دانشکده مهندسی

گروه برق

پایان نامه کارشناسی

گرایش : مخابرات

عنوان :

توربو کد

استاد راهنما :

دکتر مصطفوی

نگارش :

ایمان نصرت (87442190)

مرداد 1392

فهرست

مقدمه

معرفی

فکر استفاده از کد های به هم پیوسته اولین بار توسط Forney به عنوان روشی برای دستیابی به بهره ی کدینگ بالا با ترکیب دو یا تعداد بیشتری بلوک نسبتاً ساده یا کدهای اصلی (کدهای سازنده) پیشنهاد شد . کدهای حاصل قابلیت تصحیح کدهای بیشتری را دارد و ساختار آنها اجازه رمزگشایی از کدینگ های نسبتاً ساده تا پیچیده را می دهد. یک اتصال سری از این کدها اغلب در سیستم های توان پایین مانند فرستنده های فضاپیماها استفاده می شود. محبوب ترین نوع این روش کد خروجی Reed-Solomon (استفاده از اولی – حذف آخری) که از کدهای درونی کانولوشنالی (استفاده از آخری – حذف اولی) پیروی و تبعیت می کند ، است. توربو کد را می توان همچون پالایشی از ساختار رمزگذاری به هم پیوسته بعلاوه الگوریتمی تکراری برای رمزگشایی دنباله ی کدهای به هم پیوسته تصور کرد.

توربو کد برای اولین بار در سال 1933 توسط Berrou ، Glavieux و Thitimajshima معرفی شد. که دراین روش به ازای هر بیت به احتمال خطای 10^{-5} با نرخ $1/2$ ، با انتقال روی یک کانال همراه با نویز سفید گوسی افزایشی (AWGN) و مدولاسیون BPSK در $E_b/N_0 = 0.7\ dB$ ، دست پیدا می کنیم. کدها با استفاده از دو یا تعداد بیشتری کد اصلی با جابجایی کدهای دنباله ای از همان اطلاعات ایجاد می شود. در حقیقت ، برای کدهای کانولوشن ، گام نهایی در دیکدر نتیجه می دهد بیت های رمزگشایی شده که از سخت تصمیمی (یا عموماً سمبل های رمزگشایی) حاصل شده است ، برای اجرای طرح هایی چون توربو کدها برای آنکه عملکردی صحیحی داشته باشند ، الگوریتم رمزگشایی نباید خود را محدود به عبور سخت تصمیمی از دیکدر کنیم. برای بهره بیشتر و دریافت بهتر اطلاعات بدست آمده از هر دیکدر ، الگوریتم رمزگشایی باید از نرم تصمیمی نسبت به سخت تصمیمی بیشتر تاثیر گرفته و استفاده کند. برای یک سیستم با دو کد اصلی و سازنده ، مفهوم توربو کد یعنی عبور نرم تصمیمی از خروجی یک دیکدر به عنوان ورودی دیکدر بعدی ، و تکرار این عملیات برای چندین بار به منظور دستیابی به یک تصمیم مطمئن تر است.

کدینگ کانال

کدهای تصحیح خطا (FEC) حاصل از خطای کانال انتقال ، عموماً به منظور صرفه جویی و تامین انرژی مورد نیاز در سیستم های بی سیم ، مورد استفاده قرار می گیرند. در سمت فرستنده ، یک انکدر ، دیتا بیت های اضافی را به شکل زوج های اطلاعات به اطلاعات ارسالی اضافه می کند. سپس در سمت دریافت کننده ، یک دیکدر با قابلیت استخراج این اطلاعات اضافی قرار دارد ، که به این ترتیب قابلیت تصحیح خطای کانال را به ما می دهد.

یک انکدر باینری با قابلیت تصحیح خطا ، k بیت را در لحظه دریافت میکند و n بیت خروجی را تولید می کند ، که در آن n>k است. در حالی که از n بیت ، می توان 2^n دنباله تشکیل داد ، تنها 2^k از این حالات صحیح هستند. کسر k/n را نرخ بیت نامیده و آن را با r نمایش می دهند.

نرخ بیت های پایین ، که با حرف کوچک r مشخص شدند ، عموماً می توانند خطاهای بیشتری را نسبت به نرخ های بالاتر تصحیح کنند و به همین جهت توان بیشتری را نیاز دارند. به هر صورت ، کدهایی با نرخ های بالا نیز پهنای باند بیشتری از کدهای با نرخ پایین نیاز دارند. از این رو انتخاب نرخ بیت ، نیازمند مصالحه ای مابین توان و پهنای باند است.

برای هر ترکیب از کدها ، نرخ کد (r) ، طول کلمه رمز (n) ، نوع مدولاسیون ، نوع کانال و توان نویز دریافتی ، مواردی هستند که میزان توان مصرفی را برای یک بیت حامل اطلاعات ، محدود می کنند. این محدودیت را ظرفیت کانال یا همان ظرفیت شانون می نامند. که Claude Shannon در سال 1948 پس از به کارگیری روابط ریاضی آن را به دست آورد. پس از آن ، مهندسین و ریاضیدانان سعی کردند کدهایی با بهره بالا تولید کنند که به ظرفیت شانون نزدیک باشد. اگرچه هر کد جدید با قابلیت تصحیح خطا ، درجه ای نسبت به کد قبلی خود به ظرفیت شانون نزدیک است ، اما همچنان مانند سال 1990 ، فاصله ی بین تئوری و عمل برای مدولاسیون باینری که از کانال گوسی سفید استفاده می کنند ، در حدود 3dB است. به بیان دیگر ، کدهای عملی که در سیستم های موبایل ، ماهواره و ... مورد استفاده قرار می گیرند به توان دو برابر بیشتر از آنچه که در تئوری بدست آمده است ، نیازمند هستند. برای کانال های فیدینگ که ناگوارتر از کانال های گوسی سفید هستند ، این شکاف و فاصله بیشتر از این مقدار نیز می باشد.

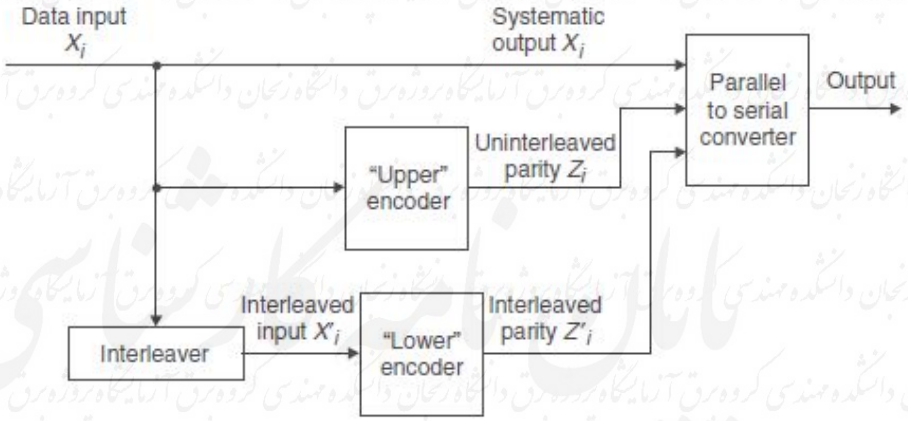
ظهور و شروع توربو کدها

پیشرفت اصلی و ابتدایی در تئوری کدینگ در سال 1993 اتفاق افتاد ، زمانی که گروهی از محققان در فرانسه توربو کدها را کشف کردند. نتایج ابتدایی نشان داد که توربو کدها می توانند به بازدهی با نیمی از ظرفیت شانون دست یابد. این امر فوق العاده ، ابتدا منجر به ایجاد شک و تردید شد. اما بار دیگر محققان صحت این امر را جداگانه بررسی کردند ، و به زودی تحقیقات وسیعی با هدف توضیح ، این افزایش عملکرد قابل توجه در توربو کدها شکل گرفت ، که تاکنون نیز ادامه دارد. بیشتر این تحقیقات برروی بهبود و پیشرفت عملکرد توربو کدها متمرکز شده اند ، که ما قسمت کوچکی را مورد بحث قرار می دهیم.

با پایان سال های 1990 ، خواص و ویژگی های توربو کدها به خوبی شناخته شد ، و شروع به بکارگیری از آن در سیستم های مختلف شد. و اکنون تبدیل به استانداردی برای ارتباط با فضاپیماها در شرکت NASA ، ارتباط تصویری دیجیتال و همچنین نسل سوم ارتباطات شده است.

انکدرهای با اتصال موازی

یکی دیگر از ویژگی های جالب توربو کدها این است که آن ها یک کدینگ تکی و تنها نیستند. در حقیقت آن ها ، ترکیبی از دو یا تعداد بیشتری از کدها است که با مساعدت یکدیگر کار می کنند ، که با استفاده یکی از آن ها به تنهایی نمی توان به کارایی مورد نظر رسید. در یک مورد بخصوص ، یک توربو کد از اتصال موازی دو کد اصلی تشکیل شده است که با یک لایه جابجا کننده از هم جدا شده اند. هر کدام از کدهای اصلی و سازنده ممکن است یکی از انواع مرسوم کدهای تصحیح کننده خطا در ایجاد ارتباطات باشند. اگرچه این دو انکدر اصلی ممکن است با هم متفاوت باشند ، اما در عمل معمولاً مشابه یکدیگرند. یک ساختار معمول و عمومی برای توربو کدها در شکل 1 نشان داده شده اند. همان طور که می توان دید ، توربو کد از دو انکدر اصلی مشابه تشکیل شده است. دیتای ورودی و زوج های خروجی از دو انکدر موازی پس از متوالی شدن و تبدیل به یک کلمه توربو کد ارسال می شوند.



شکل 1 – یک انکدر معمول در توربو کد

یک جابجا کننده بیت ها ، جز و شاخصه اصلی سازنده توربو کد است. در یک روش ساده آن ، نظم بیت های اطلاعات را به می ریزد. اگرچه همان دسته از بیت های اطلاعات ورودی در خروجی این جابجا کننده ظاهر می شوند اما ترتیب این بیت ها تغییر کرده است. این بهم رختگی برای بیت های بعد نیز همانند بیت های قبلی انجام می شود. بدون وجود این جابجا کننده ، دو انکدر اصلی که همان بیت های ورودی را دریافت می کنند ، خروجی های یکسان خواهند داشت. به هر حال با استفاده از جابجا کننده ، دیتا $\{X_i\}$ طوری جابجا می شود که انکدر دوم دیتای متفاوت یعنی $\{X_i'\}$ را دریافت کند. به این ترتیب تقریباً می توان اطمینان داشت که خروجی انکدر دوم با خروجی انکدر اول فرق دارد ، به جز موارد معدود که دیتا پس از عبور از جابجا کننده دقیقاً همان دیتا ورودی باشد. توجه شود که جابجا کننده ای که در توربو کدها مورد استفاده قرار می گیرد ، با جابجاکننده های مستطیلی که عموماً در سیستم های بی سیم برای از بین بردن محوشدگی استفاده می شود ، متفاوت است. در حالی که یک جابجا کننده مستطیلی سعی در فاصله دادن بین اطلاعات خروجی براساس یک الگوی خاص دارد ، جابجا کننده در توربو کدها سعی در جابجایی تصادفی نظم اطلاعات ، براساس یک روش بی قاعده را دارد.

چرا توربو کدها عملکرد مناسبی دارند؟

برای اینکه متوجه شویم که چرا عملکرد توربو کدها خوب است ، باید بفهمیم اولین قدم برای خوب بودن یک کد در چیست. اما ابتدا ، دو عبارت باید تعریف شود. یک کد خطی ، کدی است که از جمع باینری دو حالت ممکن (با جمع باینری هر بیت با بیت متناظر خود) بدست می آید. بیشتر کدها ، شامل توربو کدها ، کدی خطی هستند. وزن همینگ (برای راحتی به عنوان وزن نیز استفاده می شود) یک حالت ، که تعداد یک هایی است که در آن حالت وجود دارد. بنابراین یک کد خطی می توانست از دو حالت ممکن تشکیل شود ، {000} و {111} ، که وزن همینگ در حالت اول 0 و در حالت دوم 3 است.

یک کد خطی خوب ، کدی است که بیشترین وزن حالت ها را داشته باشد. بیشترین وزن حالات ایده آل هستند زیرا به این معنی است که آن ها ممتاز هستند ، و بنابراین دیکدر راحت تر آن ها را تشخیص می دهد. درحالی که احتمال وقوع یک حالت با وزن کم باید پایین بیاید. یک راه برای کم کردن تعداد حالات با وزن کم استفاده از توربو انکدر است. از آنجا که وزن حالات توربو کد به سادگی از جمع ورودی و دو زوج خروجی از حالات اصلی بدست می آید ، یکی از این زوج های خروجی می تواند وزن کم داشته باشد (به شرط آنکه حالات دیگر وزن بالا داشته باشد). زیرا ورودی انکدر دوم توسط جابجا کننده تغییر کرده ، خروجی آن نیز معمولاً با خروجی انکدر اول کاملاً متفاوت است. بنابراین اگرچه ممکن است یکی از دو خروجی ها دارای وزن کم باشد ، اما احتمال کم بودن وزن هردو خروجی انکدرها به صورت همزمان خیلی پایین است. این بهبود ایجاد شده در عملکرد را بهره ی جابجا کننده می نامند و این یکی از دلایل اصلی عملکرد خوب توربو کدها است.

فصل اول

کدهای کانولوشن

کدهای کانولوشن معمولا با سه پارامتر مشخص می شود : (n,k,m).

n : تعدادبیت های خروجی

k: تعداد بیت های ورودی

m: تعداد حافظه های رجیستری

مقدار و اندازه k/n را نرخ کد می نامند ، که معیاری برای تاثیرگذاری کد است. معمولا k و n دارای مقادیر 1 تا 8 ، و m از 2 تا 10 و نرخ کد از 1/8 تا 7/8 است به جز کاربردهای در فضاپیماها که در آنها نرخ کد تا 1/00 پایین می آید.

اغلب کارخانه هایی که چیپ های کدهای کانولوشن را تولید می کنند آن را با پارامترهای (n,k,L) مشخص می کنند ، که مقدار L محدودیت طول کد است که با رابطه زیر تعریف می شود :

Constraint Length: L = k (m-1)

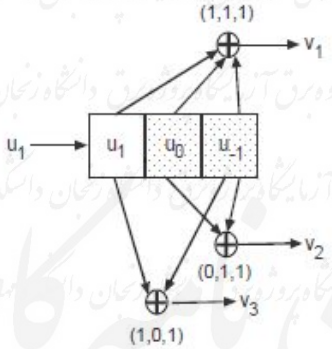
طول محدود L معرف تعداد بیت هایی در حافظه انکدر است که در بیت های خروجی n تولیدی تاثیرگذار است.

پارامترهای کد و ساختاری از کدهای کانولوشن

ساختار و ساختمان کدهای کانولوشن به راحتی از روی پارامترهای آن قابل رسم است. ابتدا m باکس را می کشیم که معرف m حافظه رجیستری است. سپس n جمع کننده باینری را رسم می کنیم که معرف n بیت خروجی می باشد. سپس حافظه رجیسترها را توسط جملات تولید کننده کدها به جمع کننده ها وصل می کنیم ، همان طور که در شکل 2 نشان داده شده است.

دانشجویان محترم:

جهت دسترسی به متن کامل پایان نامه‌ها به کتابخانه دانشکده مهندسی و یا آزمایشگاه پروژه گروه برق مراجعه فرمایید.



شکل 2 – یک کد کانولوشن (3,1,3) که دارای 3 حافظه ، 1 ورودی و 3 خروجی است

این مثال دارای نرخ کد 1/3 است. هر بیت ورودی به سه بیت خروجی کد می شوند. محدودیت طول نیز برابر 2 است. 3 بیت خروجی نیز از طریق سه جمع کننده باینری که بیت های حافظه رجیسترهای مخصوص خود را جمع می کنند ، تولید می شوند. انتخاب این رجیسترها که کدام می توانند در جمع شرکت داشته باشند به چندجمله ای تولید کننده برای بیت های خروجی بستگی دارد. برای مثال ، اولین بیت خروجی با چند جمله ای (1,1,1) تولید می شود. بیت دوم خروجی با چندجمله ای (0,1,1) و بیت سوم خروجی با چند جمله ای (1,0,1) تولید می شود. بیت های خروجی تنها با جمع این بیت ها بدست می آید.

انتخاب چند جمله ای

تعداد بسیاری گزینه برای انتخاب چند جمله ای با m رجیستر وجود دارد. همه این چند جمله ای ها منجر به یک دنباله خروجی که ویژگی بروز و محافظت در برابر خطا را داشته باشند ، نمی شود. در کتاب Petersen و Weldon یک لیست کامل از این چند جمله ای ها آمده است. چند جمله ای های مناسب معمولا با شبیه سازی کامپیوتر بدست می آید. یک لیست مناسب از این چند جمله ای ها با نرخ 1/2 در جدول زیر آمده است.

نتیجه گیری :

این مقاله مفهوم توربو کدینگ را توصیف کرد ، که موقعیت اساسی و پایه آن به نحوه اتصال دو یا تعداد بیشتری از عناصر اصلی و سازنده ی کدها بستگی دارد. ابزارهای اندازه گیری آماری همچون احتمال گذشته و تابع احتمال را مرور و دوره کردیم ، و از این ابزارها برای توصیف و توضیح عملکرد تصحیح خطای ورودی نرم/خروجی نرم در دیکدر استفاده کردیم. نشان دادیم که عملکرد چگونه بهبود پیدا می کند زمانی که از خروجی نرم دیکدرهای اتصالی برای یک پردازش رمزگشایی تکراری استفاده کردیم. سپس سعی کردیم که این مفهوم ها و تعاریف را برای اتصال موازی کدهای سیستماتیک کانولوشن *(RSC)* به کار بگیریم ، و سپس توضیح دادیم چرا کدهای اینچنینی را برای ساخت بلوک ها در توربو کد ترجیح می دهیم. یک رمزگشای فیدبکی را توصیف کرده و عملکرد قابل توجه آن را معرفی کردیم.

منابع :

1. Matthew C. Valenti and Jian Sun , “Turbo Code”
2. Sklar , Bernard , “Fundamentals of Turbo Codes”
3. Langto , Charan , “Coding and decoding with Convolutional Codes”