulated Annealing", Proc. Of ACM SIGMOD Conf. on

the Management of Data, pp.9-22, San Francisco, CA, 1987.

[9] M. Stillger and M. Spiliopoulou, "Genetic Programming in Database Query Optimization", Proc. Of the First Annual Conf. on Genetic Programming, pp. 388-393, Stanford University, CA, USA, 1996.

[10] V. Muntes-Mulero, J. Aguilar-Saborit, C. Zuzarte, and J.-L. Larriba-Pey, "CGO: a sound genetic optimizer for cyclic query graphs", In Proc. Of ICCS 2006, pp. 156-163, Reading, Springer-Verlag, UK, 2006.

[11] H. Beigy and M. R. Meybodi, "Randomized Las Vegas Algorithm for Graph Isomorphism", Proceedings of Third International Conference on Intelligent Data Engineering and Automated Learning, Manchester, UK, 2002.