Journal of Soft Computing and Information Technology (JSCIT)

Babol Noshirvani University of Technology, Babol, Iran

Journal Homepage: jscit.nit.ac.ir

Volume 9, Number 3, Fall 2020, pp. 163-179

Received: 10/23/2017, Revised: 04/05/2020, Accepted: 05/14/2020

An Improved View Selection Algorithm in Data Warehouses by Shuffled Frog Leaping Algorithm in 0/1 Knapsack Problem

Reyhaneh Sabbagh Gol[1] and Negin Daneshpour[2*]

1- Faculty of Computer Engineering, Shahid Rajaee Teacher Training University, Tehran, Iran.

2*- Faculty of Computer Engineering, Shahid Rajaee Teacher Training University, Tehran, Iran.

[1]sabbagh.rsg@gmail.com and [2*]ndaneshpour@srttu.edu

Corresponding author's address: Negin Daneshpour, Faculty of Computer Engineering, Shahid Rajaee Teacher Training University, Tehran, Iran.

Abstract- A data warehouse is designed for responding analytical queries. The data in data warehouse are historical. The response time in data warehouse is long. So the response time problem should be solved. Using views is a solution for the problem. But it is impossible to materialize all views. On the other hand, materializing optimal views is a NP-Complete problem. Therefore view selection algorithms were introduced. Some of these algorithms materialize frequent queries. Previously queries have important queries and will be used in the future probably. This paper, proposes an algorithm for materializing proper views. The algorithm finds proper views by using previous queries and materializes them. The views are able to respond many future queries. This paper uses shuffled frog leaping algorithm to find proper views in 0/1 knapsack problem. So the proposed algorithm improves the response time of the previous algorithms.

Keywords- Data Warehouse, optimal queries, view selection, 0/1 knapsack.

مجله علمی پژوهشی رایانش نرم و فناوری اطلاعات

دانشگاه صنعتی نوشیروانی بابل

صفحه مجله: jscit.nit.ac.ir

جلد ۹، شماره ۳، پائیز ۱۳۹۹ ، صفحه ۱۶۳-۱۷۹

دریافت ۱۳۹۶/۰۸/۰۱، بازنگری: ۱۳۹۹/۰۱/۱۷، پذیرش: ۱۳۹۹/۰۲/۲۵

بهبود الگوریتم انتخاب دید در پایگاه داده‌ی تحلیلی با استفاده از الگوریتم جهش ترکیبی قورباغه در حل مساله کوله‌پشتی صفرویک

ریحانه صباغ گل [۱]، نگین دانشپور* [۲]

۱- دانشکده مهندسی کامپیوتر، دانشگاه تربیت دبیر شهید رجایی، لویزان، تهران، ایران.

*۲- دانشکده مهندسی کامپیوتر، دانشگاه تربیت دبیر شهید رجایی، لویزان، تهران، ایران.

[1] sabbagh.rsg@gmail.com, [2*] ndaneshpour@srttu.edu

* نشانی نویسنده مسئول: نگین دانشپور، تهران، لویزان، خیابان شعبانلو، دانشگاه تربیت دبیر شهید رجایی، دانشکده مهندسی کامپیوتر، کد پستی: ۱۶۷۸۸-۱۵۸۱۱.

چکیده- پایگاه داده‌ی تحلیلی، برای پاسخ‌گویی به پرس‌وجوهای تحلیلی طراحی می‌شود. داده‌های موجود در پایگاه داده‌ی تحلیلی، داده‌های تاریخی هستند. در پایگاه داده‌ی تحلیلی، زمان پاسخ‌گویی به پرس‌وجوهای تحلیلی، زمان زیادی است. بنابراین باید به دنبال روشی برای کاهش این مدت زمان بود. استفاده از دید، راه‌حل مناسبی برای کاهش زمان پاسخ‌گویی است. اما امکان ذخیره‌سازی تمام دیدهای ممکن وجود ندارد. از طرفی دیگر، ذخیره‌سازی دیدهای بهینه، یک مسئله‌ی NP-Complete می‌باشد. به این منظور، الگوریتم‌های انتخاب دید زیادی ارائه شده‌اند که از جمله‌ی این الگوریتم‌ها می‌توان به الگوریتم‌هایی اشاره کرد که دیدهای پرکاربرد را ذخیره می‌کنند. پرس‌وجوهایی که قبلا مورد استفاده‌ی پایگاه داده‌ی تحلیلی بوده‌اند، حاوی اطلاعات مهمی هستند که به احتمال زیاد در آینده نیز مورد استفاده خواهند بود. این مقاله، الگوریتمی برای ذخیره‌سازی دیدهای مناسب ارائه می‌دهد. این الگوریتم با استفاده از پرس‌وجوهای قبلی، دیدهای مناسب را یافته و آن‌ها را ذخیره می‌کند. این دیدها توانایی پاسخ‌گویی به بسیاری از پرس‌وجوهایی که در آینده اتفاق خواهند افتاد را دارند. این مقاله از الگوریتم جهش ترکیبی قورباغه برای یافتن دیدهای بهینه در حل کوله‌پشتی صفرویک استفاده کرده است که باعث بهبود روش‌های قبلی و کاهش زمان پاسخ به پرس‌وجوها شده‌است.

واژه‌های کلیدی: پایگاه داده‌ی تحلیلی، پرس‌وجوهای بهینه، ذخیره‌سازی دید، کوله‌پشتی صفرویک.

۱- مقدمه

روزانه در سراسر جهان حجم زیادی از داده‌ها تولید می‌شوند. این داده‌ها توسط مدیران برای تصمیم‌گیری استخراج می‌شوند. لازمه‌ی تصمیم‌گیری درست و به‌جا، داشتن اطلاعات تحلیلی به‌روز می‌باشد. بنابراین نتایجی که به ازای هر تغییر کوچک، باید مجددا از پایگاه‌داده استخراج شوند، کارایی بالایی ندارند و باعث اتلاف زمان زیادی می‌شود[۳۴]. برای دسترسی به این داده‌ها دو روش وجود دارد. روش اول روش "به محض تقاضا"[۱] و روش دوم روش "از

پیش"[۲] نامیده می‌شود. در روش اول، پس از اجرای پرس‌وجوی کاربر، داده‌ها از پایگاه داده‌های مختلف جمع‌آوری می‌شوند. در روش دوم، داده‌ها در منبع بزرگی به نام پایگاه داده‌ی تحلیلی[۳]، جمع‌آوری می‌شوند، سپس پرس‌وجوها با استفاده از آن، پاسخ داده می‌شوند [۱]. پایگاه داده‌ی تحلیلی یک منبع موضوع‌گرا[۴]، یکپارچه[۵]، تاریخی[۶] و پایدار[۷] می‌باشد که برای تصمیم‌گیری استفاده می‌شود. موضوع‌گرا بودن آن به این معنی است که یک پایگاه داده‌ی تحلیلی، راجع به یک موضوع ساخته می‌شود و به جای آن‌که روی عملیات روزانه‌ی یک سازمان تمرکز کند، روی مدل‌سازی و تحلیل داده‌ها برای تصمیم‌گیری تمرکز می‌کند. یکپارچگی پایگاه داده‌ی تحلیلی به این معنی است که پایگاه داده‌ی تحلیلی با استفاده از یکپارچه ساختن داده‌های چندین منبع داده‌ی مختلف ساخته می‌شود که این منابع داده معمولا غیر یکنواخت هستند. تاریخی بودن آن به این معنی است که داده‌های ذخیره شده در پایگاه داده‌ی تحلیلی مربوط به چندین سال هستند(مثلا ۵ تا ۱۰ سال). پایدار بودن آن به این معنی است که داده‌ها در آن خودبه خود از بین نخواهند رفت [۲].

پایگاه داده‌ی تحلیلی، منبع داده‌ای است که داده‌های آن از منابع مختلفی جمع‌آوری و یکپارچه شده‌است و برای تصمیم‌گیری از آن استفاده می‌شود. بنابراین باید بتواند به پرس‌وجوهای تحلیلی کاربران پاسخ دهد. در پاسخ‌گویی به پرس‌وجوهای تحلیلی کاربران، سرعت پاسخ‌گویی فاکتور مهمی می‌باشد.

پایگاه داده‌ی تحلیلی به منظور کاهش زمان پاسخ به پرس‌وجوهای تحلیلی، از دید ذخیره‌شده[۸] استفاده می‌کند. با توجه به محدودیت فضای ذخیره‌سازی، و نیز زمان زیاد به‌روز نگه‌داشتن دیدها، نمی‌توان تمام دیدها را ذخیره کرد [۳۲]. بنابراین باید از بین تمامی دیدهای ممکن، تنها مجموعه‌ای از دیدها را ذخیره نمود. به منظور انتخاب مجموعه‌ای از دیدها، الگوریتم‌های مختلفی ارائه شده است که برخی از آن‌ها برپایه پرس‌وجوهایی است که قبلا مورد استفاده‌ی پایگاه داده‌ی تحلیلی بوده‌اند. یکی از الگوریتم‌های انتخاب دید که از پرس‌وجوهای قبلی استفاده می‌کند بدین شرح است: این الگوریتم با استفاده از پرس‌وجوهای قبلی دیدهای مناسب را انتخاب می‌کند. این پرس‌وجوها حاوی اطلاعات مهمی هستند؛ زیرا به احتمال زیاد، در آینده نیز این پرس‌وجوها اتفاق خواهند افتاد. ابتدا پرس‌وجوهای قبلی خوشه‌بندی می‌شوند. سپس در هر خوشه پرس‌وجوهای پرتکرار به‌دست می‌آیند. در مرحله‌ی بعد با توجه به مقدار فضای در دسترس برای هر خوشه، پرس‌وجوهای مناسب انتخاب می‌شوند و در نهایت پرس‌وجوهای مناسب در هر خوشه پیوند[۹] می‌شوند [۳]. در آخرین مرحله، پرس و جوهای موجود در هر خوشه پیوند می شوند تا به ازای هر خوشه تنها یک دید به دست بیاید و آن دید را ذخیره کند. این الگوریتم به علت این‌که برای یافتن پرس‌وجوهای مناسب در هر خوشه از روش برنامه‌نویسی پویا استفاده کرده است زمان اجرای بالایی دارد. با توجه به این‌که در الگوریتم‌های انتخاب دید، مهم‌ترین فاکتور، زمان می‌باشد بنابراین در این مقاله الگوریتمی پیشنهاد شده‌است که این مشکل را رفع کرده و دارای سرعت بالاتری می‌باشد و در زمان کم‌تری به پرس‌وجوی تحلیلی پاسخ می‌دهد.

الگوریتم پیشنهادی ارائه شده در این مقاله با استفاده از الگوریتم جهش ترکیبی قورباغه، پرس‌وجوهای بهینه را در هر خوشه به دست می‌آورد. بدین معنی که در هر خوشه، با توجه به فضای موجود برای هر خوشه، با استفاده از الگوریتم جهش ترکیبی قورباغه، دیدهای مناسب را یافته و ذخیره می‌کند. به عبارت دیگر، منظور از پرس‌و-جوهای بهینه، پرس‌وجوهایی هستند که از بین پرس‌وجوهای پرتکرار، با توجه به فضای موجود در هر خوشه انتخاب می‌شوند. فضای موجود در هر خوشه با استفاده از کوله پشتی صفر و یک، مدل‌سازی ریاضی (به صورت معادلات) می‌شود و معادلات، با استفاده از الگوریتم جهش ترکیبی قورباغه حل می‌شوند. نوآوری این مقاله در یافتن پرس‌وجوهای مفید با استفاده از الگوریتم جهش ترکیبی قورباغه می‌باشد. به عبارت دیگر در مرحله‌ای که پرس و جوهای مناسب در هر خوشه یافته می‌شوند، با استفاده از الگوریتم جهش ترکیبی قورباغه، مساله کوله‌پشتی صفر و یک حل می‌شود و دیدهای مناسب در هر خوشه یافته می‌شود. بنابراین در هر خوشه، با توجه به فضای موجود برای هر خوشه فرضیات مساله کوله‌پشتی صفر و یک نوشته می‌شود و با استفاده از الگوریتم جهش ترکیبی قورباغه، کوله‌پشتی صفر و یک حل می‌شود و دیدهای مناسب یافته و ذخیره می‌شوند.

مزایای الگوریتم جهش ترکیبی قورباغه نسبت به سایر الگوریتم‌های فراابتکاری عبارتند از [۲۶] و [۳۲]:

۱- الگوریتم جهش ترکیبی قورباغه نسبت به سایر الگوریتم‌های فراابتکاری دارای دقت بالاتری می‌باشد.

۲- در الگوریتم جهش ترکیبی قورباغه، علاوه بر جستجوی محلی، در جستجوی سراسری نیز پیام ها مبادله می‌شوند. بدین ترتیب جستجوی محلی و سراسری، به خوبی در این الگوریتم ترکیب می‌شوند.

۳- همگرایی الگوریتم جهش ترکیبی قورباغه نسبت به سایر الگوریتم‌های فراابتکاری سریعتر و پایدارتر است.

۴- برخی از الگوریتم‌های فراابتکاری از جمله PSO به راحتی در نقاط بهینه‌ی محلی می‌افتند ولی این الگوریتم در نقاط بهینه‌ی محلی نمی‌افتد.

الگوریتم ارائه شده در [۳] و الگوریتم پیشنهادی پیاده‌سازی شده‌اند و مشاهده شده‌است که الگوریتم پیشنهادی نسبت به الگوریتم [۳] بهبود داشته است.

ساختار این مقاله به شرح زیر است: در بخش ۲ مقالات و الگوریتم‌های مرتبط با این مقاله ارائه می‌شود. در بخش ۳ الگوریتم انتخاب دیدی که این مقاله ارائه می‌دهد بیان شده‌است. بخش ۴، به شرح پیاده‌سازی الگوریتم می‌پردازد و بخش ۵ به نتیجه‌گیری می‌پردازد.

۲- پیشینه‌ی تحقیق

اکثر الگوریتم‌های انتخاب دید با توجه به بیشینه کردن سود، دیدهای مناسب را انتخاب می‌کنند. از جمله‌ی این الگوریتم‌ها می‌توان به الگوریتم‌های حریصانه اشاره کرد [۴] و [۵]. این الگوریتم‌ها بهترین دیدها را با توجه به فضای موجود انتخاب می‌کنند. البته این الگوریتم‌ها برای مسائل با ابعاد کم مناسب هستند ولی برای مسائل با ابعاد بزرگ مناسب نیستند.

برای رفع این مشکل الگوریتم‌های ژنیتک ارائه شده‌اند [۶] تا [۹]. این الگوریتم‌ها نسبت به الگوریتم‌های تکاملی مشابه، دیدهای مناسب‌تری انتخاب می‌کنند اما دارای زمان اجرای بالایی هستند. ورودی برخی از الگوریتم‌ها، بارکاری[۱۰] می‌باشد. به این معنی که این الگوریتم‌ها از پرس‌وجوهای قبلی استفاده می‌کنند؛ زیرا پرس‌وجوهای آینده، بسیار شبیه به پرس‌وجوهای قبلی هستند. برخی از این الگوریتم‌ها، زیرعبارت‌های مشترک را در پرس‌وجوهای بارکاری یافته و آنها را به عنوان دید ذخیره می‌کنند [۱۰] و [۱۱]. البته این الگوریتم‌ها، با توجه به محاسبات سنگین و پیچیده، برای مسائل با ابعاد بزرگ قابل اجرا نمی‌باشند.

برخی از الگوریتم‌ها مسئله را به صورت گراف مدل می‌کنند و بر اساس آن تصمیم‌گیری می‌کنند. برخی از این گراف ها عبارتست از گراف AND-OR و یا گراف MVPP[11]. درواقع پس از رسم گراف و با توجه به تابع هزینه‌ی تعریف شده، دیدهای مناسب را انتخاب می‌کنند [۱۴] تا [۱۲]. با توجه به این‌که باید گراف مورد نظر ترسیم شود، این الگوریتم‌ها برای مسائل با ابعاد بزرگ مناسب نمی‌باشند.

برخی از روش‌ها با بیشینه کردن سود و کمینه کردن هزینه، دیدهای مناسب را برای ذخیره‌سازی انتخاب می‌کنند [۱۵]. برخی از فاکتورهایی که در تعریف تابع سود مؤثر هستند عبارتند از: فرکانس تکرار هر پرس‌وجو، زمان اجرای هر پرس‌وجو، تعداد پیوندها و مجتمع‌های[۱۲] موجود در هر پرس‌وجو، هزینه‌ی نگه‌داری دید، فرکانس به‌روزرسانی جداول پایه، تعداد درج‌ها، تعداد تغییرات و تعداد حذف‌ها. با استفاده از روشی که در [۱۵] مطرح شده‌است، گراف بدون دور مستقیم[۱۳] مربوط به هر دید رسم می‌شود. منظور از رسم گراف، بازنمایی گراف در حافظه می باشد. یال های این گراف، دیدها می‌باشند. سپس کوتاه‌ترین مسیر در آن گراف به دست می آید. بدین ترتیب دیدهای با بیش‌ترین سود برای ذخیره‌سازی انتخاب می‌شوند. البته این الگوریتم برای مسائل با ابعاد کم مناسب می‌باشد و برای مسائل با ابعاد بزرگ، پیچیده می‌باشد.

برخی از الگوریتم‌ها، با پیش‌بینی کردن پرس‌وجوی بعدی، دیدهای لازم برای پاسخ‌گویی به آن پرس‌وجو را ذخیره می‌کنند [۱۶]. پیش‌بینی کردن پرس‌وجوی بعدی باعث می‌شود دید مورد نیاز برای آن پرس‌وجو، به درستی ذخیره گردد اما اگر تابع پیش‌بینی، پرس‌وجوی بعدی را اشتباه پیش‌بینی نماید، زمان زیادی برای پاسخ‌گویی به آن پرس‌وجو صرف خواهد شد.

برخی از الگوریتم‌ها از مدل‌های ریاضی استفاده می‌کنند. این الگوریتم‌ها مسئله‌ی انتخاب دید را به معادلات ریاضی تبدیل می‌کنند و با حل کردن معادلات، دیدهای مناسب را برای ذخیره‌سازی انتخاب می‌شوند. برخی از این الگوریتم‌ها با استفاده از مسئله‌ی ارضای محدودیت‌ها[۱۴] [۱۷] تا [۱۹] و برخی دیگر با استفاده از روش برنامه‌نویسی صحیح[۱۵] [۲۰] و [۲۱] مسئله را مدل می‌کنند. این الگوریتم‌ها با توجه به این که مسئله را با استفاده از معادلات ریاضی حل می‌کنند، جزء قدرتمندترین روشهای حل مسائل هستند. اما با توجه به این‌که محدودیت های مورد نظر را برای هر پرس‌وجو در نظرمی گیرند، برای مسائل با ابعاد بزرگ پیچیده می‌باشد و دارای زمان اجرای بالایی خواهند بود.

الگوریتم بیان شده در[۲۸] با استفاده از جفت‌گیری زنبور عسل دیدهای مناسب را انتخاب و ذخیره می‌کند. این الگوریتم در مقایسه با الگوریتم حریصانه دارای کارایی بالاتری می‌باشد.

الگوریتم بیان شده در [۳۰] با استفاده از روش‌های داده‌کاوی پرس و جوهای پرتکرار را می‌یابد. در این الگوریتم با تغییر داده‌ها، تمامی جدول ها نیز به‌روزرسانی می‌شوند. این الگوریتم نسبت به الگوریتم حریصانه بهتر عمل می‌کند.

الگوریتم معرفی شده در [۲۹] تعدادی دید نماینده[۱۶] انتخاب می‌کند، سپس عملیات حذف و اضافه کردن دیدها انجام می‌شود؛ به این صورت که دیدهایی که هزینه‌ی بالایی دارند، حذف شده و دیدهایی که هزینه‌ی کمتری دارند اضافه می‌شوند. در صورتیکه این

الگوریتم دیدهای نماینده را به درستی انتخاب نکند، زمان پاسخ الگوریتم بسیار بالا خواهد بود.

برخی از الگوریتم‌ها، بهبود یافته‌ی الگوریتم حریصانه هستند. این الگوریتم‌ها علاوه بر معیارهای موجود در الگوریتم حریصانه، بر اساس سایز پرس‌وجو، فرکانس تکرار پرس‌وجوها و میزان توانایی پرس‌وجو در تصمیم‌گیری[17] دیدهای مناسب را انتخاب کرده و ذخیره‌سازی می‌کنند [۲۲]. این الگوریتم‌ها با توجه به این که از شبکه‌ای از کعب[18] ها استفاده می‌کنند، برای مسائل با ابعاد کوچک مناسب هستند.

الگوریتم بیان شده در [۳۳] انتخاب دید را با استفاده از الگوریتم SPSOVSA که بر گرفته از الگوریتم ازدحام ذرات (PSO) است، انجام می‌دهد. این الگوریتم شبکه ای از کعب ها را به عنوان ورودی می گیرد. در ابتدا جمعیت اولیه‌ای از دیدها را تولید می‌کند و در هر مرحله با به روزرسانی موقعیت هر ذره (دید)، دیدهای مناسب برای ذخیره‌سازی به دست می‌آیند.

الگوریتم انتخاب دید بیان شده در [۳۲]، با استفاده از پرس‌وجوهای قبلی و الگوریتم Index-BittableFI، پرس‌وجوهای پرتکرار را به دست آورده و با توجه به آن‌ها، دیدهای مناسب را ذخیره‌سازی می‌کند.

الگوریتم تکاملی بیان شده در [۳۵] که EGTMVS[19] نامیده می شود، مسئله‌ی انتخاب دید را به عنوان یک بازی تکاملی مدل می‌کند. به این صورت که در ابتدا یک جمعیت تصادفی تولید کرده که هر کدام از اعضای این جمعیت به عنوان یک بازیکن عمل می‌کند. برای هر بازیکن سه استراتژی تعریف شده است، که هر کدام از آن‌ها به هزینه‌ی نگهداری دید، هزینه‌ی پردازش دید و هزینه‌ی پردازش گره قبلی مربوط می‌باشد. در طی بازی، هر کدام از بازیکن‌ها باید یک استراتژی که دارای هزینه بالاتری است را انتخاب کند. در پایان هر مرحله، جمعیت به‌روزرسانی شده و در پایان بازی، بهترین جمعیت انتخاب می‌شود. یکی از مزایای این الگوریتم این است که الگوریتم همگراست و برای پایگاه‌داده‌های تحلیلی حجیم، مناسب است.

الگوریتم [۳۶]، مسئله‌ی انتخاب دید را با استفاده از ترکیب روش رتبه‌ی تصادفی[20] و عقبگرد[21] حل می کند. مزایای این الگوریتم این است که استفاده از روش عقبگرد، باعث افزایش سرعت استخراج دیدها می‌شود و از طرفی دیگر، روش رتبه‌ی تصادفی برای مدیریت محدودیت‌ها و پارامترها مناسب می‌باشد که درنتیجه زمان اجرای پرس‌وجو کاهش داده می‌شود.

الگوریتم CROMVS [۳۷] با شبیه‌سازی صخره‌های مرجانی، دیدهای مناسب را ذخیره می‌کند. این الگوریتم از گراف MVPP نیز استفاده می‌کند. در ابتدا جمعیت تصادفی از مرجان ها انتخاب می‌شوند و با توجه به تابع سازگاری، بعد از هر مرحله، راه حل‌های بهتر انتخاب می‌شوند و رشد می‌کنند. مزیت مهم این الگوریتم رشد سریع همگرایی آن می‌باشد.

الگوریتم GTMV [۳۸]، با استفاده از مدل‌سازی مسئله به صورت یک بازی، دیدهای مناسب را پیدا می‌کند. به این صورت که هزینه پردازش پرس‌وجو و هزینه نگه‌داری پرس‌وجو با هم رقابت می‌کنند و این رقابت تا جایی ادامه پیدا می‌کند که این دو هزینه به موازنه برسند. بنابراین بعد از موازنه، دیدهای مناسب به صورت مستقیم یا غیر مستقیم انتخاب میشوند. اما تنها نقطه‌ی منفی این الگوریتم این است که این الگوریتم پویا[22] نمی‌باشد.

الگوریتم بیان شده در [۳۹]، K دید بالای شبکه‌ای از کعب‌ها را انتخاب می کند با به‌روزرسانی دیدهای انتخاب شده، دیدهای مناسب برای ذخیره‌سازی را انتخاب می‌کند. همچنین این الگوریتم، علاوه بر هزینه ذخیره‌سازی دیدها، به هزینه‌ی ارزیابی دیدهای ذخیره نشده نیز توجه می‌کند.

الگوریتم توضیح داده شده در [۴۰] از روش حریصانه استفاده می‌کند. به این صورت که از نتایج پاسخ به پرس‌وجوهای قبلی، استفاده کرده تا پاسخ هر پرس‌وجو را به‌دست‌آورد. در نتیجه این الگوریتم، زمان اجرای پرس‌وجو را تا حد زیادی کاهش داده‌است.

در برخی دیگر از روش‌ها با استفاده از الگوریتم‌های خوشه‌بندی و داده‌کاوی، دیدهای مناسب انتخاب می‌شوند [۳] و [۲۳] تا [۲۶]. در این الگوریتم‌ها از پرس‌وجوهای قبلی استفاده می‌شود. علت این امر این است که به احتمال زیاد در آینده نیز، این پرس‌وجوها اتفاق خواهند افتاد. این روش‌ها از الگوریتم MVCF[23] استفاده می‌کنند. این الگوریتم از چهار مرحله تشکیل شده‌است:

الف) ابتدا با استفاده از روش‌های خوشه‌بندی، پرس‌وجوهای قبلی خوشه‌بندی می‌شوند. در این مقاله با استفاده از روش خوشه‌بندی سلسله‌مراتبی پرس‌وجوها خوشه‌بندی می‌شوند.

ب) سپس با استفاده از روش‌های داده‌کاوی برای یافتن آیتم‌های پرتکرار، در هر خوشه پرس‌وجوهای پرتکرار یافته‌می‌شوند. در این مرحله از الگوریتم Apriori برای یافتن پرس‌وجوهای پرتکرار استفاده شده است.

ج) سپس در هر خوشه، پرس‌وجوهای بهینه با استفاده از مساله کوله

پشتی صفر و یک انتخاب می‌شوند.

د) در این مرحله، در هر خوشه پرس‌وجوهای بهینه پیوند می‌شوند تا به ازای هر خوشه، یک دید به‌دست آید و آن دید ذخیره می‌شود.

در بخش سوم این مراحل به تفضیل توضیح داده می‌شود.

در مرحله سوم برای یافتن پرس و جو های مناسب، می‌توان از استراتژی های حریصانه استفاده کرد؛ زیرا هر کدام از خوشه ها فضایی محدود در اختیار دارند و با توجه به استراتژی های حریصانه می توان از بین پرس و جوهای موجود در هر خوشه، پرس و جوهایی را انتخاب کرد که بیشترین سود و کمترین فضا را اختیار کنند. بنابراین با توجه به این توصیف، می توان از مساله کوله پشتی صفر و یک استفاده کرد.

بنابراین در[۳] و [۲۳] تا [۲۶] برای یافتن پرس‌وجوهای مفید در هر خوشه، از کوله‌پشتی صفرویک استفاده شده‌است و برای حل معادلات کوله‌پشتی صفرویک از روش برنامه‌نویسی پویا استفاده شده‌است. این روش به دلیل این‌که دارای مصرف زمانی بالایی است کارا نمی‌باشد. علاوه بر این، این روش برای داده‌های حجیم بهینه نمی‌باشد و زمان اجرای الگوریتم بسیار طولانی خواهد بود.

بنابراین در این مقاله با توجه به اهمیت زمان پاسخ‌گویی به پرس-وجوها در پایگاه داده تحلیلی، برای حل کوله‌پشتی صفرویک از الگوریتم جهش ترکیبی قورباغه [۲۷] استفاده شده است تا الگوریتم انتخاب دید دارای کارایی بالاتر و مصرف زمانی پایین‌تری باشد.

۳- الگوریتم انتخاب دید پیشنهادی

در این بخش الگوریتم ارائه شده در این مقاله برای انتخاب دید توصیف می‌شود. این الگوریتم از روش MVCF که در بخش کارهای مرتبط توضیح داده شده‌است، استفاده می‌کند.

در این الگوریتم از پرس‌وجوهایی که قبلا مورد استفاده‌ی پایگاه داده‌ی تحلیلی بوده‌اند، استفاده شده‌است. زیرا این پرس‌وجوها حاوی اطلاعات مفیدی هستند که به احتمال زیاد در آینده نیز مورد استفاده خواهند بود [۳۲]. در الگوریتم پیشنهادی این مقاله، روش یافتن پرس‌وجوهای بهینه در الگوریتم MVCF بیان شده در [۳] و [۲۳] تا [۲۶] بهبود داده شده‌است که باعث بهبود نهایی الگوریتم انتخاب دید می‌شود. الگوریتم یافتن پرس‌وجوهای بهینه که در منبع [۳] و [۲۵] بیان شده، با توجه به نام نویسندگان و سال انتشار، الگوریتم KD2013 نامیده می‌شود. الگوریتم پیشنهادی در این مقاله نیز با توجه به این‌که از الگوریتم جهش ترکیبی قورباغه استفاده کرده است، الگوریتم SRTTU-Frog نامیده شده‌است.

الگوریتم پیشنهادی در ادامه توضیح داده می‌شود.

ابتدا پرس‌وجوهای قبلی خوشه‌بندی می‌شوند. به هریک از این خوشه‌ها[۲۴]، دسته یا دامنه[۲۵] گفته می‌شود. برای خوشه‌بندی پرس‌وجوها از معیار شباهت جاکارد[۲۶] [۲۵] استفاده می‌شود [۳۲].

فرض می‌شود دو خوشه به نام‌های C_i وC_j وجود دارد، در این صورت:

$Q_{i1},Q_{i2},\dots Q_{im}$ پرس‌وجوهای متعلق به خوشه C_i هستند و $R_{i1},R_{i2},\dots,R_{ir}$ جدول‌های موردنیاز برای پرس‌وجوهای $Q_{i1},Q_{i2},\dots Q_{im}$ می‌باشند و $Q_{j1},Q_{j2},\dots Q_{jm}$ پرس‌وجوهای متعلق به خوشه C_j هستند و $R_{j1},R_{j2},\dots,R_{jr}$ جدول‌های موردنیاز برای پرس‌وجوهای $Q_{j1},Q_{j2},\dots Q_{jm}$ می‌باشند. بنابراین معیار شباهت جاکارد با استفاده از رابطه‌ی (۱) تعریف می‌شود.

$$\mathrm{SIM}(C_i,C_j)=\frac{\left|\{R_{i1},R_{i2},\dots,R_{ir}\}\cap\{R_{j1},R_{j2},\dots,R_{js}\}\right|}{\left|\{R_{i1},R_{i2},\dots,R_{ir}\}\cup\{R_{j1},R_{j2},\dots,R_{js}\}\right|}=\frac{\left|\mathrm{Rel}(C_i)\cap\mathrm{Rel}(C_j)\right|}{\left|\mathrm{Rel}(C_i)\cup\mathrm{Rel}(C_j)\right|} \quad (۱)$$

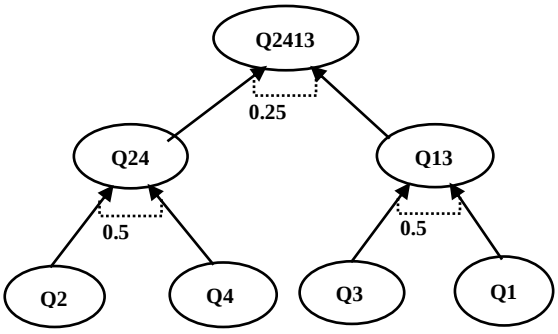
$Rel(C_i)$ جدول‌هایی هستند که در خوشه C_i وجود دارند. این معیار هر چه به یک نزدیک‌تر باشد، دو پرس‌وجو به هم شبیه‌تر هستند. با توجه به تعریف معیار جاکارد، هر چه پرس‌وجوها به هم شبیه‌تر باشند، معیار جاکارد به ۱ نزدیکتر است. از طرف دیگر، هدف خوشه‌بندی این است که پرس‌وجوهایی که بیشتر به هم شبیه‌اند در یک خوشه قرار بگیرند، بنابراین از معیار جاکارد برای یافتن پرس-وجوهای شبیه به هم برای قرار دادن در یک خوشه استفاده شده است.

برای یافتن خوشه‌هایی که دارای پرس‌وجوهای پرتکرار هستند ابتدا باید الگوریتم ادغام پرس‌وجوها[۲۷] [۲۵] اجرا شود تا براساس معیار شباهت جاکارد، پرس‌وجوها با هم پیوند شوند. سپس براساس ترتیب پیوند پرس‌وجوها درخت آن را تشکیل داده و خوشه‌ها از روی درخت یافته شوند [۳۲].

برای ساخت درخت از روی الگوریتم ادغام پرس‌وجوها، روند ساخت خوشه‌ها به صورت درخت رسم می‌شود. با توجه به الگوریتم ادغام پرس‌وجوها، خوشه‌ها براساس بیش‌ترین میزان شباهت به هم از پایین به بالا پیوند می‌شوند تا این‌که در آخرین مرحله همه‌ی پرس‌وجوها در یک خوشه قرار می‌گیرند و درنهایت درخت ساخته می‌شود [۳۲]. نمونه‌ای از این درخت در شکل ۱ آمده‌است.

پس از ساخت درخت، گره‌های درخت از بالا به پایین ملاقات[۲۸] می‌شوند تا زمانی که خوشه‌ای پیدا شود که درجه‌ی شباهتی بیش‌تر

از حداقل شباهت حدآستانه[۲۹] داشته باشد. این خوشه‌ها نشان‌دهنده‌ی دامنه‌ها خواهند بود. به عنوان مثال در شکل ۱، اگر حدآستانه ۰.۴ باشد خوشه‌ها Q24 و Q13خواهندبود. توجه شود که Q13 به این معنی است که این خوشه شامل Q1 وQ3 است.



شکل ۱- درخت ساخته شده از روی الگوریتم ادغام پرس‌وجوها

پس از یافتن خوشه‌ها، باید در هر خوشه پرس‌وجوهای پرتکرار یافته شوند. بدین منظور از الگوریتم Apriori استفاده می‌شود [۲۵].

پس از به‌دست‌آوردن پرس‌وجوهای پرتکرار در هر خوشه، باید پرس‌وجوهای بهینه در هر خوشه به‌دست آیند [۳] و [۳۲]. این پرس‌وجوهای بهینه با توجه به فضای موجود برای هر خوشه انتخاب می‌شوند.

این مسئله مانند مسئله‌ی کوله‌پشتی می‌باشد، فرض می‌شود فضایی که می‌تواند برای نگه‌داری دیدهای هر خوشه استفاده شود، فضای کوله‌پشتی باشد، در این صورت نمی‌توان تمام پرس‌وجوها را در کوله‌پشتی قرار داد و با توجه به محدودیت ظرفیت کوله‌پشتی، پرس‌وجوها را در آن می‌توان قرار داد. پس باید با توجه به فضایی که هر خوشه می‌تواند در اختیار داشته باشد، پرس‌وجوهای بهینه را انتخاب نمود [۳].

فرض کنید پرس‌وجوهای $Q_1,Q_2,...,Q_n$ پرس‌وجوهای پرتکرار در خوشه‌ی k ام باشد. برای هر پرس‌وجوی Q_i، دو متغیر P_i و S_i تعریف می‌شود. P_i، سود آن و S_i سایز پرس‌وجوی Qi می‌باشد، که در واقع S_i، وزن پرس و جو می باشد. نتیجه‌ی اجرای پرس‌وجوی Qi روی پایگاه داده‌ی تحلیلی، جدولی می‌باشد. این جدول Ti نامیده می‌شود. تعداد رکوردهای جدول Ti در متغیر P_i ذخیره می‌شود. پرس‌وجوی Qi در قسمت from خود دارای تعدادی نام جدول می‌باشد، تعداد کل رکوردهای این جداول محاسبه و به متغیر S_i اختصاص داده می‌شود. بدین ترتیب برای هر Qi، متغیرهای P_i و S_i به دست آمده است و معادلات مسئله‌ی کوله‌پشتی صفرویک به صورت بیان شده در رابطه‌ی (۲) خواهد بود.

$$\begin{aligned} &Maximize\ P_1Q_1 + P_2Q_2 + ... + P_nQ_n \\ &Subject\ to\ S_1Q_1 + S_2Q_2 + ... + S_nQ_n \le S \end{aligned} \qquad (۲)$$

که در آن S_i به صورت بیان شده در رابطه‌ی(۳) تعریف می‌شود.

$$S_i = \sum_{R \in Q_i} t(R) \qquad (۳)$$

در این رابطه $t(R)$ تعداد رکوردهای جدول R_i می‌باشد. منظور از جدول R جدولی می‌باشد که با پرس‌وجوی Qi در ارتباط است. به عبارتی دیگر R جزء جداولی می‌باشد که در قسمت from پرس‌وجوی Qi قرار دارند. مقدار Qi می‌تواند صفر یا یک باشد، در صورتی‌که پرس‌وجوی i انتخاب شود، مقدار Qi یک خواهد بود درغیر اینصورت مقدار آن صفر خواهد بود [۲۶]. به عنوان مثال نمونه‌ای از معادلات کوله‌پشتی صفرویک در رابطه‌ی (۴) بیان شده‌است.

$$\begin{aligned} &\text{Maximise } 13Q_1 + 15Q_7 + 28Q_{16} + 35Q_{17} + 32Q_{20} \\ &\text{subject to} \\ &\quad 120Q_1 + 250Q_7 + 430Q_{16} + 310Q_{17} + 360Q_{20} \le 1000 \\ &\quad \text{and} \\ &\quad Q_i s = 0 \text{ or } 1 \text{ where } i = 1,7,16,17,20 \end{aligned} \qquad (۴)$$

معادلات به دست آمده در رابطه‌ی (۴) معادلات کوله‌پشتی صفرویک می‌باشد که با استفاده از الگوریتم جهش ترکیبی قورباغه حل می‌شود و مقادیر Qi به دست می‌آید.

ابتدا الگوریتم جهش ترکیبی قورباغه مورد بررسی و ارائه قرار می‌گیرد سپس الگوریتم جهش ترکیبی قورباغه‌ی تغییر یافته برای حل مسئله‌ی کوله‌پشتی صفرویک بیان می‌شود. قسمت الف الگوریتم جهش قورباغه و قسمت ب به بیان این الگوریتم برای حل کوله‌پشتی صفرویک می‌پردازد:

الف) الگوریتم جهش ترکیبی قورباغه:

الگوریتم جهش ترکیبی قورباغه [۲۷] مبتنی بر رفتار گروهی قورباغه‌ها هنگام یافتن مکانی با بیش‌ترین مقدار غذا می‌باشد. شرح این الگوریتم به صورت زیر می‌باشد:

در این الگوریتم جمعیت[۳۰] شامل مجموعه‌ای از قورباغه‌ها (راه‌حل‌ها[۳۱]) می‌باشد. این جمعیت به تعدادی زیرمجموعه تقسیم می‌شود. این زیرمجموعه‌ها ممپلکس[۳۲] نامیده می‌شوند. ممپلکس‌های مختلف به عنوان گروه‌های مختلفی از قورباغه‌ها شناخته می‌شوند که هر ممپلکس دارای فرهنگ[۳۳] مختلفی می‌باشد. در هر ممپلکس، هر قورباغه دارای نظری می‌باشد که می‌تواند متأثر

از نظر سایر قورباغه‌ها باشد و نظرات قورباغه‌ها از طریق فرایند تکامل تدریجی ممپلکسها[۳۴]، رشد و نمو می‌کند. پس از این‌که این فرایند به تعداد گام مشخصی اجرا شد، نظرات بین ممپلکس‌ها منتقل می‌شوند. فرایند جستجوی محلی، که درون هر ممپلکس انجام می‌شود، و فرایند انتقال نظرات بین ممپلکس‌ها تا زمانی ادامه پیدا می‌کند که شرط خاتمه برقرار شود. این الگوریتم در شکل ۲ نشان داده شده‌است.

جمعیت اولیه‌ی P از قورباغه‌ها به صورت تصادفی تولید می‌شود. برای مسئله‌ای با S بعد (یعنی دارای S متغیر باشد)، قورباغه‌ی i به صورت $X_i=(X_{i1},X_{i2},\ldots,X_{iS})$ نمایش داده می‌شود. سپس قورباغه‌ها بر اساس شایستگی[۳۵] به صورت نزولی مرتب می‌شوند.

تابع شایستگی $f(i)$ در رابطه‌ی شماره‌ی (۵) تعریف می‌شود.

$$f(i)=\sum_{j=1}^{S} X_{ij}P_j \qquad (۵)$$

که در رابطه‌ی فوق، P_j سود پرس‌وجوی jام می‌باشد. به عبارت دیگر تابع $f(i)$ میزان شایستگی قورباغه‌ی iام را با توجه به سود حاصل از انتخاب آن قورباغه مشخص می‌کند.

Procedure Shuffled Frog Leaping Algorithm

Input: number of frogs *P*; number of memeplexes *m*; number of generation for each memeplex before shuffling *n*; number of shuffling iterations *it*; and maximum number of iterations *iMax*.

Output: best solution

Generate random population of *P* solutions (frogs)

for each individual $i \in P$ **do**

  Calculate fitness(*i*);

end for

Sort the population *P* in descending order of their fitness;

Divide *P* into *m* memeplexes;

for each memeplex **do**

  Determine the best and worst frogs;

  Improve the worst frog position;

  Repeat for a specific number of iterations;

end for

Combine the evolved memeplexes;

Sort the population *P* in descending order of their fitness;

if termination = true **then**

  Return best solution;

end if

شکل۲- الگوریتم جهش ترکیبی قورباغه [۲۷]

جمعیت قورباغه‌ها به m بخش (ممپلکس) تقسیم می‌شوند که هر بخش دارای n قورباغه می‌باشد (یعنی جمعیت قورباغه‌ها $P=m*n$ می‌باشد). در این فرایند قورباغه‌ی اول به ممپلکس اول اختصاص داده می‌شود، قورباغه‌ی دوم به ممپلکس دوم اختصاص داده می‌شود، قورباغه‌ی m ام به ممپلکس m ام و قورباغه‌ی $m+1$ ام به ممپلکس اول اختصاص داده می‌شود.

در هر ممپلکس، قورباغه‌ی دارای بهترین شایستگی با X_b و قورباغه‌ی دارای کم‌ترین شایستگی با X_w نشان داده می‌شود. در بین تمام ممپلکس‌ها قورباغه‌ی دارای بیش‌ترین شایستگی با X_g نشان داده می‌شود. سپس درون حلقه‌ای X_w بهبود داده می‌شود. روابط شماره‌ی (۶) و (۷) تابع بهبود X_w را نشان می‌دهد.

$$D_i = Rnad() \times (X_b - X_w) \qquad (۶)$$

$$X_w(new) = X_w + D_i - Di_{max_{max}} \qquad (۷)$$

که در آن D_i تغییرات جایگاه i امین قورباغه می‌باشد. رابطه‌ی شماره‌ی (۷) جایگاه جدید قورباغه‌ی X_w را نشان می‌دهد، که در آن Rand() تابع تولید اعداد تصادفی بین صفرویک می‌باشد و Dmax حداکثر میزان تغییرات ممکن در جایگاه قورباغه می‌باشد. در صورتیکه این فرایند جایگاه بهتری را تولید کند، $X_{w(new)}$ جایگزین X_w خواهد شد. درغیر این صورت در روابط شماره‌ی (۶) و (۷) X_g، جایگزین X_b می‌شود و این روابط تکرار می‌شوند. در صورتیکه $X_{w(new)}$ نسبت به X_w بهبودی نداشته باشد، راه‌حلی جدید، به صورت تصادفی، تولید می‌شود. این فرایند به اندازه‌ی تعداد مشخصی تکرار، تکرار می‌شود.

در قسمت (ب) الگوریتم جهش ترکیبی قورباغه برای حل کوله‌پشتی صفرویک بیان می‌شود.

ب) الگوریتم جهش ترکیبی قورباغه برای حل کوله‌پشتی صفرویک

الگوریتم جهش ترکیبی قورباغه به طور مستقیم نمی‌تواند کوله‌پشتی صفرویک را حل نماید، بنابراین الگوریتم جهش ترکیبی قورباغه، تغییر داده می‌شود تا بتواند کوله‌پشتی صفرویک را حل نماید [۲۷]. الگوریتم جهش ترکیبی قورباغه به سرعت همگرا می‌شود ولی گاهی اوقات در نقاط بهینه‌ی محلی می‌افتد. برای حل این مشکل از تابعی برای به هم ریختن قورباغه‌ها استفاده می‌شود. تغییرات الگوریتم جهش ترکیبی قورباغه برای حل کوله‌پشتی صفرویک به طور کامل در زیر بیان شده‌است:

۱. ساخت جمعیت اولیه‌ی قورباغه:

برای این‌که بتوان کوله‌پشتی صفرویک را با استفاده از الگوریتم

جهش ترکیبی قورباغه حل کرد باید هر قورباغه را به صورت یک عدد n-بیتی در نظر گرفت، به طوریکه n، تعداد ابعاد مسئله‌ی کوله‌پشتی صفرویک می‌باشد.

۲. گسسته کردن متغیرها:

با توجه به این‌که مسئله‌ی کوله‌پشتی صفرویک، مسئله‌ای گسسته است پس باید رابطه‌ی شماره‌ی (۷) را گسسته نمود. رابطه‌ی شماره‌ی (۸) نحوه‌ی گسسته نمودن جایگاه $X_{w(new)}$ را نشان می‌دهد:

$$t = 1/(1+\exp(-D))$$

$$Xw(new) = \begin{cases} 0 & \text{if} \quad t \le \alpha \\ X_w & \text{if} \quad \alpha < t \le \frac{1}{2}(1+\alpha) \\ 1 & \text{if} \quad t \ge \frac{1}{2}(1+\alpha) \end{cases} \qquad (۸)$$

متغیر D در رابطه‌ی شماره‌ی (۶) بیان شده‌است. متغیر α، پارامتری ثابت است.

۳. بهینه‌سازی محدودیت‌ها

حل مسائلی که دارای محدودیت می‌باشند، نسبت به مسائلی که بدون محدودیت هستند، مشکل‌تر است. در مسائلی که دارای محدودیت هستند باید تعادلی بین پیدا کردن جواب بهینه و برقراری شرایط محدودیت‌ها پیدا کرد. یکی از این راه‌ها، استفاده از توابع اصلاحی[۳۶] است. تابع اصلاحی که در این مقاله در نظر گرفته شده‌است به این صورت عمل می‌کند که تمام اجزای کوله‌پشتی بر اساس نسبت سود بر وزن، به صورت نزولی مرتب شوند، سپس تابع اصلاحی آخرین جزء را برای حذف کردن انتخاب می‌کند. تابع اصلاحی در صورتی اجرا می‌شود که پس از به دست آوردن راه‌حل‌های بهینه توسط الگوریتم جهش ترکیبی قورباغه، مجموع وزن‌های راه‌حل‌ها بیش‌تر از حجم کوله‌پشتی باشد.

۴. جهش ژنتیکی[۳۷]

گاهی اوقات الگوریتم جهش ترکیبی قورباغه در نقاط بهینه‌ی محلی در دام می‌افتد. برای حل این مسئله، از تابع جهش ژنتیکی استفاده می‌شود. این تابع جمعیت اولیه‌ی قورباغه‌ها را کمی تغییر می‌دهد. این عمل باعث می‌شود که نقاط بهینه‌ی دیگر را هم جستجو کند.

۵. شرط خاتمه:

همانطور که در شکل شماره‌ی ۲ بیان شده‌است، $iMax$، حداکثر تعداد تکرارهای الگوریتم می‌باشد. اگر شرط خاتمه، عددی ثابت باشد، در مسائل بزرگ امکان دارد قبل از این‌که مسئله همگرا شود و جواب بهینه پیدا شود، الگوریتم خاتمه یابد. پس باید با توجه به شرایط مسئله و بر اساس تجربه، شرط خاتمه به صورت $\left\lceil \frac{iMax}{20} \right\rceil \le \Delta \le \left\lceil \frac{iMax}{10} \right\rceil$ انتخاب شود.

با توجه به ۵ مورد بیان شده می‌توان با استفاده از الگوریتم جهش ترکیبی قورباغه، مسئله‌ی کوله‌پشتی صفرویک را حل کرد. به عبارت دیگر معادلات بیان شده در رابطه‌ی (۱) با استفاده از الگوریتم جهش ترکیبی قورباغه حل می‌شود و پرس‌وجوهای مفید در هر خوشه به دست می‌آید. الگوریتم یافتن پرس‌وجوهای بهینه، با استفاده از کوله‌پشتی صفرویک در هر خوشه در شکل شماره‌ی ۳ نشان داده شده‌است.

بنابراین بازنمایی راه حل با توجه به شکل شماره‌ی ۳ به این صورت می‌باشد:

همانطور که در شکل شماره‌ی ۳ نشان داده شده‌است، به ازای هر خوشه، پرس‌وجوهای بهینه یافته می‌شود. در مرحله‌ی قبل، پرس‌وجوهای پرتکرار در هر خوشه به دست آمد. حال در این مرحله، با توجه به میزان فضای ذخیره‌سازی، پرس‌وجوهای بهینه در هر خوشه به دست می‌آیند. ابتدا باید برای هر خوشه به صورت زیر عمل کرد: با داشتن مقدار فضای موجود برای آن خوشه، مقادیر P_i و S_i برای تمام پرس‌وجوهای آن خوشه، معادلات کوله‌پشتی صفرویک با توجه به رابطه‌ی شماره‌ی (۲) نوشته می‌شود. اگر تعداد پرس‌وجوهای موجود خوشه‌ی مورد بررسی، d باشد و تعداد قورباغه‌ها P باشد، برای تولید جمعیت اولیه‌ی قورباغه‌ها، P تا عدد d-بیتی تولید می‌شود. سپس برای هر قورباغه، تابع شایستگی محاسبه می‌شود و قورباغه‌ها براساس مقدار شایستگی به صورت نزولی مرتب می‌شوند. سپس قورباغه‌ها به m تا ممپلکس تقسیم می‌شوند. برای هر ممپلکس، X_b و X_w به دست می آید. سپس X_w با توجه به رابطه‌ی شماره‌ی (۷) بهبود داده می‌شود. این عملیات به اندازه‌ی مقدار it، برای هر ممپلکس تکرار می‌شود. سپس تابع جهش ژنتیک اعمال می‌شود و جمعیت قورباغه‌ها به صورت نزولی مرتب می‌شود. در صورتیکه شرط خاتمه برقرار باشد از این حلقه خارج می‌شود. X_g به عنوان جواب الگوریتم جهش ترکیبی قورباغه می‌باشد. این عدد را به مبنای دو برده تا به صورت یک عدد d-بیتی نمایش داده شود. بیتی که مقدارش یک است، به این معنی است که پرس‌وجوی متناظر با آن باید ذخیره شود. به عنوان مثال فرض کنید عدد به صورت ۱۰۱۰۰۱۱۱ باشد. در این عدد بیت‌های

شماره‌ی ۰، ۲، ۵، ۶ و ۷، یک هستند و سایر بیت‌ها صفر هستند. بنابراین باید پرس‌وجوهای شماره‌ی ۰، ۲، ۵، ۶ و ۷ ذخیره شوند. با توجه به این‌که هر کدام از این پرس‌وجوها S_i دارند، اگر مجموع S_i این پرس‌وجوها از مقدار فضای موجود برای آن خوشه بیش‌تر باشد، آنگاه از تابع اصلاحی استفاده می‌شود.

Procedure Finding Optimal Queries

Input: number of frogs *P*; number of memeplexes *m*; number of generation for each memeplex before shuffling *n*; number of shuffling iterations *it*, maximum number of iterations *iMax*; and Dimensions of the 0-1 Knapsack problem *d*

Output: best solution

for each cluster **do**

Calculate equations explained in Eq. (۰) like Eq. (٤);

//Solve the 0-1 knapsack problem equations:

Generate random population of *P=n*m* solutions (frogs), so that each solution is a *d*-bit binary number;

for each individual $i \in P$ **do**

Calculate fitness(*i*);

end for

do

if termination = true **then**

Return best solution;

end if

Sort the population *P* in descending order of their fitness;

Divide *P* into *m* memeplexes;

for each memeplex **do**

Determine the best and worst frogs;

Improve the worst frog position by using Eq. (6v);

Repeat for a specific number of iterations;

end for

Combine the evolved memeplexes;

Apply genetic mutation on population;

Sort the population *P* in descending order of their fitness;

while (termination = False)

Find the optimal queries according to best solution;

if the best solution is an infeasible solution **then**

Execute repair methods to find optimal queries;

end if

end for

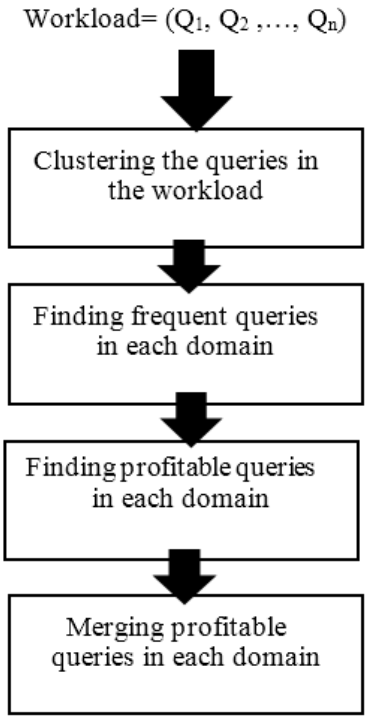
شکل۳- الگوریتم یافتن پرس‌وجوهای مفید

پس از به‌دست‌آوردن پرس‌وجوهای مفید در هر خوشه، باید این پرس‌وجوها پیوند شوند تا به ازای هر خوشه، یک دید ذخیره‌شود. هدف از این مرحله این است که برای هر دامنه، یک پرس‌وجوی بهینه ذخیره شود [۲۳]. در این مرحله جداول مربوط به هر پرس‌وجو در هر خوشه، Natural Outer Join می‌شوند. در نهایت برای هر دامنه یک دید به‌دست می‌آید و آن دید ذخیره می‌شود. شکل شماره‌ی ۵ الگوریتم پیشنهادی را به صورت فلوچارت نشان می‌دهد.

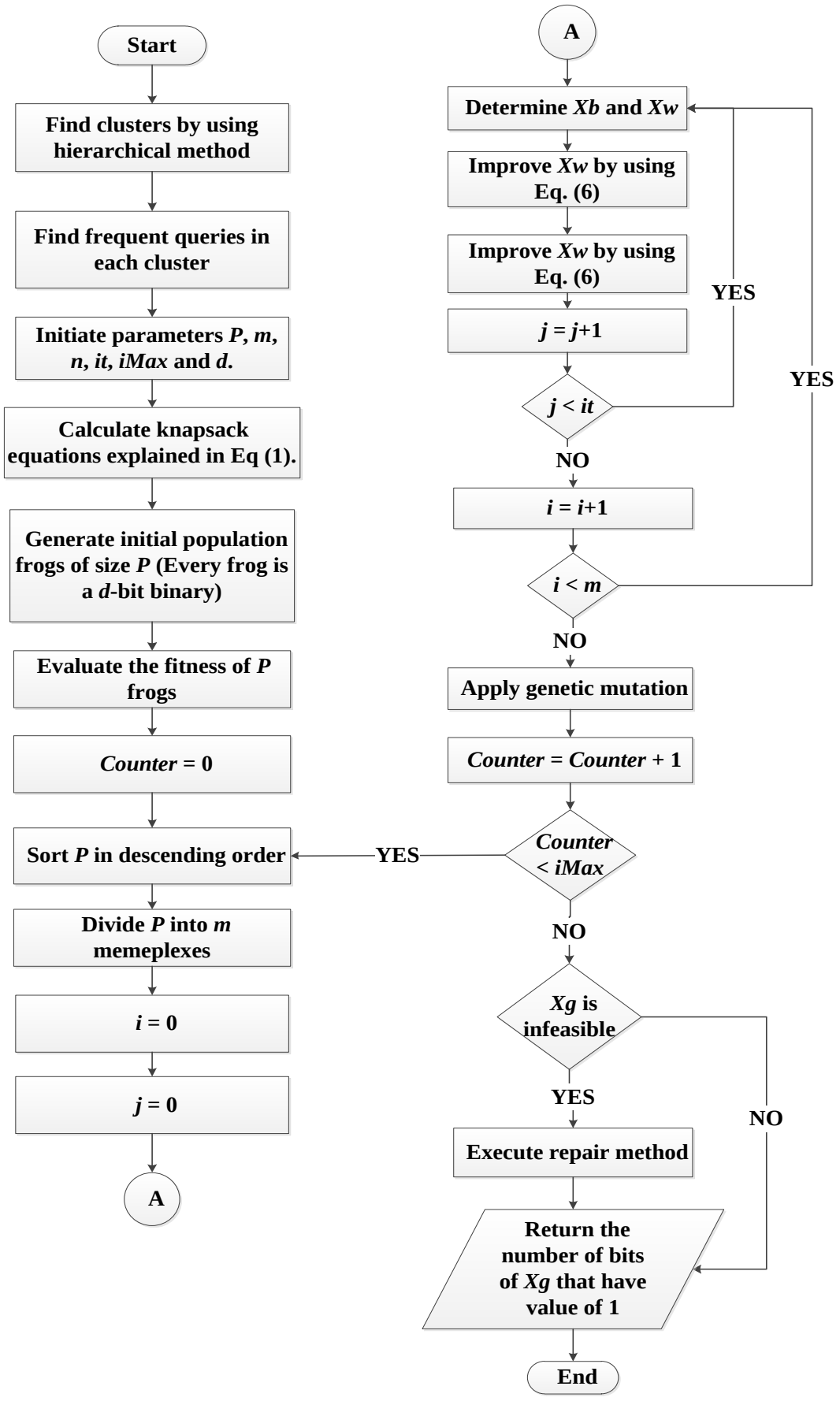
فلوچارت مربوط به شکل شماره‌ی ۴و ۵ نشان‌دهنده‌ی مراحل الگوریتم پیشنهادی می‌باشد. شکل شماره ۴ مراحل را به صورت کلی نشان می‌دهد و در شکل شماره ۵ مراحل الگوریتم با جزئیات نشان داده شده است.

۴- نتایج شبیه‌سازی

در این قسمت به بررسی نتایج و آزمایشات انجام شده پرداخته می‌شود. الگوریتم پیشنهادی این مقاله(SRTTU-Frog)، با الگوریتم‌های KD2013، SRTTU-2015 [۳۲]، SPSOVSA [۳۳] وEGTMVS [۳۵] مقایسه می‌شود. الگوریتم‌های مذکور با استفاده از نرم افزار Microsoft Visual Studio پیاده‌سازی شده‌است. پایگاه داده‌ی مورد استفاده برای پیاده‌سازی الگوریتم‌ها، Microsoft SQL Server می‌باشد.



شکل۴- فلوچارت کلی متد MVCF



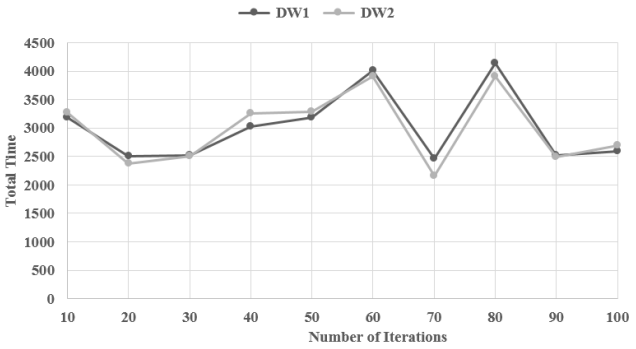
شکل۵- فلوچارت الگوریتم پیشنهادی SRTTU-Frog

داده‌های مورد استفاده برای الگوریتم‌های فوق به صورت تصادفی توسط نرم افزار Microsoft Visual Studio تولید شده‌است. با توجه به این‌که داده‌ها به صورت تصادفی تولید شده‌اند، پس در آزمایشات صورت گرفته، نتایج حاصل به یک نوع پایگاه داده‌ی خاص، حساس نمی‌باشند. برای بررسی و مقایسه‌ی دقیقتر الگوریتم‌ها، داده‌های دو پایگاه داده‌ی تحلیلی به نام‌های DW1 و DW2 به صورت تصادفی تولید شده‌اند. پایگاه داده‌ی تحلیلی DW1 دارای ده جدول بعد[۳۸] و یک جدول حقیقت[۳۹] است که دارای۱۰ میلیون رکورد می‌باشد. ابعاد این پایگاه داده‌ی تحلیلی برگرفته از پایگاه داده‌ی ثبت احوال می‌باشد که ۵ بعد آزمایشی، جهت افزایش تعداد ابعاد اضافه شده است. بنابراین ابعاد این پایگاه داده‌ی تحلیلی عبارتند از :نام، جنسیت، مکان (دارای سلسله مراتب کشور و شهر میباشد)، زمان (دارای سلسله مراتب سال و فصل میباشد)، تحصیلات، Dim1، Dim2، Dim3، Dim4، Dim5. همچنین، پایگاه داده‌ی تحلیلی DW2 دارای ۸ جدول بعد و یک جدول حقیقت است که دارای ۱۰ میلیون رکورد می‌باشد. ابعاد این پایگاه داده‌ی تحلیلی بر گرفته از پایگاه داده‌ی یک فروشگاه زنجیره-ای می‌باشد که ۴ بعد آزمایشی، جهت افزایش تعداد ابعاد اضافه شده است. بنابراین ابعاد این پایگاه داده‌ی تحلیلی عبارتند از :نام کالا، مکان (دارای سلسله مراتب کشور و شهر میباشد)، زمان (دارای سلسله مراتب سال و فصل میباشد)، شعبه، Dim1، Dim2، Dim3، Dim4. مقدار این فیلدها به صورت تصادفی و از طریق برنامه نویسی C# نسبت داده شده‌است تا الگوریتم پیشنهادی نسبت به پایگاه داده‌ی تحلیلی حساس نباشد..سیستم مورد استفاده برای پیاده‌سازی الگوریتم‌ها دارای 4GB RAM و CPU Corei3 2.2GHz می‌باشد.

برای مقایسه‌ی الگوریتم‌های مزبور، الگوریتم‌ها بر روی یک سیستم با مشخصات فوق اجرا شده‌است و آزمایشات متعددی برای مقایسه‌ی این دو الگوریتم صورت گرفته است. در این آزمایشات دو نوع پرس‌وجو وجود دارد: پرس‌وجوهای ورودی الگوریتم و پرس‌وجوهای تست. پرس‌وجوهای ورودی الگوریتم، پرس‌وجوهایی هستند که قبلا اتفاق افتاده‌اند. پرس‌وجوهای ورودی و پرس‌وجوهای تست، هر دو به صورت تصادفی تولید می‌شوند. پس از اجرای الگوریتم انتخاب دید، الگوریتم مورد نظر توسط پرس‌وجوهای تست مورد آزمایش قرار می‌گیرد. با توجه به این‌که زمان پاسخ به پرس‌وجوهای تحلیلی و میزان فضای ذخیره‌سازی دیدهای انتخاب شده از جمله مهم‌ترین معیارهای مورد بررسی می‌باشند [۲۳] و [۲۶] ، در این آزمایشات نیز این معیارها مورد بررسی قرار می گیرند. برای بررسی دقیق‌تر معیارهای فوق، سه معیار بیان شده در زیر، به عنوان معیارهای ارزیابی این مقاله انتخاب شده اند، معیارهای مورد بررسی در این آزمایشات به شرح زیر می‌باشند:

پارامترهای مربوط به جهش ترکیبی قورباغه، تعداد تکرارهای الگوریتم، تعداد ممپلکس‌ها، حداکثر تغییرات D پارامتر آلفا و حداکثر تغییرات جایگاه X_w می‌باشد. در ادامه نمودارهای مربوط به این پارامترها ترسیم می‌شود تا بهترین مقدار برای هر کدام از پارامترها به دست آید. هر کدام از این نمودارها، روی دو پایگاه‌داده‌تحلیلی DW1 و DW2 شبیه سازی شده‌اند. در تمامی نمودارها واحد زمان میلی ثانیه (ms) می‌باشد.

شکل شماره‌ی ۶ نمودار تغییرات زمان کل در برابر تغییرات تعداد تکرارهای الگوریتم جهش ترکیبی قورباغه را نشان می‌دهد. در این نمودار محور افقی، تعداد تکرارها می‌باشد و محور عمودی مقدار زمان کل است.



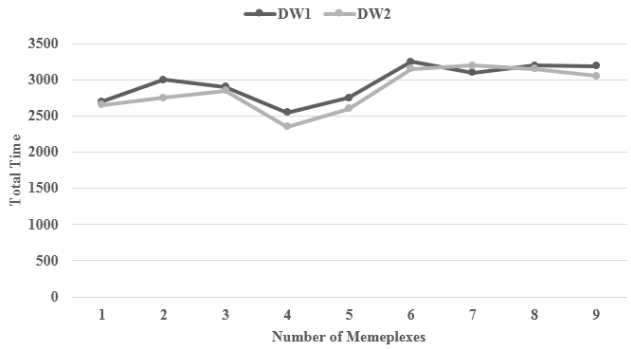
شکل ۶- زمان کل در برابر تغییرات تعداد تکرارهای الگوریتم جهش ترکیبی قورباغه

همانطور که در شکل شماره‌ی ۶ مشاهده می‌شود بهترین مقدار برای تعداد تکرارها ۷۰ می‌باشد.

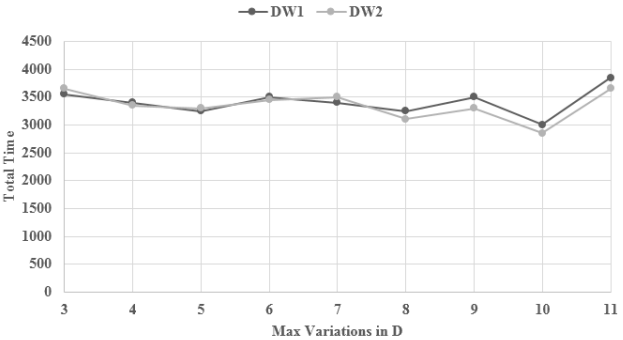
با توجه به اینکه شکل شماره‌ی ۶ خطی افقی نمی‌باشد بنابراین این نمودار نشان می‌دهد که الگوریتم جهش ترکیبی قورباغه در نقاط محلی نمی‌افتد. علت این امر استفاده از تابع جهش ژنتیکی می‌باشد.

شکل شماره‌ی ۷ نمودار تغییرات زمان کل در برابر تغییرات تعداد ممپلکس‌ها را نشان می‌دهد. در این نمودار محور افقی، تعداد ممپلکس‌ها می‌باشد و محور عمودی مقدار زمان کل است.

همانطور که در شکل شماره‌ی ۷ مشاهده می‌شود بهترین مقدار برای تعداد ممپلکس‌ها، ۴ می‌باشد. شکل شماره‌ی ۸ نمودار تغییرات زمان کل در برابر حداکثر تغییرات D را نشان می‌دهد. در این نمودار محور افقی، حداکثر تغییرات D می‌باشد و محور عمودی مقدار زمان کل است.

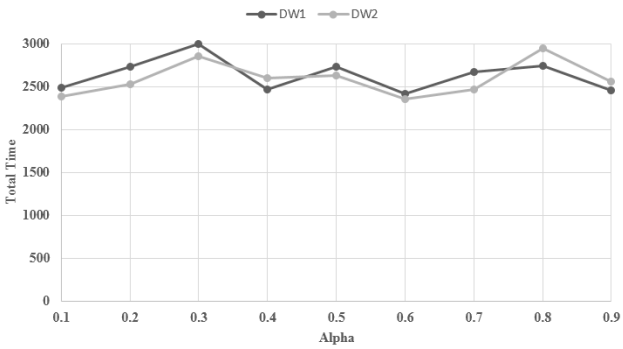


شکل ۷- زمان کل در برابر تغییرات تعداد ممپلکس‌ها



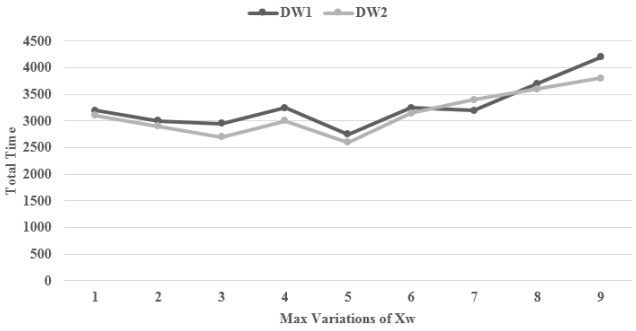
شکل ۸- زمان کل در برابر حداکثر تغییرات D

همانطور که در شکل شماره‌ی ۸ مشاهده می‌شود بهترین مقدار برای حداکثر تغییرات D، ۱۰ می‌باشد. شکل شماره‌ی ۹ نمودار تغییرات زمان کل در برابر تغییرات آلفا را نشان می‌دهد. در این نمودار محور افقی، مقدار آلفا می‌باشد و محور عمودی مقدار زمان کل است.



شکل ۹- زمان کل در برابر تغییرات پارامتر آلفا

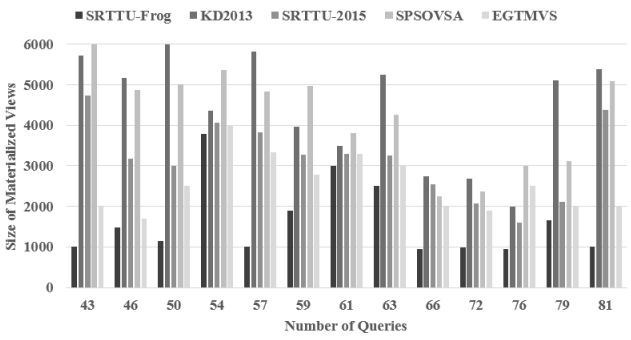
همانطور که در شکل شماره‌ی ۹ مشاهده می‌شود بهترین مقدار برای پارامتر آلفا، ۰.۶ می‌باشد. شکل شماره‌ی ۱۰ نمودار تغییرات زمان کل در برابر تغییرات جایگاه X_w را نشان می‌دهد. در این نمودار محور افقی، تغییرات جایگاه X_w می‌باشد و محور عمودی مقدار زمان کل است.



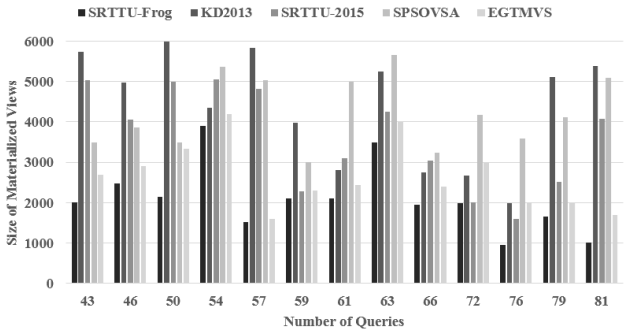
شکل ۱۰- زمان کل در برابر تغییرات جایگاه X_w

همانطور که در شکل شماره‌ی ۱۰ مشاهده می‌شود بهترین مقدار برای تغییرات جایگاه X_w ، ۵ می‌باشد. بنابراین پارامترهای مربوط به الگوریتم جهش ترکیبی قورباغه، تعداد تکرارهای الگوریتم برابر ۷۰، تعداد ممپلکس‌ها برابر ۴، حداکثر تغییرات D برابر ۱۰، پارامتر آلفا برابر ۰.۶ و حداکثر تغییرات جایگاه X_w برابر ۵ می‌باشد.

در آزمایشات انجام شده، برای به دست آوردن مقدار معیار مربوطه، آزمایشات به طور جداگانه روی DW1 و DW2 پنج بار تکرار شده‌اند و میانگین عددی پنج عدد به دست آمده، محاسبه شده و در نمودارهای این بخش ترسیم شده‌اند. شکل‌های شماره‌ی ۱۱ و ۱۲ نشان دهنده‌ی مقایسه‌ی الگوریتم‌های SRTTU-Frog، KD2013، SRTTU-2015، SPSOVSA و EGTMVS بر اساس معیار شماره‌ی (۱) روی پایگاه داده DW1 و DW2 می‌باشند. در این نمودار محور افقی نشان‌دهنده‌ی تعداد پرس‌وجوها می‌باشد و محور عمودی نشان‌دهنده‌ی تعداد رکوردهای دیدهای ذخیره‌شده‌ی نهایی می‌باشد.



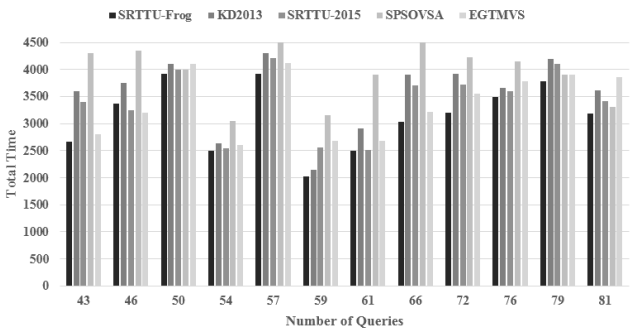
شکل۱۱- مقایسه‌ی تعداد سطرهای دیدهای ذخیره شده با افزایش تعداد پرس‌وجوها در پایگاه داده‌ی تحلیلی DW1



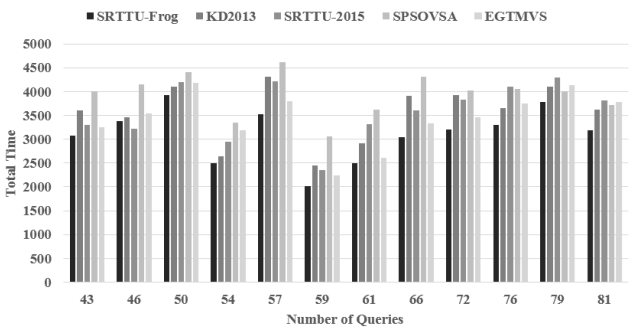
شکل۱۲- مقایسه‌ی تعداد سطرهای دیدهای ذخیره شده با افزایش تعداد پرس‌وجوها در پایگاه داده‌ی تحلیلی DW2

با توجه به نمودارهای رسم شده در شکل‌های۱۱ و ۱۲ مشخص می‌شود که تعداد سطرهای دیدهای ذخیره‌شده توسط الگوریتم پیشنهادی، SRTTU-Frog، همیشه کم‌تر از تعداد سطرهای به‌دست‌آمده توسط سایر الگوریتم‌ها می‌باشد. این بدین معناست که الگوریتم پیشنهادی نیاز به فضای ذخیره‌سازی کم‌تری دارد این مقایسه نشان می‌دهد که الگوریتم پیشنهادی SRTTU-Frog از لحاظ فضای ذخیره‌سازی بهتر است.

شکل‌های شماره‌ی ۱۳ و ۱۴ نشان دهنده‌ی مقایسه‌ی الگوریتم‌های SRTTU-Frog، KD2013، SRTTU-2015، SPSOVSA و EGTMVS بر اساس معیار شماره‌ی (۲) روی پایگاه داده DW1 و DW2 می‌باشند. محور افقی نشان‌دهنده‌ی تعداد پرس‌وجوها می‌باشد و محور عمودی نشان‌دهنده‌ی زمان کل می‌باشد. این زمان شامل زمان اجرای الگوریتم و زمان صرف‌شده برای پاسخ به پرس‌وجوهای تست می‌باشد.



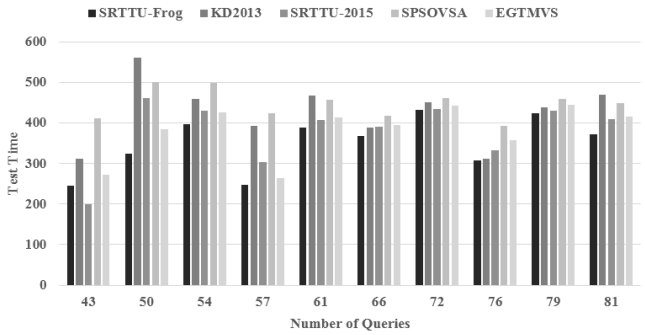
شکل ۱۳- مقایسه‌ی زمان کل با افزایش تعداد پرس‌وجوها در پایگاه داده‌ی تحلیلی DW1



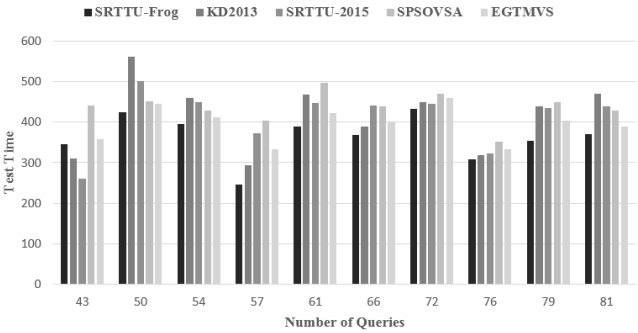
شکل۱۴- مقایسه‌ی زمان کل با افزایش تعداد پرس‌وجوها در پایگاه داده‌ی تحلیلی DW2

نمودارهای رسم شده در شکل‌های ۱۳و ۱۴ نشان می‌دهد که زمان کل الگوریتم پیشنهادی SRTTU-Frog همیشه کم‌تر از زمان کل سایر الگوریتم‌ها می‌باشد. با توجه به اهمیت زمان، هر الگوریتم انتخاب دیدی که در زمان کم‌تری به پرس‌وجوهای تست پاسخ دهد، الگوریتم انتخاب دید مناسب‌تری خواهدبود. با توجه به نمودار رسم شده در شکل‌های۱۳ و ۱۴، مشاهده می‌شود که الگوریتم پیشنهادی، SRTTU-Frog، دارای زمان پاسخ کم‌تری بوده‌است.

شکل‌های شماره‌ی ۱۵ و ۱۶ نشان دهنده‌ی مقایسه‌ی الگوریتم‌های SRTTU-Frog، KD2013، SRTTU-2015، SPSOVSA و EGTMVS بر اساس معیار شماره‌ی (۳) روی پایگاه داده DW1 و DW2 می‌باشند. در این نمودارها محور افقی نشان‌دهنده‌ی تعداد پرس‌وجوها می‌باشد و محور عمودی نشان‌دهنده‌ی زمان صرف‌شده برای پاسخ به پرس‌وجوهای تست می‌باشد.



شکل۱۵- مقایسه‌ی زمان تست با افزایش تعداد پرس‌وجوها در پایگاه داده‌ی تحلیلی DW1



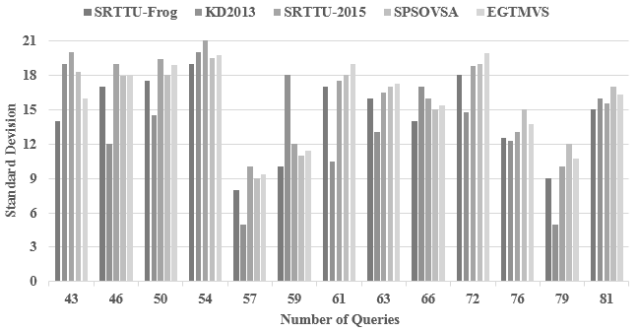
شکل۱۶- مقایسه‌ی زمان تست با افزایش تعداد پرس‌وجوها در پایگاه داده‌ی تحلیلی DW2

با توجه به نمودار رسم‌شده مشاهده‌می‌شود که زمان لازم برای پاسخ‌دادن به پرس‌وجوهای تست، در الگوریتم پیشنهادی، SRTTU-Frog، نسبت به سایر الگوریتم‌ها، کم‌تر می‌باشد. الگوریتمی که در زمان کم‌تری به پرس‌وجوهای تست پاسخ دهد الگوریتم مناسب‌تری است.

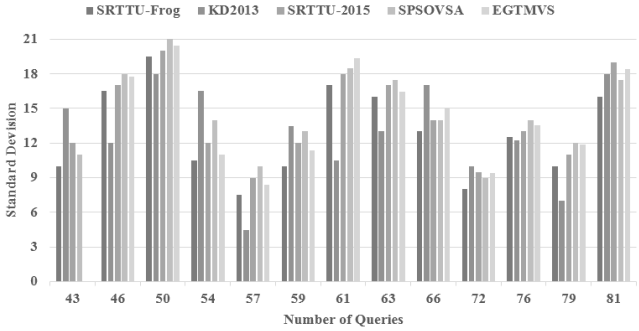
شکل‌های ۱۱الی۱۶ میانگین پنج بار تکرار هر آزمایش را نشان می‌دهد.

نمودارهایی که در ادامه ترسیم شده اند، انحراف معیار داده ها را نشان می دهند. شکل ۱۷ نشان‌دهنده‌ی انحراف معیار برای تعداد سطرهای دیدهای ذخیره‌شده در برابر تعداد پرس‌وجوها می‌باشد. در این نمودار محور افقی نشان‌دهنده‌ی تعداد پرس‌وجوها می‌باشد و محور عمودی نشان‌دهنده‌ی انحراف معیار می‌باشد.

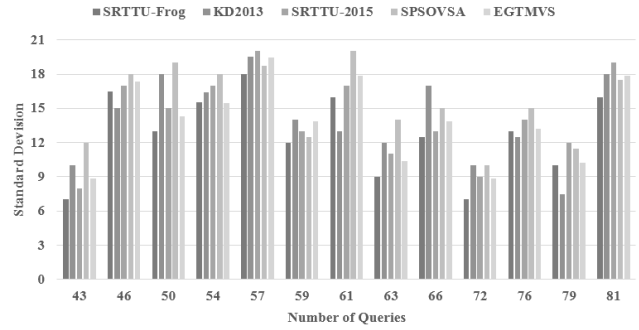
شکل های ۱۸ و ۱۹ به ترتیب نشان دهنده‌ی انحراف معیار زمان کل و انحراف معیار زمان تست در برابر افزایش تعداد پرس‌وجوها می‌باشند.



شکل۱۷- مقایسه‌ی انحراف معیار تعداد سطرهای دیدهای ذخیره‌شده با افزایش تعداد پرس‌وجوها



شکل۱۸- مقایسه‌ی انحراف معیار زمان کل با افزایش تعداد پرس‌وجوها



شکل۱۹- مقایسه‌ی انحراف معیار زمان تست با افزایش تعداد پرس‌وجوها

انحراف معیار یکی از شاخص‌های پراکندگی است که نشان می‌دهد به طور میانگین داده‌ها چه مقدار، از مقدار متوسط فاصله دارند. اگر انحراف معیار مجموعه‌ای از داده‌ها نزدیک به صفر باشد، نشانه آن است که داده‌ها نزدیک به میانگین هستند و پراکندگی اندکی دارند؛ در حالی که انحراف معیار بزرگ بیانگر پراکندگی قابل توجه داده‌ها می‌باشد.

همانطور که در شکل های ۱۷، ۱۸ و ۱۹ نشان داده شده است، انحراف معیار الگوریتم پیشنهادی نسبت به سایر الگوریتم‌ها کمتر می باشد. بنابراین با توجه به تعریف انحراف معیار پراکندگی جواب های الگوریتم پیشنهادی نسبت به سایر الگوریتم‌ها کمتر می باشد.

با توجه به آزمایشات انجام شده مشاهده می‌شود که الگوریتم پیشنهادی، SRTTU-Frog، نسبت به الگوریتم‌هایKD2013، SRTTU-2015، SPSOVSA و EGTMVS دارای کارایی بیش‌تری می‌باشد.

جدول ۱ نشان دهنده‌ی درصد بهبود الگوریتم پیشنهادی نسبت به سایر الگوریتم‌ها می‌باشد. این جدول بیان می‌کند که الگوریتم SRTTU-Frog نسبت به سایر الگوریتم‌ها از نظر معیارهای زمانی (زمان کل و زمان تست)، و از نظر تعداد سطرهای دیدهای ذخیره شده بهبود داشته است.

جدول۱- درصد بهبود الگوریتم پیشنهادی

	تعداد سطرهای دیدهای ذخیره شده		زمان کل		زمان تست	
	DW1	DW2	DW1	DW2	DW1	DW2
KD2013	59.9%	48.1%	14.1%	14.5%	16.6%	15.0%
SRTTU-2015	46.3%	37.8%	9.8%	16.0%	12.2%	11.5%
SPSOVSA	59.5%	49.7%	28.9%	28.2%	19.0%	18.8%
EGTMVS	36.5%	22%	8.5%	10.4%	9.3%	4.8%

در پایان این بخش با استفاده از روش‌های آماری الگوریتم پیشنهادی با الگوریتم KD2013 مقایسه می‌شود. روش آماری مورد استفاده در این مقاله، آزمون t (student's t test) می‌باشد. آزمون t روشی آماری است که میزان تفاوت میانگین‌های دو گروه را با هم مقایسه می‌کند. در صورتیکه این مقدار کمتر از ۰.۰۵ (۵درصد) باشد، قابل قبول می‌باشد. هر چه این عدد کم‌تر باشد، بهتر است؛ زیرا به این معنی است که فاصله داده‌ها از میانگین کم‌تر می‌باشد و داده‌ها یکدست هستند. جدول شماره ۲ نشان‌دهنده‌ی مقدار *p-value* در نتایج آزمون t می‌باشد.

جدول۲- مقدار *p-value* در آزمون t

داده ها	KD2013	SRTTU-Frog
تعداد سطرهای دیدهای ذخیره شده	۰.۰۴۵	۰.۰۳
زمان تست	۰.۰۳۶	۰.۰۲۴
زمان کل	۰.۰۳۴	۰.۰۱۹

با توجه به جدول فوق مشاهده می‌شود که مقدار *p-value* برای هر دو الگوریتم کمتر از ۰.۰۵ می‌باشد و مقدار *p-value* برای الگوریتم پیشنهادی نسبت به الگوریتم KD2013 کمتر می‌باشد بنابراین الگوریتم پیشنهادی دارای عملکرد بهتری می‌باشد.

جدول ۳ نمودار همگرایی الگوریتم جهش ترکیبی قورباغه را نشان می‌دهد. در این آزمایش، یکی از خوشه ها که دارای ۸ پرس‌وجو می‌باشد مورد بررسی قرار گرفته است.

همانطور که در جدول شماره ۳ مشاهده می‌شود با افزایش تعداد تکرارها، مقدار X_g همگرا می‌شود.

جدول۳- مقدار X_g به ازای تعداد تکرار های مختلف

Iteration Number	X_g
10	11010110
20	11010101
30	11010110
40	11010100
50	11010100
60	11010100
70	11010100

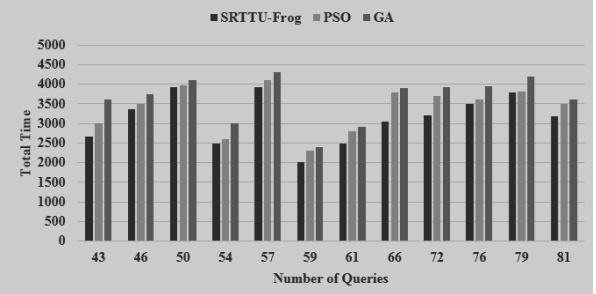
۵- مقایسه الگوریتم پیشنهادی با الگوریتم ژنتیک و الگوریتم ازدحام ذرات[۴۰]

علاوه بر مقایسات انجام شده، الگوریتم پیشنهادی با الگوریتم های ژنتیک و الگوریتم ازدحام ذرات نیز مقایسه می گردد. در ابتدا هر دو الگوریتم توضیح داده می شود.

الگوریتم ژنتیک تکنیک جستجو در علم رایانه برای یافتن راه‌حل تقریبی برای بهینه‌سازی و مسائل جستجو است. الگوریتم ژنتیک نوع خاصی از الگوریتمهای تکاملی می‌باشد. این الگوریتم برای اولین بار توسط جان هلند معرفی شد. الگوریتم‌های ژنتیک [۳۱] از اصول انتخاب طبیعی داروین برای یافتن فرمول بهینه جهت پیش‌بینی یا تطبیق الگو استفاده می‌کنند. الگوریتم‌های ژنتیک اغلب گزینه خوبی برای تکنیک‌های پیش‌بینی بر مبنای رگرسیون هستند. مختصراً گفته می‌شود که الگوریتم ژنتیک یک تکنیک برنامه‌نویسی است که از تکامل ژنتیکی به عنوان یک الگوی حل مسئله استفاده می‌کند. مسئله‌ای که باید حل شود ورودی است و راه‌حل‌ها طبق یک الگو کدگذاری می‌شوند که تابع برازش نام دارد. هر راه حل کاندید را ارزیابی می‌کند که اکثر کاندیدها به صورت تصادفی انتخاب می‌شوند.

الگوریتم ازدحام ذرات [۳۳] از حرکت دسته جمعی پرندگانی که به دنبال غذا می‌باشند الهام گرفته است. گروهی از پرندگان در فضایی به صورت تصادفی به دنبال غذا می‌گردند. تنها یک تکه غذا در فضای مورد بحث وجود دارد. هیچ یک از پرندگان محل غذا را نمی‌دانند. یکی از بهترین استراتژی‌ها می تواند دنبال‌کردن پرنده‌ای باشد که کمترین فاصله را تا غذا داشته باشد. الگوریتم ازدحام ذرات از این استراتژی استفاده می کند. در این الگوریتم به هر ذره، راه حل گفته می شود. هر ذره یک مقدار شایستگی دارد که توسط یک تابع شایستگی محاسبه می‌شود. هر چه ذره در فضای جستجو به هدف -غذا در مدل حرکت پرندگان- نزدیک‌تر باشد، شایستگی بیشتری دارد. همچنین هر ذره دارای یک سرعت است که هدایت حرکت ذره را بر عهده دارد. هر ذره با دنبال کردن ذرات بهینه در حالت فعلی، به حرکت خود درفضای مساله ادامه می‌دهد. بنابراین در ابتدا، گروهی از ذرات به صورت تصادفی به وجود می آیند و با به‌روز کردن نسل‌ها سعی در یافتن راه حل بهینه می‌نمایند. در هر گام، هر ذره با استفاده از دو بهترین مقدار به روز می‌شود. اولین مورد، بهترین موقعیتی است که تا کنون ذره موفق به رسیدن به آن شده است. موقعیت مذکور با نام pbest شناخته و نگهداری می‌شود. بهترین مقدار دیگری که توسط الگوریتم مورد استفاده قرار می‌گیرد، بهترین موقعیتی است که تا کنون توسط جمعیت ذرات بدست آمده است.

این موقعیت با gbest نمایش داده می شود. حال، الگوریتم پیشنهادی مقاله و الگوریتم‌های ژنتیک و الگوریتم ازدحام ذرات با توجه به معیار زمان کل مقایسه می‌شوند. نمودار شکل ۲۰ نشان دهنده‌ی مقایسه‌ی الگوریتم‌ها از لحاظ معیار زمان پاسخ می‌باشد.



شکل ۲۰- مقایسه‌ی زمان کل با افزایش تعداد پرس‌وجوها

همانطور که در شکل ۲۰ نشان داده شده است، زمان پاسخ الگوریتم پیشنهادی نسبت به الگوریتم ژنتیک و الگوریتم ازدحام ذرات بهتر می باشد.

۶- نتیجه گیری

در این مقاله الگوریتمی برای انتخاب دید ارائه شده‌است. این الگوریتم SRTTU-Frog نامیده شده‌است. الگوریتم SRTTU-Frog بر اساس پرس‌وجوهای قبلی که بر روی پایگاه داده‌ی تحلیلی اجرا شده‌اند کار می‌کند. ابتدا پرس‌وجوهای قبلی با توجه به الگوریتم‌های خوشه‌بندی، خوشه‌بندی می‌شوند. سپس در هر خوشه، پرس‌وجوهای پرتکرار به دست می‌آیند. پس از آن با توجه به میزان فضای ذخیره‌سازی موجود، پرس‌وجوهای بهینه در هر خوشه به دست می‌آیند. در این مرحله، پرس‌وجوهای بهینه با استفاده از الگوریتم کوله‌پشتی صفرویک که با استفاده از الگوریتم جهش ترکیبی قورباغه حل شده‌است، به دست می‌آید. در نهایت در هر خوشه، پرس‌وجوهای مفید پیوند می‌شوند تا به ازای هر خوشه یک دید ذخیره‌سازی شود. الگوریتم پیشنهادی SRTTU-Frog بهبود یافته‌ی الگوریتم KD2013 می‌باشد. در الگوریتم SRTTU-Frog روش یافتن پرس‌وجوهای مفید با استفاده از کوله‌پشتی صفرویک بهبود یافته است. با توجه به نمودارهای رسم شده در بخش قبل مشاهده می‌شود که الگوریتم پیشنهادی SRTTU-Frog نسبت به الگوریتم‌هایKD2013، SRTTU-2015 ، SPSOVSA و EGTMVS دارای کارایی بالاتری می‌باشد. علت این بهبود استفاده از الگوریتم جهش ترکیبی قورباغه برای یافتن پرس‌وجوهای مفید می‌باشد.

مراجع

[1] T. v. Kumar and S. Kumar, "Materialized View Selection Using Iterative Improvement," Advances in Computing & Inf. Technology, vol. 3, pp. 205-213, 2013.

[2] J. Han and M. Kamber, Data mining Concepts and Techniques, vol. third edition, NewYork, 2012.

[3] T. V. V. Kumar, G. Dubey and A. singh, "Frequent Queries Selection for View Materialization," Advances in Computing and Information Technology, vol. 177, pp. 521-530, 2013.

[4] J. Yang, K. Karlapalem and Q. Li, "Algorithms for materialized view design in data warehousing environment," VLDB, vol. 97, 1997.

[5] P. Kalnis, N. Mamoulis and D. Papadias, "View selection using randomized search," Data and Knowledge Engineering, vol. 42, no. 1, pp. 89-111, 2002.

[6] J. Horng, Y. Chang, B. Liu and C. Kao, "Materialized view selection using genetic algorithms in a data warehouse system," Evolutionary Computation, vol. 3, 1999.

[7] J. Horng, Y. Chang and B. Liu, "Applying evolutionary algorithms to materialized view selection in a data warehouse," Soft Computing, vol. 7, no. 8, pp. 574-581, 2003.

[8] C. Zhang, X. Yao and J. Yang, "Evolving materialized views in a data warehouse," Evolutionary Computation, vol. 2, pp. 823-829, 1999.

[9] C. Zhang, X. Yao and J. Yang, "An evolutionary approach to materialized views selection in a data warehouse environment," IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 31, no. 3, pp. 282-294, 2001.

[10] S. Rizzi and E. Saltarelli, "View materialization vs, indexing: balancing space constraints in data warehouse design," in Advanced Information Systems Engineering, Klagenfurt, Austria, 2003.

[11] D. Theodoratos and W. Xu, "Constructing search spaces for materialized view selection," in 7th ACM International Workshop on Data Warehousing and OLAP, Washington, USA, 2004.

[12] I. Mami and Z. Bellahsene, "A survey of view selection methods," ACM SIGMOD, vol. 41, no. 1, pp. 20-29, 2012.

[13] C. A. Dhote and M. S. Ali, "Materialized view selection in data warehousing: a survey," Journal of Applied sciences, vol. 9, no. 3, pp. 401-414, 2009.

[14] J.-S. Sohn, J.-H. Yang and I.-J. Chung, "Improved view selection algorithm in data warehouse," IT Convergence and Security, pp. 921-928, 2013.

[15] W. Xu, D. Theodoratos, C. Zuzarte, X. Wu and V. Oria, "A dynamic view materialization scheme for sequences of query and update statements," Data Warehousing and Knowledge Discovery, pp. 55-56, 2007.

[16] N. Daneshpour and A. Abdollahzadeh Barfourosh, "Dynamic view Management System for Query Prediction to view materialization," International Journal of Data Warehousing and Mining, vol. 7, no. 2, pp. 67-96, 2011.

[17] I. Mami, R. Coletta and Z. Bellahsene, "Modeling view selection as a constraint satisfaction problem," in International Conference on Database and Expert Systems Applications, France, 2011.

[18] I. Mami, Z. Bellahsene and R. Coletta, "A Declarative Approach to View Selection Modeling," Transactions on Large-Scale Data-and Knowledge-Centered Systems, pp. 115-145, 2013.

[19] I. Mami, Z. Bellahsene and R. Coletta, "View selection under multiple resource constraints in a distributed context," in International Conference on Database and Expert Systems Applications, Vienne, 2012.

[20] R. Huang, R. Chirkova and Y. Fathi, "Advances in Databases and Information Systems," in Deterministic view selection for data

analysis queries: Properties and algorithms, Berlin, Springer Berlin Heidelberg, 2012, pp. 195-208.

[21] Z. Asgharzadeh, R. Chirkova and Y. Fathi, "Exact and inexact methods for selecting views and indexes for olap performance improvement," in international conference on Extending database technology: Advances in database technology, France, 2008.

[22] T. V. Kumar and M. Haider, "Query answering-based view selection," International Journal of Business Information Systems, vol. 18, no. 3, pp. 338-353, 2015.

[23] T. V. Kumar and K. Devi, "Materialised view construction in data warehouse for decision making," International Journal of Business Information Systems, vol. 11, no. 4, pp. 379-396, 2012.

[24] T. V. V. Kumar, A. Singh and G. Dubey, "Mining Queries for Constructing Materialized Views in a Data Warehouse," Advances in Computer Science, Engineering & Applications, pp. 149-159, 2012.

[25] T. V. Kumar and K. Devi, "Frequent Queries Identification for Constructing Materialized Views," in Electronics Computer Technology (ICECT), Kanyakumari, 2011.

[26] T. V. V. Kumar, A. Goel and N. Jain, "Mining information for constructing materialised views," Int. J. Information and Communication Technology, vol. 2, no. 4, pp. 386-405, 2010.

[27] K. K. Bhattacharjee and S. P. Sarmah, "Shuffled frog leaping algorithm and its application to 0/1 knapsack problem," Applied Soft Computing, vol. 19, pp. 252-263, 2014.

[28] T. V. V. Kumar and B. Arun, "Materialized View Selection Using HBMO," International Journal of System Assurance Engineering and Management, pp. 1-14, 2015.

[29] D. Yao, A. Abulizi and R. Hou, "An improved algorithm of materialized view selection within the confinement of space," IEEE Fifth International Conference on Big Data and Cloud Computing., 2015.

[30] P.R. Vishwanath, R. Rajyalakshmi and S. Reddy "An Association Rule Mining for Materialized View Selection and View Maintenance," International Journal of Computer Applications, vol. 109 pp. 15-20, 2015.

[31] S.H. Talebian and S.A. Kareem "A Lexicographic Ordering Genetic Algorithm for Solving Multi-objective View Selection Problem," IEEE Second International Conference on Computer Research and Development, pp. 110-115, 2010.

[۳۲] ریحانه صباغ گل، نگین دانشپور، "بهبود الگوریتم دید در پایگاه داده تحلیلی با استفاده از یافتن پرس و جوهای پرتکرار"، ژورنال پردازش علائم و داده ها، شماره ۱، پیاپی ۳۱، ۱۳۹۶.

[33] A. Kumar, T.V. Kumar," Materialized View Selection Using Set Based Particle Swarm Optimization," International Journal of Cognitive Informatics and Natural Intelligence, pp. 18-39, 2018.

[34] M. Nikolic, "Distributed Incremental View Maintenance,". In: Sakr S., Zomaya A.Y. (eds) Encyclopedia of Big Data Technologies. Springer, Cham, 2019.

[35] M. K. Sohrabi and H. Azgomi, "Evolutionary game theory approach to materialized view selection in data warehouses," Knowledge-Based Systems, vol. 163, pp. 558-571, 2019.

[36] A.Gosain and K.Sachdeva, "Selection of materialized views using stochastic ranking based Backtracking Search Optimization Algorithm," International Journal of System Assurance Engineering and Management, vol. 10, no. 4, pp. 801-810, 2019.

[37] H. Azgomi and M.K., Sohrabi, "A novel coral reefs optimization algorithm for materialized view selection in data warehouse environments," Applied Intelligence, vol. 49, no. 11, pp. 3965-3989, 2019.

[38] H. Azgomi and M.K. Sohrabi, "A game theory based framework for materialized view selection in data warehouses," Engineering Applications of Artificial Intelligence, vol. 71, pp. 125-137, 2018.

[39] J.Prakash and T.V. Kumar, "Multi-objective materialized view selection using MOGA," International Journal of System Assurance Engineering and Management, pp. 1-12, 2020.

[40] H. Ehsan and M.A. Sharaf, "Materialized View Selection for Aggregate View Recommendation," In Australasian Database Conference (pp. 104-118). Springer, Cham, 2019.

پاورقی‌ها:

[1] on-demand
[2] in-advance
[3] Data warehouse
[4] Subject-oriented
[5] Integrated
[6] Time-variant
[7] Nonvolatile
[8] Materialized View
[9] join
[10] Workload
[11] Multi-View Processing Plan
[12] Aggregate
[13] DAG
[14] Constraint Satisfaction Problem
[15] integer programming
[16] Candidate
[17] Decision making
[18] Lattice of cuboids
[19] Evolutionary Game Theory-based Materialized View Selection
[20] Stochastic Ranking (SR)
[21] Backtracking Search Optimization (BSA)
[22] Dynamic
[23] Materialized Views Construction Framework
[24] Clusters
[25] domain, area, subject area
[26] Jaccard's coefficient
[27] Query Merger
[28] visit
[29] Minimum threshold
[30] population
[31] solutions
[32] Memeplex
[33] culture
[34] Memeplex evolution
[35] fitness
[36] Repair Methods
[37] Genetic Mutation
[38] Dimension
[39] Fact
[40] Particle Swarm Optimization (PSO)