Hardness amplification and error correcting codes

حمید پوررربیع رودسری ۸۹۱۰۴۴۵۹

تا کنون همواره سعی در کاهش پیچیدگی محاسباتی داشتیم اما در این بخش هدف سخت کردن محاسبه بعضی از توابع است. چرا که با استفاده از این توابع است که می‌توانیم به اهداف رمزنگاری دسترسی پیدا کنیم. در این مطلب با استفاده از لم یائو روشی را معرفی می‌کنیم که با استفاده از آن می‌توانیم توابع نسبتا سخت در حالت میانگین را به توابع بسیار سخت در حال میانگین تبدیل کنیم. پس از آن با استفاده از کدهای تصحیح شونده روشی را معرفی می‌کنیم که در آن توابع سخت در بدترین حالت را به توابع نسبتا سخت در حالت میانگین تبدیل کنیم با استفاده از این روش و لم یائو می‌توانیم از توابع سخت در بدترین حالت توابع بسیار سخت در بدترین حالت به دست آوریم. البته با روش فوق سایز مدارهای لازم برای حل کردن توابع به دست آمده نسبت به توابع اصلی کمتر می‌شود که برای ما مطلوب نیست. به همین دلیل با استفاده از تکنیکی به نام لیست دیکودینگ به توابع بسیار سخت در حالت میانگین می‌رسیم بدون اینکه سایز مدار لازم برای محاسبه آن‌ها را افزایش دهیم.

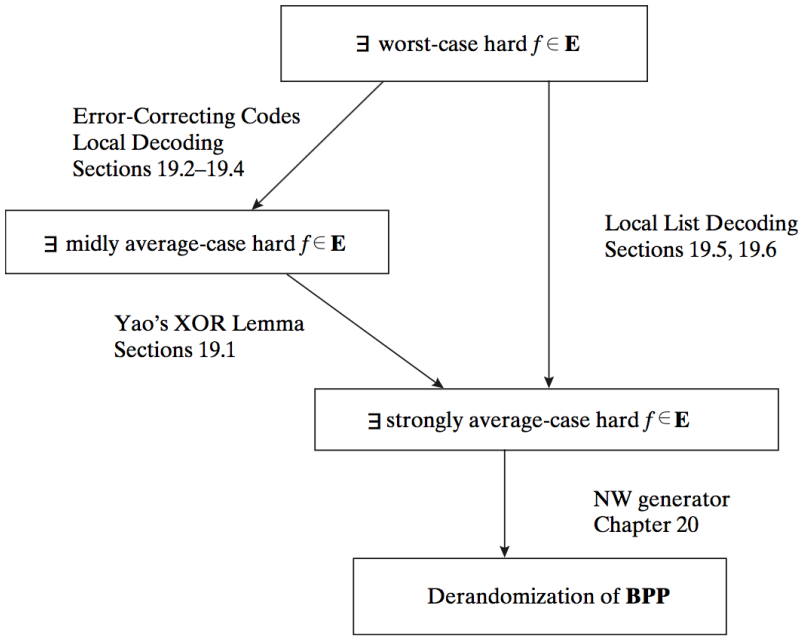


Figure 19.1. Organization of Chapter 19.

لم یائو

همانطور که پیش از این نیز گفته شده این لم روشی است که با استفاده از آن می‌توان از توابع نسبتا سخت در حالت میانگین به توابع بسیار سخت در حالت میانگین رسید. اما پیش از گفتن این لم لازم است که سختی در حالت میانگین را تعریف کنیم.

تعریف: (سختی در بدترین حالت و سختی در حالت میانگین) برای تابع $f : \{0,1\}^n \to \{0,1\}$ و $\rho \in [0,1]$ مقدار سختی حالت میانگین ρ برای f را به صورت $H^{\rho}_{avg}(f)$ نمایش می‌دهیم و مقدار آن برابر است با بیشترین مقدار S به صورتی که برای هر مدار C با سایز حداکثر S داشته باشیم $\Pr_{x \in_R \{0,1\}^n}[C(x) = f(x)] < \rho$.

حال مقدار سختی در حالت بیشینه را به صورت H_{wrs} (f) نمایش می‌دهیم و مقدار آن را برابر H^1_{avg} (f) تعریف می‌کنیم.
همچنین سختی در حالت میانگین را به صورت H_{avg}(f) نمایش می‌دهیم و مقدار آن را ماکزیمم $\{S : H^{1/2+1/S}(f) \geq S\}$ قرار می‌دهیم.

با استفاده از تعریف فوق حال می‌توانیم لم یائو را بیان کنیم:

قضیه: (لم یائو) برای هر تابع به صورت $f:\{0,1\}^n \rightarrow \{0,1\}$, $\delta>0$ و $k\in N$ اگر $\epsilon>2(1-\delta)^k$ آنگاه:

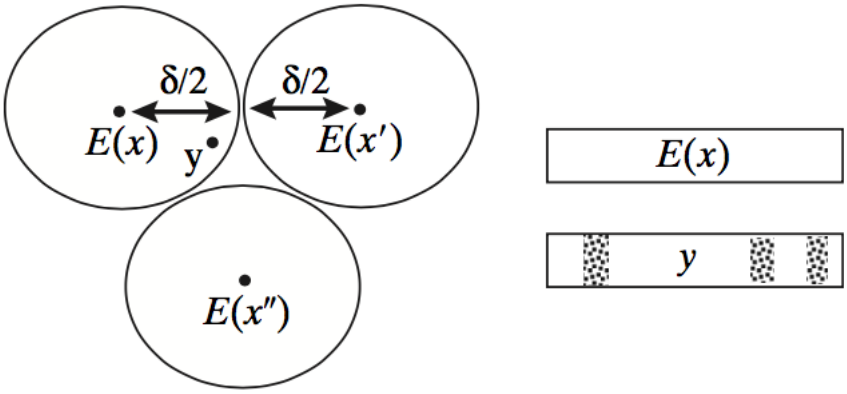
$$H^{1/2+\epsilon}_{avg}(f^{\oplus k}) \geq \frac{\epsilon^2}{400n} H^{1-\delta}_{avg}(f)$$

و مقدار $f^{\oplus k} : \{0,1\}^{nk} \rightarrow \{0,1\}$ به صورت $f^{\oplus k}(x_1,\ldots,x_k) = \sum_{i=1}^{k} f(x_i) \pmod 2$ تعریف شده است.

برای اثبات قضیه‌ی فوق از لمی استفاده می‌کنیم که بیان می‌کند تمامی توابع سخت به ازای بخشی از ورودی‌ها خیلی سخت هستند و نام این مجموعه را هسته سخت می‌نامیم.

کدهای تصحیح شونده:

کدهای تصحیح شونده به این صورت هستند که رشته‌ها را با تاکید بر تفاوت‌های آنها به یک گروه بزرگتر از رشته‌ها می‌نگارند که به ازای دو رشته فاصله‌ی رشته‌های به دست آمده از هم بیشتر شده باشد.



تعریف: فاصله همینگ بین x و y به صورت $\Delta(x,y)$ نمایش داده می‌شود و برابر $\frac{1}{m}|\{i : x_i \neq y_i\}|$ است.

تابع به صورت $E : \{0,1\} \rightarrow \{0,1\}^n$ یک کد تصحیح شونده یا ECC با فاصله δ است اگر به ازای هر $x\neq y \in \{0,1\}$ داشته باشیم $\Delta(E(x), E(y)) \geq \delta$

هدف اصلی کدهای تصحیح شونده از ارسال پیام بر روی کانال‌های با نویز به وجود آمده است. چرای که در صورت که مثلا پس از ارسال یک کد بر روی کانال با نویز 0.1 اگر از کد تصحیح شونده با فاصله 0.2 یا بیشتر استفاده شده باشد به راحتی می‌توان پیام را بازیابی کرد.

لم: به ازای هر δ با فاصله حداکثر ۰٫۵ تابعی تصحیح شونده $E : \{0,1\}^n \to \{0,1\}^{n/(1-H(\delta))}$ با فاصله δ وجود دارد و داریم $H(\delta) = \delta \log(1/\delta) + (1-\delta)\log(1/(1-\delta))$

در لم بالا ثابت می‌شود که برای δ بیش از ۰٫۵ کد تصحیح شونده‌ای وجود ندارد و به ازای δ مساوی ۰٫۵ نیز سایز کد بسیار بزرگ می‌شود.

کد والش هادامارد:

این کد برای رشته x مقدار $x \odot y = \sum_{i=1}^{n} x_i y_i \pmod 2$ را به ازای تمامی رشته‌های y با طول n حساب کرده و کنار هم می‌گذارد و یک رشته به طول دو به توان n تولید می‌کند.

می‌توان به سادگی ثابت کرد که کد والش هادامارد یک کد با فاصله ۰٫۵ است. اما مشکل این کدینگ طول زیاد رشته‌های تولید شده است.

کد رد سولومن:

برای این کدینگ به ازای الفبای صفر و یک از الفبای بزرگتری استفاده می‌کنیم. قبل از گفتن کد باید به این نکته اشاره کنیم که تفاوت فاصله در تمامی الفباها به صورت بیان شده در قبل است.

به بیان ساده این کدینگ بر روی یک رشته به طول n به این صورت عمل می‌کند که یک چند جمله‌ای از درجه $n-1$ می‌سازد که المان‌های رشته ضرایب آن هستند و مقدار آن را بر روی m نقطه‌ی از قبل معلوم شده حساب می‌کند و به عنوان رشته کد شده بر می‌گرداند.

به سادگی می‌توان ثابت کرد که کدینگ رد سولومن یک کدینگ تصحیح شونده با فاصله $1-n/m$ است.

کد رد مولر:

این کدینگ حالت کلی‌تر کدینگ‌های گفته شده در بالا است.

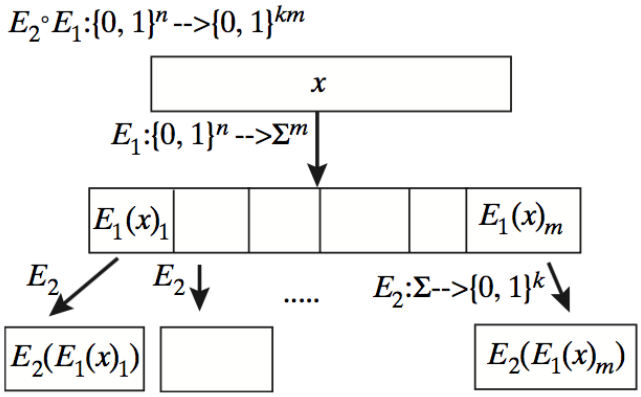
فیلد F و دو عدد l و d را به صورتی که $d < |F|$ در نظر بگیرید. کدینگ رد مولر یک تابع به صورت زیر است.

$$P(x_1, \ldots, x_\ell) = \sum_{i_1 + \cdots + i_\ell \le d} c_{i_1,\ldots,i_\ell} x_1^{i_1} x_2^{i_2} \cdots x_\ell^{i_\ell}$$

کدهای چسبیده:

برای اینکه به صورت همزمان از فواید کد والش هادامورد و رد سولومن استفاده کنیم اینگونه عمل می‌کنیم که ابتدا با استفاده از کد رد سولومن رشته را کد می‌کنیم و سپس با استفاده از کد والش هادامورد الفبای کد رد سولومن را به فضای صفر و یک می‌بریم به این صورت یک کد در فضای صفر و یک و با طول نه چندان زیاد خواهیم داشت.

به سادگی می‌توان ثابت کرد که کد به دست آمده فاصله‌ای برابر حاصل ضرب کدهای رد سولومن و والش هادامورد دارد.

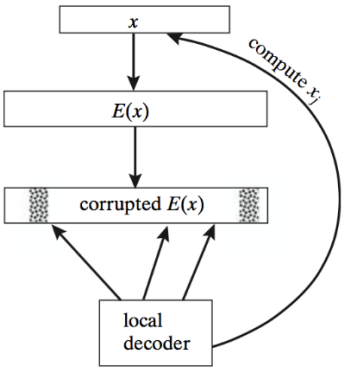


دیکود کردن کدهای تصحیح شده:

کدهای تصحیح شده تا زمانی مفید هستند که بتوانیم به صورت بهینه رشته‌ی کد شده را به رشته‌ی اولیه تبدیل کنیم. برای تمامی کدهای گفته شده در بالا الگوریتم‌های بهینه‌ای برای دیکود کردن رشته‌ی کد شده وجود دارد.

دیکود کردن منطقه‌ای:

یک تابع به صورت $f : \{0,1\}^n \rightarrow \{0,1\}$ را می‌توان به صورت یک رشته به طول $N = 2^n$ در نظر گرفت. به این صورت که تمامی خروجی‌های آن را پشت سر هم قرار می‌دهیم. حال اگر با استفاده از یک کد تصحیح شونده با فاصله $2*\delta$ ببریم آنگاه اگر تابع کد شده را با دقت δ محاسبه کنیم با استفاده از ویژگی‌های کد تصحیح شونده می‌توانیم تابع اصلی را به صورت کامل حساب کنیم. پس اگر تابع اولیه سخت در بدترین حالت بود تابع جدید سخت در حالت میانگین است!



با توجه به توضیح بالا لازم است که بتوانیم از روی رشته‌ی کد شده به طول نمایی مقدار حرف i ام رشته‌ی اصلی را در زمان چند جمله‌ای و با دسترسی‌های محدود به خانه‌های رشته‌ی اصلی به دست آوریم. به این عمل دیکود کردن منطقه‌ای گفته می‌شود.

به سادگی می‌توان نشان داد که برای تمامی کدهای گفته شده در بالا یک دیکودر منطقیه‌ای بهینه وجود دارد.

حال با توجه به نکته‌ی فوق می‌توان به سادگی از یک تابع سخت در بدترین حالت به یک تابع نسبتا سخت در حالت میانگین رسید و سپس با استفاده از لم یائو می‌توانیم تابع نسبتا سخت در حالت میانگین را به تابع سخت در حالت میانگین تبدیل کرد.

دیکود کردن لیست محلی

یکی از مشکلات روش گفته شده این است که سایز مدار لازم برای حل کردن تابع نهایی کمتر از سایز مدار اولیه خواهد بود. و این موضوع به خاطر لم یائو است. پس سعی می‌کنیم که بدون استفاده از لم یائو بتوانیم توابع قویا سخت در حالت میانگین بسازیم. برای اینکار باید بتوانیم با دقت کمتر از نصف فاصله کدینگ هم توابع به دست آمده را حساب کنیم و با استفاده از آن به توابع اصلی برسیم. اما در کدینگ مشکلی که وجود دارد اینجاست که اگر فاصله بیشتر از نصف فاصله‌ی کدینگ شود دیگر تنها یک جواب وجود ندارد. اما ثابت می‌شود که تعداد این جواب‌ها محدود است. پس با توجه به این نکته می‌توان توابع سخت در حالت میانگین ایجاد کرد.

برای ایجاد این توابع باید بتوانیم به صورت بهینه لیست محلی را پیدا کنیم. می‌توان به سادگی نشان داد که تمامی کدهای بالا به صورت بهینه دیکود محلی می‌شوند.