# مقاله مروری

# بهبود نسبت فشرده سازی متن برای متن ASCII

**مریم سادات موردگر**

دانشگاه پیام نور تهران  پست الکترونیک: M.mourdgar@gmail.com

**چکیده**

فشرده سازی داده ها زمینه ای است که به طور گسترده مورد تحقیق قرار گرفته است. بیشتر الگوریتم های فشرده سازی که امروزه مورد استفاده قرار می گیرند، از چندین دهه پیش است، مانند *کدگذاری هافمن* و *کدگذاری دیکشنری که* همه منظوره هستند. الگوریتم های فشرده سازی و می تواند در هر چیزی از داده های متنی گرفته تا تصاویر و ویدئو استفاده شود. با این حال، الگوریتم های بسیار کمی برای فشرده سازی انواع خاصی از داده ها، مانند متن اسکی وجود دارد که بدون اتلاف داده باشند. این پروژه در مورد ایجاد یک راه حل مخصوص فشرده سازی متن با استفاده از ترکیبی از سه مورد است الگوریتم فشرده سازی کدگذاری دیکشنری، الگوریتم فشرده سازی اسکی و الگوریتم فشرده سازی کدگذاری هافمن.

نتایج حاصل از ارزیابی نشان می دهد که ترکیب کدگذاری دیکشنری، فشرده سازی اسکی و کدینگ هافمن از نسبت فشرده سازی به دست آمده از برنامه های فشرده سازی عمومی بهتر نمی‌باشد [۳].

**واژگان کلیدی: فشرده سازی متن، کدگذاری دیکشنری، اسکی، کدینگ هافمن**

## ۱. مقدمه

فشرده سازی داده ها فرایند کاهش تعداد بیت های مورد نیاز برای نمایش رسانه است [۱]. زیر مجموعه فشرده سازی داده ها فشرده سازی متن است که شامل کاراکترها و نمادها می باشد. یک تفاوت مهم بین فشرده سازی متن و داده های باینری، مانند تصاویر و ویدئو، این است که فشرده سازی متن باید بدون اتلاف باشد یعنی فایل اصلی بدون از دست دادن اطلاعات بازیابی شود [۱]. الگوریتم های فشرده سازی که قادر به فشرده سازی داده ها نیستند از فشرده سازی با اتلاف استفاده می کنند که در آن برخی از داده ها از بین می روند [۲].

## ۲. تاریخچه

فشرده سازی داده ها توسط الگوریتم های فشرده سازی انجام می شود. چندین الگوریتم متداول در برنامه های فشرده سازی استفاده می شود، و بسیاری از آن ها چندین دهه است که وجود دارند. الگوریتم متداولی که هنوزمورد استفاده قرار می گیرد، کدگذاری هافمن است که در سال ۱۹۵۲ اختراع شد [۵]. کدگذاری هافمن کاراکترهای پرکاربردتر را با رشته‌های بیتی کوتاه‌تری نسبت به آن‌هایی که کاربردشان کمتر است، نشان داده می‌شوند. یکی دیگر از الگوریتم ها، کدگذاری دیکشنری است که تکنیک جایگزینی زیر رشته ها و کلمات با ارجاع به دیکشنری است. دیکشنری و منابع، فایل فشرده را تشکیل می دهند که می توان برای بازیابی فایل اصلی آن ها را از حالت فشرده خارج کرد. LZ77 و LZW از پرکاربردترین الگوریتم های کدگذاری دیکشنری هستند [۶]. هر دو الگوریتم مبتنی بر تکنیک جایگزینی زیر رشته های تکراری با منابع هستند، اما از رویکردهای متفاوتی استفاده می کنند.
الگوریتم DEFLATE ترکیبی از کدگذاری دیکشنری (LZ77) و کدگذاری هافمن است. DEFLATE در چندین برنامه فشرده سازی مانند Gzip، zip و WinRar استفاده می شود . با وجود این که در زمینه فشرده سازی داده‌ها تحقیقات گسترده‌ای انجام شده است، فناوری‌های جدید همچنان در حال ظهور هستند، مانند شوکو ، که در سال ۲۰۱۴ منتشر شد. شوکو یک برنامه فشرده سازی متن است که به ویژه بر روی متن ASCII موثر است. ASCII یک کدگذاری برای متن ساده است؛ یعنی متنی که از تنوع زیادی از کاراکترها و نمادها استفاده نمی کند. رمزگذاری کاراکتر یک متن به کامپیوتر می گوید چگونه داده ها باید خوانده شوند تا انسان بتواند آن را درک کند.

## ۳. طرح پیشنهادی

طرح پیشنهادی یک راه حل فشرده سازی مخصوص متن ASCII به نام TCS است که از سه ماژول تشکیل شده است: ماژول فشرده سازی ASII (ACM)، ماژول کدگذاری دیکشنری و ماژول کدگذاری هافمن. ماژول ها در C یا Python نوشته شده اند و می توانند به صورت متوالی با استفاده از یک برنامه پایتون یا یک اسکریپت پوسته یونیکس اجرا شوند.
TCS ابتدا برای ACM در نظر گرفته شده است، سپس رمزگذار دیکشنری و در نهایت ماژول کدگذاری هافمن استفاده کند. ACM ابتدا استفاده می شود زیرا فقط قادر به فشرده سازی کاراکترهای اسکی است. خروجی رمزگذار دیکشنری یا ماژول های کدگذاری هافمن، داده های دودویی هستند که شامل کاراکترهای اسکی نیستند، بنابراین این ماژول ها نباید قبل از ACM استفاده شوند. خروجی ACM یک فایل متنی در کدگذاری cpo37 است. این خروجی به کدگذار دیکشنری ارسال می شود که زیر رشته های تکراری را در متن جستجو می کند. سپس خروجی دودویی از رمزگذار دیکشنری به ماژول کدگذاری هافمن ارسال می شود که مقادیر بایت را که بیشتر از بقیه استفاده می شود، جستجو می کند. فایل خروجی از ماژول کدگذاری هافمن آخرین فایل فشرده شده از TCS است.

## ۴. الزامات پیاده سازی و مجموعه داده ها

۱- الزامات عملکردی TCS عبارتند از:

- در طول فشرده سازی هیچ داده ای از بین نمی رود.
- TCS می تواند متن را با رمزگذاری های معمول (UTF-8 و ASCII) و غیرمعمول فشرده کند.
- ماژول های TCS به راحتی متصل می شوند.

۲- الزامات غیر کاربردی برای TCS عبارتند از:

- TCS می تواند فایل های بالای ۲۰ مگابایت را فشرده کند.
- TCS می تواند فایلها را حداقل به نصف اندازه اصلی خود فشرده کند.
- تمام کدهای شخص ثالثی که در TCS استفاده می شود باید منبع باز باشد.

مجموعه داده های مورد بررسی، نمونه هایی از فایل های متنی بزرگتر هستند که کاربر ممکن است TCS را روی آنها اعمال کند. مجموعه داده ها شامل ۶ متن با ویژگی های مختلف است؛

۱- یک فایل XML ویکی پدیا. رمزگذاری UTF-8 ؛ ۲۱,۶ مگابایت

۲- یک فایل XML ویکی پدیا. کدگذاری cpo37 ؛ ۱۸,۱ مگابایت

۳- یک فایل کد نوشته شده با C. کدگذاری ASCII؛ ۶۴ کیلوبایت

۴- کتابی که به زبان انگلیسی نوشته شده است. رمزگذاری UTF-8 ؛ ۶۸۰ کیلوبایت

۵- کتابی که به زبان ایتالیایی نوشته شده است. رمزگذاری UTF-8 ؛ ۶۲۶ کیلوبایت

۶- کتابی که به زبان چینی نوشته شده است. رمزگذاری UTF-8 ؛ ۲۸۵ کیلوبایت

### ۵. انتخاب پیاده سازی ماژول ها

برای پیاده سازی کدگذاری دیکشنری از سه الگوریتم LZ77 (با پیاده سازی پایتون) و LZ77 (با پیاده سازی C) و LZW (با پیاده سازی C)، روی مجموعه داده منتخب استفاده شده است. با مقایسه این سه الگوریتم نتایج زیر حاصل شد:

اجرای پایتون از الگوریتم LZ77 بدترین نسبت فشرده سازی را دارد. پیاده سازی C برای الگوریتم LZ77 دارای نسبت فشرده سازی بسیار بهتری است، و پیاده سازی LZW نسبت به پیاده سازی C در LZ77 برای برخی از متون دارای نسبت فشرده سازی بالاتری است، اما برای متون دیگر نسبت کمتری دارد.

بنابراین پیاده سازی C الگوریتم LZW به عنوان ماژول کدگذاری دیکشنری استفاده شده است.

برای کدگذاری هافمن از دو پیاده سازی C، که توسط کاربران گیت هاب Yaikhom و Richardson ساخته شده و یک پیاده سازی پایتون به نام Dahuffman که توسط کاربر گیت هاب Lippens ساخته شده است، استفاده شده است. از مقایسه این سه الگوریتم نتایج زیر حاصل شد:

دو پیاده سازی C نتایج دقیقا یکسانی را به دست آوردند، و پیاده سازی پایتون، Dahuffman، نسبت فشرده سازی کمی بهتر از دو پیاده سازی C برای کتاب انگلیسی و ایتالیایی، اما نسبت بهتری برای کتاب چینی دارد.

Dahuffman، دارای بالاترین نسبت فشرده سازی متوسط پیاده سازی ها است و بنابراین به عنوان ماژول کدگذاری هافمن برای TCS استفاده شده است.

برنامه فشرده سازی TCS شبیه به Shoco است. هر دو TCS و Shoco برنامه های فشرده سازی متن بدون اتلاف هستند که در فشرده سازی متن ASCII تخصص دارند. یک تفاوت اصلی این است که Shoco یک الگوریتم است در حالی که TCS یک راه حل

فشرده سازی است که از ۳ الگوریتم کدگذاری LZW، ACM و Huffman تشکیل شده است. ACM عملکرد و راه حل بسیار مشابهی با Shoco دارد.

ارزیابی ACM شامل اندازه گیری نسبت فشرده سازی آن برای همه متون مجموعه داده و مقایسه نتایج با Shoco است. مجموعه داده ها شامل ۶ متن با ویژگی های متفاوت است. برخی از متون به طور کامل یا عمدتا کاراکترهای ASCII هستند و متون دیگر عمدتا یا به طور کامل کاراکتر UTF-8 هستند. متون UTF-8 برای آزمایش اینکه چگونه ACM و Shoco می توانند فایل هایی را که مورد استفاده نیستند فشرده کنند، گنجانده شده است.

| Data set | مجموعه داده ها | ACM | Shoco پیش فرض | Shoco آموزش دیده |
|---|---|---|---|---|
| XML file (21.6 MB) | فایل XML | 1.19 | 1.23 | 1.4 |
| cpo37 encoded XML file (18.1 MB) | فایل XML کدگذاری cpo37 | 1 | 0.59 | 0.75 |
| C code file (64 KB) | فایل کد C | 1.21 | 1.17 | 1.35 |
| English book (680 KB) | کتاب انگلیسی | 1.46 | 1.32 | 1.56 |
| Italian book (626 KB) | کتاب ایتالیایی | 1.29 | 1.15 | 1.5 |
| Chinese book (285 KB) | کتاب چینی | 1 | 0.53 | 0.75 |
| Average | میانگین | 1.19 | 1 | 1.22 |

جدول ۱. نتایج حاصل از مقایسه ACM و Shoco

جدول ۱ نشان می دهد، نسبت فشرده سازی متن های مختلف متفاوت است. این انتظار می رود، زیرا ACM و Shoco باید نسبت فشرده سازی بیشتری برای متون سنگین اسکی داشته باشند زیرا آنها فقط می توانند کاراکترهای ASCII را فشرده کنند.

## ۶. ارزیابی TCS

ارزیابی TCS شامل اندازه گیری نسبت فشرده سازی آن برای همه متون موجود در مجموعه داده ها و سپس مقایسه نتایج با سایر برنامه های فشرده سازی عمومی است. وقتی TCS ارزیابی شده است، از هر سه ماژول استفاده کرده است. این بدان معناست که خروجی از یک ماژول به ماژول بعدی ارسال می شود، نسبت فشرده سازی برای خروجی ماژول قبلی اندازه گیری شده است. این آگاهی را می دهد که هر ماژول چقدر از فشرده سازی را بر عهده دارد. در نهایت، نسبت تراکم کل اندازه گیری می شود. این نسبت اندازه فایل اصلی در مقایسه با اندازه نهایی فایل فشرده است.

| Data set | مجموعه داده ها | ACM | LZW | Huffman coding | Total ratio |
|---|---|---|---|---|---|
| XML file (21.6 MB) | فایل XML | 1.19 | 1.6 | 1.01 | 1.93 |
| cpo37 encoded XML file (18.1 MB) | فایل XML کدگذاری cpo37 | 1 | 1.6 | 1.01 | 1.62 |
| C code file (64 KB) | فایل کد C | 1.21 | 1.83 | 1.01 | 2.23 |
| English book (680 KB) | کتاب انگلیسی | 1.46 | 1.36 | 1 | 1.99 |
| Italian book (626 KB) | کتاب ایتالیایی | 1.29 | 1.57 | 1 | 2.02 |
| Chinese book (285 KB) | کتاب چینی | 1 | 1.59 | 1 | 1.59 |
| Average | میانگین | 1.19 | 1.59 | 1.01 | 1.9 |

جدول ۴-۴. نتایج حاصل از ارزیابی TCS

همانطور که جدول ۲ نشان می دهد، برخی از فایل ها بیشتر از ACM و برخی دیگر از LZW سود بیشتری می برند. کدگذاری هافمن تقریبا هیچ فشرده سازی را برای هیچ یک از فایل ها بدست نمی آورد. این می تواند به این دلیل باشد که برنامه نویسی هافمن برای فشرده سازی داده های باینری طراحی نشده است. همچنین ممکن است خروجی دودویی از LZW بسیار تصادفی و نامنظم باشد تا کدگذاری هافمن نتواند به فشرده سازی برسد. نتایج حاصل از این ارزیابی نیاز به ارزیابی جدیدی دارد که فقط از کدگذاری LZW و هافمن استفاده می شود، تا به پاسخ این سوال برسیم که آیا ACM قابلیت های ماژول کدگذاری هافمن را مختل می کند یا این ماژول قادر به فشرده سازی داده های باینری نیست.
نتایج این ارزیابی نشان می دهد که حتی وقتی از ACM استفاده نمی شود، ماژول کدگذاری هافمن عملاً قادر به فشرده سازی داده های باینری نیست.

### ۷. نتایج حاصل از مقایسه TCS

برنامه های فشرده سازی مورد استفاده برای مقایسه از الگوریتم های فشرده سازی بدون اتلاف استفاده می کنند که می تواند در انواع داده ها، هم در فایل های متنی و هم در فایل های باینری مانند تصویر، صدا یا ویدئو استفاده شود. این برنامه ها برای مقایسه استفاده شده اند زیرا عموماً برای همان وظایفی که TCS در نظر گرفته است استفاده شده اند. به عنوان مثال، ویکی پدیا سرویسی برای بارگیری همه مقالات پایگاه داده خود در قالب فقط متن دارد. این فایل های متنی با Bzip2، Zipیا Zip-7 فشرده شده اند.
TCS با برنامه های فوق مقایسه شده است. Gzip همچنین در مقایسه استفاده شده زیرا یک برنامه فشرده سازی محبوب است که امروزه مورد استفاده قرار گرفته است [۹] و [۱۰].

| Data set | مجموعه داده ها | TCS | Bzip2 | Zip | 7-Zip | Gzip |
|---|---|---|---|---|---|---|
| XML file (21.6 MB) | فایل XML | 1.93 | 3.79 | 2.96 | 4.24 | 2.96 |
| cpo37 encoded XML file (18.1 MB) | فایل XML کدگذاری cpo37 | 1.62 | 3.12 | 2.45 | 3.48 | 2.45 |
| C code file (64 KB) | فایل کد C | 2.23 | 4.22 | 3.88 | 4.25 | 3.91 |
| English book (680 KB) | کتاب انگلیسی | 1.99 | 3.52 | 2.65 | 3.18 | 2.65 |
| Italian book (626 KB) | کتاب ایتالیایی | 2.02 | 3.54 | 2.66 | 3.14 | 2.66 |
| Chinese book (285 KB) | کتاب چینی | 1.59 | 2.59 | 2.01 | 2.35 | 2.01 |
| Average | میانگین | 1.9 | 3.46 | 2.77 | 3.44 | 2.77 |

جدول ۳. نتایج حاصل از مقایسه TCS

همانطور که جدول ۳ نشان می دهد که TCS نسبت به سایر برنامه ها برای هر فایل در مجموعه داده نسبت فشرده سازی کمتری دارد. Bzip2 و 7-Zip بالاترین میانگین نسبت فشرده سازی را دارند زیرا از الگوریتم های متفاوتی نسبت به Zip و Gzip استفاده می کنند که بر اساس الگوریتم DEFLATE است (Korpela, 2006). TCS از راه حلی مشابه DEFLATE، اما با ACM به عنوان پیش پردازنده استفاده می کند. الگوریتم DEFLATE ترکیبی از رمزگذار دیکشنری LZ77 و کدگذاری هافمن است، در حالی که TCS از رمزگذار دیکشنری LZW و کدگذاری هافمن استفاده می کند. جدول ۲ نشان می دهد که ماژول کدگذاری هافمن مورد استفاده برای TCS با ماژول LZW سازگار نیست و بنابراین TCS جایگزینی برای الگوریتم DEFLATE ندارد. برای بررسی اینکه آیا TCS در صورت جایگزینی ماژول کدگذاری LZW و Huffman که در TCS با یک الگوریتم DEFLATE خالص جایگزین

شده است، نتایج بهتری خواهد گرفت یا خیر، باید ارزیابی جدیدی انجام شود. در این ارزیابی، از برنامه های Zip و Gzip با ACM به عنوان پیش پردازنده استفاده شده است. این ارزیابی به این سوال پاسخ می دهد که آیا استفاده از ACM به عنوان پیش پردازنده باعث افزایش نسبت فشرده سازی الگوریتم DEFLATE می شود یا خیر.

| Data set | مجموعه داده ها | Zip | ACM + Zip | Gzip | ACM + Gzip |
|---|---|---|---|---|---|
| XML file (21.6 MB) | فایل XML | 2.96 | 2.92 | 2.96 | 2.92 |
| C code file (64 KB) | فایل کد C | 3.88 | 3.77 | 3.91 | 3.82 |
| English book (680 KB) | کتاب انگلیسی | 2.65 | 2.66 | 2.65 | 2.66 |
| Italian book (626 KB) | کتاب ایتالیایی | 2.66 | 2.58 | 2.66 | 2.58 |
| Average | میانگین | 3.04 | 2.98 | 3.05 | 3 |

جدول۴. نتایج حاصل از مقایسه ACM + DEFLATE

XML کد شده cpo37 و کتاب چینی در این ارزیابی گنجانده نشده اند زیرا ACM هیچ تاثیری بر روی این فایل ها نخواهد داشت. همانطور که جدول ۴ نشان می دهد، هنگامی که ACM به عنوان پیش پردازنده Zip یا Gzip استفاده می شود، نسبت فشرده سازی کمی پایین تر است. ACM فقط نسبت به کتاب انگلیسی نسبت فشرده سازی را تا حدی بیشتر نشان می دهد، اما افزایش ۰,۰۱ نسبت فشرده سازی ناچیز است. استفاده از ACM به علاوه الگوریتم DEFLATE که Zip و Gzip استفاده می کند نسبت فشرده سازی بسیار بالاتری نسبت به TCSدارد.

## ۸. جمع بندی

نتایج حاصل از ارزیابی TCS نشان می دهد که در حالت فعلی، مدعی سایر برنامه های فشرده سازی عمومی نیست. مقایسه فشرده سازی اسکی با الگوریتم Shoco نشان می دهد که هنگام استفاده از مدل های فشرده سازی عمومی، ACM نسبت فشرده سازی متوسط بهتری دارد. بنابراین ACM می تواند مدعی رشته تخصصی تر فشرده سازی ASCII باشد. ارزیابی ACM نشان می دهد که می توان آن را در هر شکلی از متن بدون افزایش اندازه فشرده استفاده کرد، اما نسبت فشرده سازی قابل توجهی فقط در متون سنگین ASCII به دست می آید. با این وجود، نسبت فشرده سازی ACM در مقایسه با برنامه های فشرده سازی عمومی بسیار کوچک است. ترکیب ACM با برنامه های فشرده سازی مبتنی بر DEFLATE نسبت فشرده سازی متوسط را فقط تحت DEFLATE نشان می دهد، با یک مثال از مزیت جزئی ACM. ممکن است بهبود الگوریتم فشرده سازی ACM نسبت فشرده سازی آن را افزایش دهد و حتی ممکن است از DEFLATE فراتر رود (در صورتی که ACM به عنوان پیش پردازنده DEFLATE استفاده شود).

## ۹. پیشنهادات و کارهای آینده

کارهای آینده در TCS شامل یافتن یک ماژول کدگذاری هافمن است که می تواند خروجی دودویی را از ماژول کدگذاری دیکشنری فشرده کند. دستیابی به این امر می تواند قابلیت فشرده سازی TCS را با الگوریتم DEFLATE برابر کند. همچنین آزمایش الگوریتم هایی که توسط DEFLATE برای بهبود نسبت فشرده سازی استفاده نشده است، کار آینده است.

برای افزایش نسبت فشرده سازی می توان پیشرفت هایی در ACM انجام داد. ACM می تواند فشرده سازی کاراکترهای غیر ASCII را بهبود بخشد، زیرا تکنیک فعلی بهینه نیست. افزودن قابلیت ایجاد مدل های آموزش دیده ممکن است نسبت فشرده

سازی آن را نیز به میزان قابل توجهی افزایش دهد. با این پیشرفت ها، ممکن است ACM در صورت استفاده از پیش پردازنده، نسبت فشرده سازی برنامه های فشرده سازی عمومی را افزایش دهد.
همچنین می توان افزایش تأخیر ACM را بهبود داد. کد را می توان در C بازنویسی کرد و بهینه سازی را برای فرآیند فشرده سازی و خروج از فشرده سازی انجام داد. اگر ACM بیشتر بر زمان تأخیر تمرکز داشت تا نسبت فشرده سازی، استفاده از ACM در شرایطی که سرعت در اولویت بیشتری نسبت به ضریب فشرده سازی است، مفید خواهد بود. به عنوان مثال، می توان آن را در ترکیب با کمپرسور اسنپی مبتنی بر LZ77 در زمینه و پس زمینه سرور استفاده کرد. Snappy نسبت فشرده سازی را از نظر سرعت به خطر می اندازد و در فایلهای متنی نسبت به فایلهای باینری به نسبت بهتری دست می یابد. استفاده از ACM به عنوان پیش پردازنده Snappy می تواند نسبت فشرده سازی را افزایش دهد.

## ۱۰. مراجع


[1] K. Sayood, Introduction to data compression, 3rd ed. Amsterdam ; Boston: Elsevier, 2006.

[2] K. K. Shukla and M. V. Prasad, Lossy image compression: domain decomposition-based algorithms. London ; New York: Springer, 2011.

[3] S. Haldar-Iversen, «Improving the text compression ratio for ASCII tex», (Unpublished master's thesis). University of Norway, Nov. 2020

[4] «Data Compression Conference - Home». https://www.cs.brandeis.edu/~dcc/index.html (opened Sep. 13th, 2020).

[5] D. Huffman, «A Method for the Construction of Minimum-Redundancy Codes», Proc. IRE, vol. 40, nr. 9, s. 1098–1101, sep. 1952, doi: 10.1109/JRPROC.1952.273898.

[6] J. Ziv and A. Lempel, «A universal algorithm for sequential data compression», IEEE Trans. Inform. Theory, vol. 23, nr. 3, pp. 337–343, May 1977, doi: 10.1109/TIT.1977.1055714.

[7] A. Farina, G. Navarro, and J. R. Parama, "Boosting Text Compression with Word-Based Statistical Encoding", The Computer Journal, vol. 55, no. 1, pp. 111–131, Jan. 2012, doi: 10.1093/comjnl/bxr096.

[8] «Shoco - a fast compressor for short strings». https://ed-vonschleck.github.io/shoco/ (opened Sep. 14th, 2020).

[9] T. Summers, "Hardware based GZIP compression, benefits and applications," CORPUS, vol. 3, pp. 2–68, 2008.

[10] D. Belanger, K. Church, and A. Hume, "Virtual Data Warehousing, Data Publishing and Call Detail," in Databases in Telecommunications, vol. 1819, W. Jonker, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 106–117.