MAKER: الگوریتمی جدید در زمینه قوانین انجمنی

دکترمسعود یقینی[1]؛ مهسا مرتضوی[3]؛ کاوه رسولی چیذری[3]؛
مهیار حسین زاده[2]؛ عرفان خاجی[2]

چکیده

این مقاله، نتیجه تلاش برای طراحی، ایجاد و پیاده سازی الگوریتمی جدید در زمینه کاوش قوانین انجمنی[1] است. الگوریتمی که با نهایت دقت بتواند تمامی قوانین انجمنی حاکم بر یک پایگاه داده[2] را شناسایی کند. الگوریتم MAKER بر مبنای کدگذاری پایگاه داده بر مبنای اعداد اول عمل می کند. بدین صورت که به هر نوع کالا یک عدد اول تخصیص داده می شود، سپس هر سبد خرید توسط یک عدد که حاصل ضرب اعداد اول تخصیص داده شده به کالاهای درون آن سبد خرید است، مشخص می شود. به این عدد، عدد سبد خرید گفته می شود. در نتیجه با تقسیم عدد هر سبد خرید به عدد اول تخصیص داده شده به یک کالا ، وجود آن کالا درآن سبد خرید تایید و یا رد می شود. الگوریتم MAKER پس از یافتن تمامی کالاهای تکرار شونده تک عضوی، به سراغ یافتن مجموعه کالاهای تکرار شونده چند عضوی می رود، بدین صورت که هر عضو یک مجموعه چند عضوی را به طور جداگانه و به شیوه ذکر شده در یک سبد خرید جستجو کرده، در صورت وجود همه اعضا، وجود آن مجموعه در یک سبد خرید را تایید می کند. این روش، یک روش چند مرحله ای است، و از این لحاظ، از روش Apriori پیروی می کند.

واژه‌های کلیدی

داده کاوی، قوانین انجمنی، الگوریتم Apriori و الگوریتم MAKER.

MAKER: A New Algorithm in Finding Association Rules

Dr. Masoud Yaghini[1], K. R. Chizari[2], M. Mortazavi[2],
E. Khaji[1], M. Hoseynzadeh[1]

ABSTRACT

This article is the result of design, create and implementation of a new algorithm in the field of Mining Frequent Itemset which is able to find all frequent itemset in a database precisely. MAKER is based on database encoding with primes numbers. We assign a prime number to each product in the database. Consequently, each transaction in the database can be recognized just with one unique number which is the result of multiplying the prime numbers assigned to the products of that transaction. We named the unique number, transaction number. Therefore, dividing a transaction number into a prime number assigned to an especial product, we can easily recognize whether the product is involved in that

[1] Association rules mining
[2] Data Base

۱- دکتر مسعود یقینی، استاد یار دانشکده مهندسی راه آهن، دانشگاه علم و صنعت ایران، نارمک، تهران، ایران، آدرس پست الکترونیک: Yaghini@iust.ac.ir.
۲- مهیار حسین زاده، عرفان خاجی، دانشجویان کارشناسی دانشکده مهندسی راه آهن، دانشگاه علم و صنعت ایران، نارمک، تهران، ایران، آدرس پست الکترونیک: Akhadji@yahoo.com, Mahyar_hoseynzadeh@yahoo.com.
۳ – مهسا مرتضوی، کاوه رسولی چیذری، دانشجویان کارشناسی دانشکده الکترونیک، دانشگاه علم و صنعت ایران، نارمک، تهران، آدرس پست الکترونیک: Mahsa_mortazavi@yahoo.com , Kaveh_top۲۰۰۳@yahoo.com.

transaction or not. Finding all of the frequent items, we directly find k-frequent itemset by searching each item of the itemset in each transaction individually, using so-called method. MAKER algorithm is a level-wised algorithm which is the Apriori's characteristic.

KEYWORDS
Data mining, Association rules, Apriori algorithm and MAKER algorithm.

۱. مقدمه

یافتن مجموعه های تکرار شونده یکی از مهمترین بخش های دانش داده کاوی و اولین قدم در راه تولید قوانین انجمنی می باشد. یافتن این قوانین به ما در جهت شناخت هر چه بیشتر رفتار مشتری، تطبیق هر چه بهتر خدمات با رفتار وی و در نهایت کسب درآمد بیشتر، کمک شایان توجهی می کند. در نتیجه طی سالهای اخیر تحقیقات فراوانی در مورد الگوریتم های کاوش قوانین انجمنی انجام گرفته است. آزمایش های صورت گرفته نشان می دهد که برخی از الگوریتم های کاوش قوانین انجمنی برای بعضی از پایگاه های داده با دقت قابل قبولی عمل نمی کنند و یا در بعضی از موارد قابل اجرا شدن نمی باشند [۱۹]. ما در این مقاله به معرفی روشی جدید می پردازیم که با سرعت قابل قبولی بتواند تمامی قوانین انجمنی حاکم بر انواع گوناگون پایگاه های داده را به طور کاملا دقیق کشف کند. در ادامه مقاله، ابتدا به تعریف تکرار شوندگی و قوانین انجمنی می پردازیم و پس از مروری بر پیشینه تحقیق، به ارائه الگوریتم MAKER و نتایج حاصل از آزمایش این الگوریتم می پردازیم.

۲. تعریف مسئله

یک قانون انجمنی، به بیان احتمال وجود یک رابطه ی خاص بین مجموعه کالای x و مجموعه کالای y می پردازد. این رابطه را به صورت $x \rightarrow y$ نشان می دهند; بدین معنی که اگر مجموعه کالای y با مجموعه کالای x در رابطه باشد، آنگاه مجموعه کالای y به احتمال معینی به همراه مجموعه کالای x خریداری می شود. در سال ۱۹۹۳، Agrawal و همکاران [۱]، تعریف بالا را به صورت ریاضی بیان کردند:

فرض کنیم ،

$$I = I_1, I_2, \ldots, I_m,$$

مجموعه ای از کالاها باشند که یکی از اعداد ۰ و یا ۱ را می توان به آن ها نسبت داد. برای مثال اگر I_m برابر ۱ باشد، کالای I_m در یک مجموعه مشخص وجود دارد و در صورت اختیار کردن عدد ۰ نمی تواند در آن مجموعه وجود داشته باشد. فرض کنیم که D مجموعه ای از تراکنش ها باشد، طوری که که هر تراکنش مثل T، شامل مجموعه ای از کالا هاست به نحوی که:

$$T \subset I.$$

اگر x مجموعه ای از کالاها باشد طوری باشد که :

$$x \subset I,\ x \subset T,$$

آنگاه، تراکنش T شامل مجموعه کالای x است. یک قانون انجمنی به شکل عبارت $x \rightarrow y$ بیان می شود، طوری که :

$$x \subset I,\ y \subset I,\ x \cap y = \emptyset.$$

یعنی y یک مجموعه کالای مشخص در مجموعه کالاهای I است که در مجموعه ی کالاهای x عضویت ندارد.

اگر حداقل $C\ \%$ $(۱ \leq C \leq ۰)$ معاملاتی که دارای مجموعه کالای x می باشند (یعنی معاملاتی که در آن ها مجموعه کالاهای x خریداری شده باشد)، دارای مجموعه کالای y نیز باشند، (یعنی مجموعه کالای y نیز در آن ها خریداری شده باشد)، آنگاه رابطه ی $x \rightarrow y$ دارای فاکتور اطمینان C [۳] می باشد. یعنی در $C\ \%$ مواردی که مجموعه کالا ی x خریداری شده است، مجموعه کالای y نیز خریداری می شود. اگر به تعداد S، معامله از مجموعه معاملات T پیدا شود طوری که این تعداد معامله شامل مجموعه ی کالای x و مجموعه کالای y باشند، آنگاه رابطه ی $x \rightarrow y$ دارای معیار تکرارشوندگی S [٤] خواهد بود. مسئله اصلی در زمینه ی کاویدن قوانین انجمنی، ایجاد تمامی قوانین انجمنی است که دارای معیار تکرارشوندگی و فاکتور اطمینانی بیشتر از حداقل های تعیین شده باشند. به این حداقل های از قبل تعیین شده، آستانه تکرارشوندگی [٥] و حداقل اطمینان[٦] می گویند که این مقادیر بستگی به نظر صاحب پایگاه داده (صاحب شغل) دارد.

بنابراین مسئله کاوش قوانین انجمنی دارای دو مرحله اصلی می باشد :

۱- پیدا کردن مجموعه ای از کالاهای تکرار شونده که میزان تکرار آن ها در پایگاه داده، از آستانه تکرارشوندگی بیشتر باشد.

۲- پیدا کردن تمامی قوانین انجمنی به طوری که اگر f یک مجموعه کالای تکرار شونده باشد و a یک زیرمجموعه از مجموعه ی کالای f باشد، آنگاه:

$$a \longrightarrow (f - a).$$

شرط برقراری این رابطه این است که احتمال وجود مجموعه کالای $f - a$ در یک سبد خرید به شرط وجود کالای a در آن سبد خرید بیش از حداقل اطمینان باشد. با توجه به تعریف فاکتور اطمینان، احتمال وجود مجموعه کالای $f - a$ را در یک سبد خرید، به شرط وجود کالای a در آن سبد خرید به وسیله رابطه زیر حساب می شود:

$$p\ (\alpha \longrightarrow\ f - a) = p\ (\alpha\ (\ f - a)) = \frac{p\ (\alpha(\ f - a))}{p(\alpha)}\ .$$

قدم اول در راه کاوش قوانین انجمنی یعنی یافتن مجموعه کالاهای تکرار شونده، امری بسیار زمان بر، مشکل و پر هزینه می باشد و در نتیجه یکی از موارد و موضوعات مهم در زمینه های تحقیقاتی و پژوهشی در سال های اخیر به حساب می آید.

---

[۳] Confidence Factor
[٤] Support Factor
[٥] Minimum Support
[٦] Minimum Confidence

۳. پیشینه موضوع

نخستین و بی شک پایه‌ای ترین و موثرترین الگوریتم ممکن در زمینه کاوش قوانین انجمنی، الگوریتم Apriori [۳] می باشد. الگوریتم Apriori بر پایه یک فرضیه بنیان نهاده شده است که این فرضیه، به طوری چشمگیر فضای جستجوی الگوریتم را کاهش می دهد. "تمامی زیر مجموعه های یک مجموعه کالای k عضوی تکرار شونده، خود تکرار شونده هستند [۱]". این الگوریتم، پس از یک بار جستجوی کامل پایگاه داده، کالاهای تکرار شونده تک عضوی را شناسایی می کند. سپس با استفاده از کالاهای تکرار شونده تک عضوی، شروع به ساختن مجموعه های ۲ عضوی می کند و در حقیقت این مجموعه ها ر برای شمارش کاندید می کند. سپس، بار دیگر تمامی پایگاه داده را برای شمارش تعداد تکرار این مجموعه ها جستجو می کند. در نتیجه، الگوریتم Apriori، کل داده های پایگاه داده را برای یافتن کالاهای تکرار شونده k عضوی، k مرتبه جستجو می کند، تا هنگامی که بزرگترین مجموعه تکرار شونده بدست آید. روش Apriori از دیدگاه تئوریک به طور تضمینی تمامی جواب های ممکن برای یافتن مجموعه کالاهای تکرار شونده را می یابد اما در هنگام اجرای این الگوریتم به روی پایگاه های داده ی بزرگ، مشکلات فراوانی به وجود می آید.

اولا ساختن و کاندید کردن مجموعه کالاهای چند عضوی در هر مرحله، بالاخص در پایگاه های داده‌ی بزرگ بسیار زمان بر است. می توان تصور کرد که اگر تعداد کالاهای تکرار شونده تک عضوی، به بیش از ۱۰۰ عدد برسد، حداقل میزان $\binom{۱۰۰}{۲}$ یعنی ۴۹۵۰ مجموعه دو عضوی، برای جستجو کاندید می شوند، که ساختن و جستجوی این تعداد مجموعه زمان بسیار زیادی را هدر می دهد. ثانیاً، برای هر مرحله از الگوریتم Apriori، یک بار، تمامی پایگاه داده جستجو می شود که این امر در پایگاه های داده بسیار بزرگ زمان بر خواهد بود. بسیاری از الگوریتم ها مانند [۹,۸,۷,۶,۵] و [۱۰] در صدد بهبود این الگوریتم برآمدند.

الگوریتم های کاوش قوانین انجمی دارای ۳ فاکتور بسیار مهم می باشند [۱۰]:

۱. راهی که کاندیداهای جستجو، ساخته می شوند،
۲. ساختار پایگاه داده،
۳. جزئیات پیاده سازی الگوریتم.

هر الگوریتم، از یک یا چند فاکتور از فاکتورهای بالا برای بهبود روش های گذشته و یا ارائه یک روش جدید استفاده می کند.

الگوریتم FP-Growth[۷] نخستین الگوریتمی است که بدون تولید کاندیدا، تمامی قوانین انجمنی ممکن را کشف می کند و با تنها دو بار جستجوی پایگاه داده تمامی مجموعه های تکرار شونده را شناسایی می کند [۴]. الگوریتم FP-Growth از یک ساختار درختی شکل به نام ساختار درختی تکرار شونده استفاده می کند. این ساختار مانند یک پایگاه داده فشرده عمل کرده، جستجوی مجموعه ها را بسیار ساده تر می کند. این الگوریتم دارای سه ویژگی منحصر به فرد است :

۱. شیوه جستجوی استثنایی،
۲. استفاده ی بهینه از ساختار خلاصه،

[۷] Frequent Pattern Growth

۳. دو بار جستجوی پایگاه داده.

اما FP-Growth مشکلات خاص خود را نیز دارد. در مورد پایگاه های داده بسیار بزرگ، این الگوریتم برای ساختن درخت دچار مشکلات عدیده می شود.

بعضی دیگر از الگوریتم ها مانند [۱۱] سعی در استفاده از پایگاه های داده ی عمودی دارند. این نوع از پایگاه داده به جای اینکه لیست کالاهای موجود در هر سبد خرید را نشان دهد، سبد خرید هایی که هر کالا در آن ها وجود دارد را نشان می دهد.

الگوریتم هایی نیز وجود دارند که از ترکیب تمامی استراتژی های الگوریتم های پیشین استفاده می کنند، مانند [۱۳,۱۲,۱۱]. این الگوریتم ها، که در جرگه ی پیشرفته ترین الگوریتم های کاوش قوانین انجمنی قرار دارند، در صدد یافتن راه حل مناسبی هستند تا بتوانند ویژگی های منحصر به فرد همه الگوریتم ها را به بهترین شکل ممکن ترکیب کنند. همچنین در سال های اخیر تلاش هایی در زمینه کاوش قوانین انجمنی در مورد داده های خاص، مانند داده های تصویری، داده های ترتیبی و ...، صورت گرفته است [۱۶,۱۷,۱۸].

۴. الگوریتم MAKER

۱.۴. کدگذاری پایگاه داده و الگوریتم ارائه شده

کدگذاری را تغییر فرمت داده ها و یا رمزگذاری یک فرمت خاص داده، برای تبدیل آن نوع از داده به اطلاعات قابل استفاده تعریف کرده اند [۱۰]. معمولا در کدگذاری کالا ها در فروشگاه های بزرگ، عمل کدگذاری برای شناسایی و ذخیره راحت تر کالاها انجام می گیرد. برای این نوع کدگذاری از اعداد طبیعی، حروف لاتین و یا ترکیبی از هر دو استفاده می کنند. اما برای اکثر الگوریتم های قوانین انجمنی، نوع کدگذاری تاثیری در روند الگوریتم ندارد. ولی اگر یک پایگاه داده را بر مبنای اعداد اول کدگذاری کنیم، شرایط خاصی برای آن پایگاه داده ایجاد می شود که خصوصیت الگوریتم MAKER استفاده از همین شرایط خاص است.

یک مجموعه ی داده را مانند جدول ۱.۴ را در نظر می گیریم.

$T_۱$	۱	۲	۳	۴	۵
$T_۲$	۱	۲	۳	۵	۶
$T_۳$	۱	۲	۴	۶	۷
$T_۴$	۲	۳	۴	۵	۶
$T_۵$	۱	۵	۶	۷	۸

جدول ۱.۴: مثالی برای یک مجموعه داده.

مجمو عه داده ی فوق را به وسیله ماتریس $X_{(i\times j)}$ نشان می دهیم. سطرهای این ماتریس همان سبد های خرید هستند که به صورت $T_{(i)}$, $i = ۱,..,۵$ نشان داده می شوند. اگر ستون های ماتریس به تعداد $j = ۱,..,۵$ باشد، آنگاه متغیر $T_{(i,j)}$ به کالای شماره jام در سبد خرید $T_{(i)}$ اشاره می کند.

حال، به هر نوع کالای درون مجموعه داده یک عدد اول نسبت می دهیم. برای این کار، به تعداد کالا های درون مجموعه داده، k، عدد اول تولید کرده، هر عدد اول را به یک نوع کالای خاص، مانند جدول ۲.۴ تخصیص می دهیم. سطر دوم جدول ۲.۴ را به صورت متغیر

$$Product_{(k)} \ , \ k = ۱,..,۸$$

نشان می دهیم. برای مثال،

$$Product_{(۱)} = ۲.$$

اشاره به عدد اول تخصیص داده شده به کالای شماره ۱ دارد، که همان عدد ۲ است.

در نهایت، هر نوع کالای خاص موجود در پایگاه داده را، با عدد اول اختصاص داده شده به آن کالا تعویض می کنیم. نتیجه در جدول ۳.۴ نشان داده شده است.

کالا	۱	۲	۳	۴	۵	۶	۷	۸
عدد اول	۲	۳	۵	۷	۱۱	۱۳	۱۷	۱۹

جدول ۲.۴: تخصیص اعداد اول به هر کالا در مجموعه داده.

T۱	۲	۳	۵	۷	۱۱
T۲	۲	۳	۵	۱۱	۱۳
T۳	۲	۳	۷	۱۳	۱۷
T۴	۳	۵	۷	۱۱	۱۳
T۵	۲	۱۱	۱۳	۱۷	۱۹

جدول ۳.۴: مجموعه داده تغییر فرمت داده شده.

۲.۴. عدد سبد خرید

متغیر $Basket_{(i)}$ را عدد سبد خرید iام می گوییم، طوری که با فرض $i = m$:

$$Basket_{(m)} = \prod_{j=1}^{5} T_{(m,j)}.$$

جدول ۴.۴، اعداد سبد خرید را برای مجموعه داده جدول ۱.۴، نشان می دهد.

سبد خرید	عدد سبد خرید
T۱	۲۳۱۰
T۲	۴۲۹۰
T۳	۹۲۸۲
T۴	۱۵۰۱۵
T۵	۹۲۳۷۸

جدول۴.۴: اعداد سبد خرید برای مجموعه داده ۱.۴.

در پایگاه های داده بزرگ، با توجه به تعداد زیاد داده ها در هر سبد خرید، امکان این امر وجود دارد که عدد سبد خرید، عددی بسیار بزرگ باشد. در این صورت امکان ذخیره این عدد در یک حافظه با ظرفیت مشخص، مثل حافظه ۶۴ بیتی وجود ندارد. برای این منظور برای هر سبد خرید، ضرب های متوالی را تا جایی ادامه می دهیم که حاصل ضرب اعداد از بزرگترین عدد integer ۶۴ بیتی بزرگتر نشود. اگر کماکان اعداد اول دیگری در سبد خرید باقی بود، ضرب متوالی آنها را در یک حافظه ۶۴ بیتی دیگر قرار می دهیم و آنقدر این کار را تکرار می کنیم تا دیگر عددی در سبد خرید باقی نماند. این مراحل با ایجاد یک حلقه while در الگوریتم به انجام می رسد. در این صورت هر سبد خرید متشکل از چند عدد می گردد. برای مثال می توان جدول ۴.۵ را به شکل زیر نیز نوشت:

T۱	۶	۳۵	۱۱
T۲	۶	۵۵	۱۳
T۳	۶	۹۱	۱۷
T۴	۱۵	۷۷	۱۳
T۵	۲۲	۱۴۳	۱۹

جدول ۵.۴: مثالی برای چند عدد سبد خرید برای هر تراکنش.

در پایگاه های داده ای که سبد های خرید دارا ی تعداد کالای برابر نیستند، ممکن است که عدد یک سبد خرید در یک حافظه مشخص جا بگیرد، در حالی که سبد خرید دیگری دارای چند عدد سبد خرید باشد. در این حالت ابتدا سبد خریدی که بیشترین تعداد عدد سبد خرید را دارد را پیدا می کنیم و آن را T_{max} می نامیم. سپس، برای کل پایگاه داده ماتریسی تولید می کنیم که طول هر سطر آن برابر با تعداد اعداد سبد خرید T_{max} باشد. سپس، به جای خانه های خالی هر سبد خرید، عدد ۱ را جایگزین می کنیم..

در عین حال، در تمامی موارد، برای جلوگیری از گمراهی خواننده، ما برای هر تراکنش، یک عدد سبد خرید در نظر می گیریم، مگر در جایی که در روند الگوریتم ناثیر داشته باشد که در آن صورت، الگوریتم را برای آن حالت خاص تشریح می کنیم.

۳.۴. شیوه ی جستجوی روش MAKER.

برای $k = ۱..۸$، متغیری به نام $Count_{(k)}$ تعریف کرده، مقدار آن را برای همه عضو ها، ۰ قرار می دهیم. این متغیر تعداد تکرار هر کالا را در مجموعه ی داده شمارش می کند. حال، عدد سبد خرید mام، کالای lام و متغیر $Count_{(l)}$ را در نظر می گیریم. اگر:

$$mod\ \left[\frac{Basket_{(m)}}{Ptoduct_{(l)}}\right] = ۰,$$

باشد، به این معنی است که عدد اول تخصیص داده شده به کالای l ام، جزء مقسوم علیه های عدد سبد خرید است. به عبارت بهتر، کالایی که آن عدد اول به آن تخصیص داده شده، در آن سبد خرید وجود دارد. اگر باقی مانده این تقسیم ۰ نشد، کالای مورد نظر درآن سبد خرید وجود ندارد.

در صورت وجود کالای l ام در سبد خرید m ام، یک واحد به متغیر $Count_{(l)}$ اضافه می شود و در غیر این صورت، تغییری در مقدار آن حاصل نمی شود. در نتیجه پس از $k \times i = ۸ \times ۵ = ۴۰$ تقسیم، تعداد تکرار تمامی کالاهای موجود در مجموعه داده شمارش می شود. برای مثال، برای $Ptoduct_{(۱)} = ۲$ داریم:

m	i
۰	۲۳۱۰
۰	۴۲۹۰
۰	۹۲۸۲
۱	۱۵۰۱۵
۰	۹۲۳۷۸

جدول ۶.۴: نحوه ی جستجوی کالای شماره ۱ توسط Maker.

در نتیجه:

$$Count_{(۲)} = ۴۰$$

در پایان برای همه انواع کالا داریم:

کالا	۲	۳	۵	۷	۱۱	۱۳	۱۷	۱۹
$Count_{(k)}$	۴	۳	۳	۳	۳	۳	۲	۱

جدول ۷.۴: تعداد تکرار مجموعه کالاهای تک عضوی در مجموعه داده.

با فرض اینکه عدد ۳، آستانه تکرار شوندگی است، اعداد زیر به عنوان مجموعه کالا های تکرار شونده تک عضوی معرفی می شوند:

کالا	۲	۳	۵	۷	۱۱	۱۳
$Count_{(k)}$	۴	۳	۳	۳	۳	۳

جدول ۸.۴: مجموعه های تک عضوی تکرار شونده.

در این مرحله، به ساختن تمامی مجموعه های ۲ عضوی از عناصر تکرار شونده تک عضوی می پردازیم تا بعدا، وجود یا عدم وجود این مجموعه ها در هر سبد خرید مورد بررسی قرار گیرد. جدول ۴.۸ شامل همه مجموعه های ۲ عضوی ساخته شده از عناصر تکرار شونده تک عضوی بالاست. برای مجموعه های زیر نیز، متغیر$k= ۱..۱۵$, $Count_{(k)}$ را تعریف می کنیم تا تعداد تکرار هر مجموعه را در خود ثبت کند.

۳و۲	۵و۳	۷و۵	۱۱و۷	۱۳و۱۱	۵و۲	۷و۳	۱۱و۵
۱۳و۷	۷و۲	۱۱و۳	۱۳و۵	۱۱و۲	۱۳و۳	۱۳و۲	

جدول ۹.۴: مجموعه های ۲ عضوی کاندید برای جستجو.

۴.۴. پیدا کردن مجموعه های تکرار شونده چند عضوی

برای جستجوی یک مجموعه چند عضوی در یک سبد خرید، کافی است که اولین عضو مجموعه، به همان شیوه ذکر شده در بالا، در یک سبد خرید جستجو شود. در صورت اثبات وجود آن کالا در آن سبد خرید، عضو دوم مورد جستجو قرار می گیرد و در غیر این صورت مجموعه چند عضوی بعدی در آن سبد خرید جستجو می شود. اگر تمامی عضوهای یک مجموعه در آن سبد خرید وجود داشت، آنگاه به متغیر *Count* آن مجموعه ۱ واحد افزوده می شود و مجموعه بعدی در آن سبد خرید جستجو می شود. این روند تا جایی ادامه پیدا می کند که تمامی

مجموعه های چند عضوی مورد جستجو قرار بگیرند. توجه داشته باشید که در این مرحله، الگوریتم فقط مجموعه های ۲ عضوی را مورد جستجو قرار می دهد.

برای مثال بالا، مجموعه های زیر به عنوان مجموعه های تکرار شونده ۲ عضوی نشان داده می شود:

۳و۲	۵و۳	۱۳و۱۱	۱۱و۵	۱۱و۳	۱۱و۲	۱۳و۳	۱۳و۲

جدول ۱۰.۴: مجموعه های ۲ عضوی تکرار شونده.

مراحل بالا را برای مجموعه های ۳ عضوی تکرار می کنیم. ابتدا مجموعه های ۳عضوی را از روی مجموعه های ۲ عضوی می سازیم. نتیجه را در جدول ۱۱.۴ مشاهده می کنید.

۵و۱۱و۱۳	۳و۵و۱۳	۳و۵و۱۱	۲و۳و۱۳	۲و۳و۱۱	۲و۳و۵

جدول ۱۱.۴: مجموعه های ۳ عضوی کاندید برای جستجو.

بنا بر قانون Apriori، مجموعه های (۲و۳و۵)، (۳و۵و۱۳) و (۵و۱۱و۱۳) نمی- توانند تکرار شونده باشند، زیرا بعضی از زیر مجموعه های آن ها مثل (۲و۵) و (۱۳و۵) تکرار شونده نیستند.

در نتیجه، در این مرحله فقط به جستجوی سه مجموعه دیگر می پردازیم. دراینجا، تنها مجموعه (۳و۵و۱۱) تکرار شونده است و در نتیجه می توان فهمید که هیچ مجموعه تکرار شونده ۴ عضوی در مورد این مثال موجود نمی باشد.

اما در حالت کلی از روی مجموعه های ۳ عضوی، مجموعه های ۴ عضوی و سپس مجموعه های ۵ عضوی را پیدا کرده و مراحل بالا را برای پیدا کردن مجموعه های تکرار شونده تکرار می کنیم.

۵. آزمایش ها و اعتبار سنجی

تمامی آزمایش ها روی پایگاه های داده، به وسیله رایانه با سیستم عامل Windows XP با مشخصات HDD=۳۲۰Ghz، Processor=۳.۲Ghz و حافظه جانبی ۱G صورت گرفته است. الگوریتم MAKER به روی سه پایگاه داده استاندارد آزمایش شده است. این پایگاه های داده از روی پایگاه اینترنتی کنفرانس بین المللی داده کاوی که در سال ۲۰۰۴ میلادی در در کشور فنلاند برگزار شد،، استخراج شده است [۲۱]. در جدول ۱.۵ نام پایگاه داده ی استخراج شده با برخی از خصوصیات آنها آمده است. ضمن اینکه برای هر پایگاه داده، توضیحات کافی و نتایج حاصل از آزمایش الگوریتم MAKER به روی آن پایگاه داده، در ذیل آمده است. جواب های حاصله، تنها مجموعه های تکرار شونده را در بر می گیرد، و به محاسبه فاکتور اطمینان برای مجموعه ها نمی پردازد. جواب های بدست آمده از کاوش این پایگاه های داده، بر اساس مقایسات انجام شده، کاملا دقیق بوده، و با نتایج الگوریتم Apriori در نرم افزار Clementine به طور صد در صد هم خوانی دارد.

نام پایگاه داده	تعداد سبد خرید	اندازه ی سبد خرید	تعداد نوع کالا
MUSHROOM	۸۱۲۴	۲۳	۱۱۹
CHESS	۳۱۹۶	۳۷	۷۴
CONNECT	۶۷۵۵۷	۴۳	۱۲۷

جد

پایگاه داده Mushroom

این پایگاه داده که حاوی اطلاعاتی در مورد انواع گوناگون قارچ ها می باشد، با سه آستانه تکرار شوندگی مختلف توسط الگوریتم کاوش شده است. این آستانه های تکرار شوندگی عبارتند از: ۵۰٪، ۶۵٪ و ۸۰٪. نتایج آماری حاصل از این کاوش را در جدول ۲.۵ مشاهده می فرمایید.

پایگاه داده Chess

این پایگاه داده که حاوی اطلاعاتی در مورد بازی شطرنج در یک محل بازی می باشد، با آستانه های تکرار شوندگی گوناگون توسط الگوریتم کاوش شده است. این آستانه های تکرار شوندگی عبارتند از: ۵۰٪، ۶۵٪ و ۸۰٪. نتایج آماری حاصل از این کاوش را در جدول ۳.۵ مشاهده می فرمایید.

پایگاه داده Connect

این پایگاه داده که حاوی اطلاعاتی در مورد بازی های مختلف در یک محل می باشد، با سه آستانه های تکرار شوندگی گوناگون توسط الگوریتم کاوش شده است. این آستانه های تکرار شوندگی عبارتند از: ۵۰٪، ۶۵٪ و ۸۰٪. نتایج آماری حاصل از این کاوش را در جدول ۴.۵ مشاهده می فرمایید.

Min-sup	تعداد مجموعه های تکرار شونده های تک عضوی	تعداد مجموعه های تکرار شونده های ۲ عضوی	تعداد مجموعه های تکرار شونده های ۳ عضوی	تعداد مجموعه های تکرار شونده های ۴ عضوی
۵۰	۱۳	۴۱	۵۶	۳۵
۶۵	۶	۱۳	۱۳	۶

۸۰	۵	۹	۷	۲

جدول ۲.۵: نتایج آزمایش الگوریتم MAKER به روی پایگاه داده Mushroom.

Min-sup	تعداد مجموعه های تکرار شونده های تک عضوی	تعداد مجموعه های تکرار شونده های ۲ عضوی	تعداد مجموعه های تکرار شونده های ۳ عضوی	تعداد مجموعه های تکرار شونده های ۴ عضوی
۵۰	۳۷	۵۳۱	۴۰۰۰	۱۸۵۶۵
۶۰	۳۴	۳۸۹	۲۳۲۵	۸۸۳۱
۸۰	۱۹	۱۴۱	۵۶۶	۱۳۸۳

جدول ۳.۵: نتایج آزمایش الگوریتم MAKER به روی پایگاه داده Chess.

Min-sup	تعداد مجموعه های تکرار شونده های تک عضوی	تعداد مجموعه های تکرار شونده های ۲ عضوی	تعداد مجموعه های تکرار شونده های ۳ عضوی	تعداد مجموعه های تکرار شونده های ۴ عضوی
۵۰	۳۸	۶۳۳	۶۲۸۵	۱۰۸۵۷
۷۵	۳۰	۳۷۹	۲۷۵۷	۷۸۴۵
۹۰	۲۱	۱۷۷	۸۲۶	۲۴۵۱

جدول ۴.۵: نتایج آزمایش الگوریتم MAKER به روی پایگاه داده Connect.

نتایج زمانی برای چند آستانه ی تکرار شوندگی را در جدول ۵.۵ مشاهده می فرمایید.

پایگاه داده	آستانه تکرار شوندگی	زمان
CHESS	۸۰٪	۲۱^{S}
CHESS	۶۰٪	۳۶۰^{S}
CONNECT	۷۵٪	۳۶۰^{S}
MUSHROOM	۸۰٪	۰.۳۷۹^{S}
MUSHROOM	۶۵٪	۰.۵۸۳^{S}
MUSHROOM	۵۰٪	۶^{S}

جدول ۵.۵: بعضی از نتایج زمانی حاصل از آزمایش الگوریتم MAKER به روی پایگاه های داده استاندارد.

۶. نتیجه گیری

در این تحقیق سعی در ارائه ی روش نوینی در زمینه ی کاوش قوانین انجمنی با استفاده از تخصیص اعداد اول به کالاهای پایگاه داده شده است. این الگوریتم در زمره ی الگوریتم های پله ای قرار می گیرد، زیرا در مرحله K (مرحله ی K ام) مجموعه های تکرار شونده ی K عضوی را از درون پایگاه داده کشف می کند. با توجه به اینکه ضرب اعداد اول عددی را به دست می دهد که با تجزیه ی آن می توان به همان اعداد اولیه دست پیدا کرد، می توان برای هر سبد خرید در درون یک پایگاه داده، یک و فقط یک عدد یکتا مشخص کرد. از این رو استفاده از این روش حجم بسیار کمی را طلب می کند.

مراجع

[۱] J. Han, M. Kamber, *Data Mining: Concepts And Techniques*, Morgan Kaufmann Publishers, March ۲۰۰۶, ISBN ۱-۵۵۸۶۰-۹۰۱-۶.

[۲] D. L. Olson, D. Derren, *Advanced Data Mining Techniques,* Springer Publisher, ۲۰۰۸, XII, ۱۸۰ p. ۲۱ illus., Softcover, ISBN: ۹۷۸-۳-۵۴۰-۷۶۹۱۶-۳.

[۳] R. Agrawal and R. Srikant. *Fast algorithms for mining association rules*. In J. Bocca, M. Jarke, and C. Zaniolo, editors, Proceedings of the ۲۰th International Conference on Very.Large Data Bases (VLDB'۹٤), Santiago de Chile, September ۱۲-۱۵, pages ٤۸۷–٤۹۹. Morgan Kaufmann, ۱۹۹٤.

[٤] J. Han, J. Pei, and Y. Yin. *Mining frequent patterns without candidate generation*. In W. Chen, J. F. Naughton, and P. A.Bernstein, editors, Proceedings of the ۲۰۰۰ ACM SIGMOD.International Conference on Management of Data, pages ۱–۱۲. ACM Press, ۲۰۰۰.

[۵] H. Toivonen. *Sampling large databases for associationrules*. In VLDB, pages ۱۳٤–۱٤۵, ۱۹۹٦.

[٦] S. Orlando, P. Palmerini, R. Perego, and F. Silvestri. *Adaptive and Resource-Aware Mining of Frequent Sets. In Proc*. The ۲۰۰۲ IEEE International Conference on. Data Mining (ICDM'۰۲), pages ۳۳۸–۳٤۵, ۲۰۰۲.

[۷] Rakesh Agrawal. *Parallel Mining of Association Rules:Design, Implementation and Experience*.John C. Shafer, ۱۹۹٤.

[۸] EH Han, G Karypis, V Kumar. *Scalable parallel data mining for association rules*. ACM SIGMOD Record, ۱۹۹۷.

[۹] Ferenc Bodon, *A fast APRIORI implementation* In *FIMI*, ۲۰۰۳.

[۱۰] Walter A. Kosters, Wim Pijls, *APRIORI, A Depth First Implementation.* In *FIMI*, ۲۰۰۳.

[۱۱] Mohammed J. Zaki. *Scalable algorithms for association mining*. *IEEE Transactions on Knowledge and Data Engineering*, ۱۲(۳):۳۷۲-۳۹۰, May/June ۲۰۰۰.

[۱۲] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhai. *Mining frequent patterns with counting inference*. *SIGKDD Explorations*, ۲(۲):٦٦-۷۵, ۲۰۰۰.

[۱۳] Salvatore Orlando, Claudio Lucchese, Paolo Palmerini, Raffaele Perego, and Fabrizio Silvestri. kdci: a *multi-strategy algorithm for mining frequent sets*. In Bart Goethals and Mohammed J. Zaki, editors, FIMI'۰۳: Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, November ۲۰۰۳.

[۱٤] A.Fiat and S. Shporer. *Aim: Another itemset miner*. In *FIMI*, ۲۰۰۳.

[۱۵] S. Shporer. *AIM ۲: Improved implementation of AIM*. In *FIMI*, ۲۰۰٤.

[۱٦] Anthony J.T. Lee, Ying-Ho Liu, Hsin-Mu Tsai, Hsiu-Hui Lin, Huei-Wen Wu Mining frequent *patterns in image databases with ۹D-SPA representation*. Journal of Systems and Software, Volume ۸۲, Issue ٤, Pages ٦۰۳-٦۱۸, April ۲۰۰۹.

[١٧] Tzung-Shi Chen, Shih-Chun Hsu. *Mining frequent tree-like patterns in large datasets* Data & Knowledge Engineering, Volume ٦٢, Issue ١, Pages ٦٥-٨٣, July ٢٠٠٧.
[١٨] Syed Khairuzzaman Tanbeer, Chowdhury Farhan Ahmed, Byeong-Soo Jeong, Young-Koo Lee. *Sliding window-based frequent pattern mining over data streams.* Information Sciences, Volume ١٧٩, Issue ٢٢, Pages ٣٨٤٣-٣٨٦٥ ٧ November ٢٠٠٩.
[١٩] B. Goethals, M. J. Zaki, *FIMI ٠٣: Workshop on Frequent Itemset Mining Implementations*, in FIMI ٢٠٠٣.
[٢٠] Bob Boiko, *Content Management Bible,* pp:٧٩, ٢٤٠, ٨٣٠, Nov ٢٠٠٤.
[٢١] Http://fimi.cs.helsinki.fi/data, ٢٠٠٣.