ششمین کنفرانس بین المللی انجمن رمز ایران
۱۵ و ۱۶ مهر ماه ۱۳۸۸، دانشگاه اصفهان
انجمن رمز ایران
دانشگاه اصفهان

کشف رمزهای بلوکی جایگشتی با استفاده از الگوریتم رقابت استعماری

بهاره قادری[۱]، کارو لوکس[۲]

دانشگاه تهران، دانشکده مهندسی برق و کامپیوتر، آزمایشگاه رباتیک و هوش مصنوعی[۱]
ba.ghaderi@gmail.com
دانشگاه تهران، دانشکده مهندسی برق و کامپیوتر، قطب کنترل و پردازش هوشمند[۲]
lucas@ut.ac.ir

چکیده

مسئله کشف رمز، مسئله جستجو برای یافتن کلید صحیح، از میان همه کلیدهای ممکن است. در این مسئله، اندازهٔ فضای جستجو با افزایش طول کلید به شدت زیاد می‌شود. بنابراین روشهای کلاسیک جستجو در زمان معقول به جواب نمی‌رسند. استفاده از الگوریتم‌های بهینه‌سازی هوشمند در فضاهای جستجوی بسیار بزرگ، همواره موفقیت‌آمیز بوده است. لذا استفاده از این الگوریتم‌ها روش مناسبی برای مسئله کشف رمز است. در این مقاله، روشی برای استفاده از الگوریتم رقابت استعماری برای یافتن کلید رمز، ارائه شده است. در اینجا رمز بلوکی جایگشتی را بررسی می‌کنیم که کلید آن یک جایگشت از اعداد طبیعی است.

نتایج به دست آمده از شبیه‌سازی نشان می‌دهد که با استفاده از روش پیشنهادی این مقاله، همواره طول کلید را با دقت می‌توان به‌دست آورد. در صورتی که طول متن رمز به قدر کافی طولانی باشد، با دقت بالای ۸۰ درصد، می‌توان به کلید رمز دست پیدا کرد. بطور مثال اگر طول کلید ۱۰ رقم باشد، با این روش در بدترین حالت، ۸ رقم را بدرستی می‌یابد. همچنین در این مقاله یک روش ساده برای محاسبه قدرت نسبی امپریالیست‌ها در الگوریتم رقابت استعماری پیشنهاد شده است که نسبت به روش اولیه، محاسبات کمتری دارد و سریعتر اجرا می‌شود.

کلمات کلیدی

الگوریتم رقابت استعماری، رمزنگاری، کشف رمز[۱]، رمز بلوکی جایگشتی.

۱- مقدمه

امروزه در رمزنگاری مخفی نگاه داشتن الگوریتم منسوخ شده است. الگوریتم رمزنگاری برای همگان آشکار است و آنچه که مخفی نگاه داشته می‌شود کلید رمز است. بنابراین در کشف رمز، مسئله یافتن کلید رمز است که شامل یافتن طول کلید و مقدار آن می‌باشد. یعنی در واقع، مسئله کشف رمز، یک مسئله جستجو برای یافتن کلید صحیح، از میان همه کلیدهای ممکن است. در این مسئله اندازهٔ فضای جستجو با افزایش طول کلید به شدت زیاد می‌شود. بنابراین روشهای کلاسیک جستجو در زمان معقول به جواب نمی‌رسند. و از آنجا که در حال حاضر در بیشتر کاربردهای رمزنگاری، بخاطر مقابله با خطر افشای کلید، آنرا برای مدت طولانی ثابت نگاه نمی‌دارند، مسئله زمان در جستجوی کلید بسیار اهمیت دارد.

الگوریتم‌های بهینه‌سازی هوشمند در فضاهای جستجوی بسیار بزرگ موفقیت خوبی از خود نشان داده‌اند. لذا استفاده از این الگوریتم‌ها روش مناسبی برای مسئله کشف رمز است. پیش از این نیز کارهایی در این زمینه صورت گرفته است. A.Gorodilov و V.Morozenk[۲] برای یافتن کلید رمز بلوکی جایگشتی، در متون رمزی روسی، از الگوریتم ژنتیک استفاده کرده‌اند. همچنین R.Morelli و R.Walde [۴] الگوریتم ژنتیک را برای کشف رمزهای کوتاه، مثلاً حدود ۱۰۰ کاراکتر، بکار برده‌اند. ابتدا بر پایه روشی که A.Gorodilov و V.Morozenk در [۲] ارائه

نموده‌اند، در استفاده از تکامل برای یافتن جواب، ICA[1] را جانشین GA[2] نمودیم. نتایج شبیه‌سازی چندان قابل قبول نبود؛ مدت زمان مورد نیاز برای یافتن کلید بسیار زیاد بود. علاوه بر این، وقتی طول کلید افزایش پیدا می‌کرد، دقت کاهش می‌یافت. مثلاً کلید با طول ۱۰ رقم را با دقت متوسط ۵۰ درصد حدس می‌زد. سپس با تغییر تابع cost، و نیز با بکاربردن یک روش پیشنهادی، برای محاسبه قدرت نسبی در ICA، به نتایج قابل قبولی دست پیدا کردیم. دقت الگوریتم برای کلیدهای با طول زیاد در حد مطلوبی افزایش پیدا کرد و زمان مورد نیاز برای یافتن کلید نیز کاهش چشمگیری داشت.

در ادامه، ابتدا بطور مختصر به معرفی رمزهای بلوکی جایگشتی ساده می‌پردازیم. سپس یک شمای کلی از ICA ارائه می‌کنیم و شرح جزئیات مراحل مختلف این الگوریتم را به بخش‌های آتی موکول می‌کنیم. در بخشهای بعدی، جزئیات کشف کلید و نحوه انجام مراحل مختلف الگوریتم را، به تفصیل بیان می‌کنیم. در انتها نتایج شبیه‌سازی این روش و ایده‌هایی برای کارهای آتی روی آن، آورده شده است.

۲- رمز بلوکی جایگشتی[3] [۲]

یک رمز بلوکی جایگشتی به این صورت است که متن اصلی را به بلوک‌هایی به طول ثابت N تقسیم می‌کند، سپس کاراکترهای هر کدام از این بلوک‌ها را براساس یک جایگشت ثابت P جابجا می‌کند. P را بصورت زیر نمایش می‌دهیم.

$$P = \begin{pmatrix} 1 & 2 & 3 & \dots & i & \dots & N \\ p_1 & p_2 & p_3 & \dots & p_i & \dots & p_N \end{pmatrix} \quad (۱)$$

P_i ها اعداد طبیعی از ۱ تا N هستند که مکانی که کاراکتر iام باید به انجا منتقل شود را مشخص می‌کند. در این رمز، $P_1, \dots, P_N$- کلید مخفی به طول ثابت N است که هر بلوک و در نهایت همه متن را رمز می‌کند. در این مقاله، به یافتن طول N و سپس کلید $P_1, \dots, P_N$ می‌پردازیم.

۳- الگوریتم رقابت استعماری (ICA)

الگوریتم‌های بهینه‌سازی هوشمند، به طور عمده، از فرایندهای طبیعی الهام گرفته شده‌اند. در این الگوریتم‌ها، تکامل زیستی مورد توجه قرار گرفته است و به سایر نمودهای تکامل توجهی نشده است. الگوریتم ICA که در این مقاله مورد استفاده قرار گرفته است، از پدیدهٔ استعمار، به عنوان مرحله‌ای از تکامل سیاسی- اجتماعی بشر الهام گرفته است؛ یعنی نه یک پدیدهٔ طبیعی، بلکه یک پدیدهٔ اجتماعی- انسانی را مورد توجه قرار داده است.

بطور خلاصه، این الگوریتم از چندین کشور در حالت اولیه شروع می‌شود. کشورها در حقیقت جوابهای ممکن مساله هستند و معادل کروموزوم در الگوریتم ژنتیک و ذره در بهینه‌سازی گروه ذرات هستند. همه کشورها به دو دسته تقسیم می‌شوند: امپریالیست و مستعمره. کشورهای استعمارگر با اعمال سیاست جذب (همگون‌سازی)[4] در راستای محورهای مختلف بهینه‌سازی، کشورهای مستعمره را به سمت خود می‌کشند. رقابت امپریالیستی در کنار سیاست همگون‌سازی، هسته اصلی این الگوریتم را تشکیل می‌دهد و باعث می‌شود که کشورها به سمت مینیمم مطلق تابع حرکت کنند.[۱]

شکل زیر، فلوچارت ICA را نشان می‌دهد.

شکل ۱: فلوچارت ICA

[1] Imperialist Competitive Algorithm
[2] Genetic Algorithm
[3] Block Permutation Cipher
[4] Assimilation

۴- تشکیل کشورهای اولیه

در اینجا، جواب‌های ممکن مساله، که عبارتند از تمام جایگشت-های اعداد طبیعی ۱تا N، معادل کشور در ICA هستند. در شروع، تعدادی از این جایگشت‌ها، بعنوان کشورهای اولیه در نظر گرفته می‌شوند.

۵- تعریف تابع cost

هزینه هر کشور با تابع cost سنجیده می‌شود که معادل تابع fitness در GA است. ابتدا مشابه کاری که A.Gorodilov و V.Morozenk[۲] برای متون روسی و با استفاده از تابع ارائه شده توسط T.Jakobsen[۳] انجام داده بودند، ما برای متون انگلیسی از digram frequency حروف الفبای انگلیسی و فاصله (space) استفاده کردیم. Digram یک جفت کاراکتری است که در متن پشت سرهم می‌آیند. ما جدولی متشکل از digram frequency همه حروف الفبای انگلیسی را برای محاسبه تابع cost فراهم کردیم و تابع cost را بصورت زیر در نظر گرفتیم.

$$W(T) = \sum_{ij} \left| T_{ij} - E_{ij} \right| \qquad (2)$$

که در آن T_{ij} فرکانس ij در متن رمزگشایی شده و E_{ij} فرکانس ij در زبان انگلیسی است.

از آنجا که با استفاده از این تابع، بعنوان تابع cost در ICA، به دقت قابل قبولی در جواب نهایی نرسیدیم، تعریف تابع cost را به این صورت تغییر دادیم که علاوه بر digram ها از trigram ها نیز در محاسبه تابع cost استفاده نمودیم. trigram سه کاراکتری است که در متن پشت سرهم می‌آیند. به این ترتیب تابع cost به این ترتیب در می‌آید:

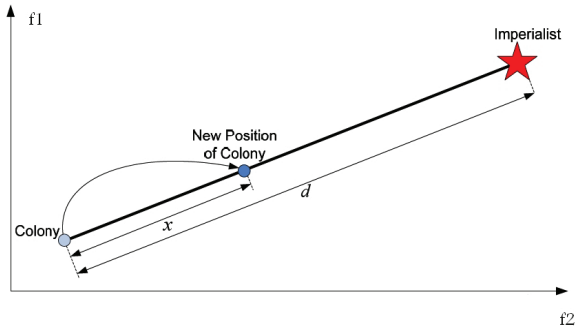
$$W(T) = \beta \cdot \sum_{ij} \left| T_{ij} - E_{ij} \right| + \gamma \cdot \sum_{ijk} \left| T_{ijk} - E_{ijk} \right| \qquad (3)$$

با مقادیر متفاوت β و γ می‌توان وزن‌های متفاوتی را برای هر کدام از n-gram ها در نظر گرفت. در اینجا ما هر دو را برابر یک فرض کرده‌ایم.

هرچه W(T) کوچکتر باشد، متنی که از رمزگشایی به دست آورده‌ایم به متون واقعی انگلیسی مشابه‌تر است. بنابراین در این مسئله به دنبال کمینه کردن W(T) هستیم. البته باید توجه شود که W(T) هرگز برابر صفر نمی‌شود حتی زمانی که کلید صحیح را یافته باشیم.[۵]

۶- نحوه اعمال سیاست همگون‌سازی

در ادامه، کشورهای استعمارگر، با هدف بهبود وضعیت و رشد دادن مستعمره‌های خود، آنها را به سمت خود می‌کشد. این حرکت در شکل ۲ نمایش داده شده است که در آن یک مستعمره به سمت یک امپریالیست x واحد حرکت کرده است و موقعیت جدید آن با رنگ تیره تر نمایش داده شده است. f_1 و f_2 ویژگی-های کشورها هستند، مثل زبان و فرهنگ، و تعداد آنها می‌تواند از ۱ تا هر عدد طبیعی متناهی باشد. در این مسئله، این تعداد برابر یک است.



شکل ۲: شمای کلی حرکت مستعمره به سمت استعمارگر

X یک متغیر تصادفی است با توزیع یکسان که برای آن داریم:

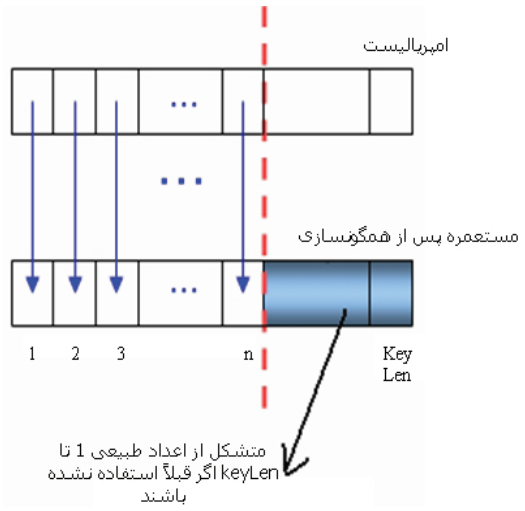
$$x \sim U(0\ ,\ \beta \times d) \qquad (4)$$

که β عددی بزرگتر از ۱ است و d فاصله بین مستعمره و امپریالیست است. یک β بزرگتر از یک باعث می شود که مستعمره ها از دو طرف به امپریالیست نزدیک شوند.

برای پیاده‌سازی سیاست همگون‌سازی، ابتدا باید بتوان فاصله میان امپریالیست را با هر یک از مستعمراتش تعریف کرد. ابتدا فاصله را تعداد بیت‌های متفاوت مستعمره و امپریالیست در نظر گرفتیم. در الگوریتم رقابت استعماری، در مرحله همگون‌سازی، $\beta>1$ و نزدیک به ۲ است تا مستعمره شانس اینکه از امپریالیست خود بهتر شود را داشته باشد. در این مسئله ما نمی‌توان β را بزرگتر از یک گرفت زیرا بیشترین همگون‌سازی که می‌توان انجام داد این است که تعداد ژن‌های متفاوت مستعمره و امپریالست را صفر کرد، یعنی $\beta=1$. البته با این فرض نیز مستعمره شانس داشتن cost ای بهتر از امپریالیست را دارد.

اما نتایج پیاده‌سازی حاکی از این بود که سرعت همگرایی به جواب بهینه بسیار پایین است. از آنجا که چگونگی اعمال سیاست همگون‌سازی در میزان سرعت همگرایی موثر است، تلاش شد به شکل بهتری آن را اعمال کنیم. به این صورت که یک عدد تصادفی بین ۱ و طول کلید انتخاب می‌کنیم، مثلاً K، و K بیت اول

امپریالیست را در K بیت اول مستعمره قرار می‌دهیم. برای بیت‌های باقیمانده، به ترتیب از اعداد ۱ تا طول کلید، اگر آن عدد در بیت‌های پیشین وجود نداشت آن را در مستعمره کپی می‌کنیم و گرنه عدد بعدی را بررسی می‌کنیم. این کار را ادامه می‌دهیم تا زمانی که همه بیت‌های باقیمانده مستعمره تعیین شوند. این عملیات در شکل زیر نشان داده شده است.



شکل ۳: اعمال همگون‌سازی در مسئله مورد بحث

نحوه اعمال همگون‌سازی به این شکل، سرعت همگرایی به جواب را قابل قبول و مطلوب ساخت.

۷- محاسبه قدرت نسبی امپریالیست‌ها

A. Atashpaz-Gargari [۱] برای محاسبه قدرت نسبی امپریالیست‌ها در ICA به روشی که در ادامه آمده، عمل کرده است.

با داشتن هزینه همه امپریالیست‌ها، هزینه نرمالیزه آنها را به صورت زیر در نظر می‌گیریم.

$$C_n = \max_i \{c_i\} - c_n \qquad (۵)$$

که در آن c_n هزینه امپریالیست nام، $\max\{c_i\}$ بیشترین هزینه میان امپریالیست‌ها، و C_n هزینه نرمالیزه شدهٔ این امپریالیست می‌باشد. هر امپریالیستی که دارای هزینه بیشتری باشد (امپریالیست ضعیفتری باشد)، دارای هزینه نرمالیزه کمتری خواهد بود. با داشتن هزینه نرمالیزه، قدرت نسبی هر امپریالیست به صورت زیر محاسبه شده و بر مبنای آن، کشورهای مستعمره بین امپریالیست‌ها تقسیم می‌شوند.

$$p_n = \left| \frac{C_n}{\sum_{i=1}^{N_{imp}} C_i} \right| \qquad (۶)$$

این روش به دلیل محاسبات زیادی که دارد منجر به طولانی شدن زمان اجرای الگوریتم می‌شود. همچنین قدرت نسبی امپریالیستی که در آن $\max\{c_i\} = c_n$ می‌باشد، برابر صفر می‌شود و منجر به حذف این امپریالیست در همان ابتدای کار می‌شود در حالی که ممکن است cost آن چندان از cost سایر امپریالیست‌ها بیشتر نباشد. ما در اینجا روش دیگری را به صورت زیر برای محاسبه قدرت نسبی پیشنهاد می‌کنیم:

$$p_n = \frac{1 - \frac{c_n}{N_{imp}}}{\sum_{i=1}^{N_{imp}} c_n} \qquad (۷)$$
$$N_{imp} - 1$$

در این روش نسبت به روش قبلی، حجم محاسبات کمتر و سرعت اجرا بسیار بالاتر است. همچنین این روش منجر به صفر شدن قدرت نسبی امپریالیست با بیشترین cost در این مرحله نمی‌شود.

۸- شبیه‌سازی

ابتدا به منظور یافتن طول کلید، با تعداد کمی کشور اولیه الگوریتم را به ازای مقادیر مختلف برای طول کلید اجرا می‌کنیم. طول کلیدی که منجر به کمترین cost در این اجراها شود، طول دقیق کلید است. سپس با داشتن طول کلید الگوریتم را با تعداد کافی کشور اولیه برای یافتن مقدار کلید اجرا می‌کنیم. تعداد مناسب کشورها در این مرحله به طول کلید وابسته است. مثلاً برای کلید با طول ۵ بایت، با ۶۰ کشور اولیه به جواب دقیق می‌رسیم. از آنجا که برای کلید با طول n تعداد جواب‌های ممکن !n است، هر چه طول کلید بالاتر برود تعداد کشورهای اولیه نیز باید به صورت نمایی افزایش پیدا کند.

شرط پایان تکرار الگوریتم را رسیدن به حالتی قرار می‌دهیم که فقط یک امپراطوری داشته باشیم. وقتی به این شرط رسیدیم شروع به همگون‌سازی می‌کنیم تا زمانی که تفاوت cost همه کشورها بسیار کم شود و تفاوتی میان آن‌ها نباشد و در نهایت آخرین امپریالیست باقیمانده، کلید رمز می‌باشد.

۹- نتایج و کارهای آینده

برای مثال، اگر متن اصلی زیر که با کلیدی ۱۰ رقمی رمز شده است، بعنوان ورودی الگوریتم در نظر گرفته شود، در مرحله اول، طول کلید با دقت ۱۰۰ درصد بدست می‌آید.

Cipher text = 'In guessing the size of the blocks it helps to count the number of letters in the cryptogram Since the total number of letters in the cryptogram must be an exact multiple of the number of letters in each block the number of letters in each block must be

a factor of the total number of letters in the cryptogram. Finding the change that needs to be made to the order of letters in each block is mainly a question of experimentation. One thing we can do is to take the letters in the first block and try to make words out of them. Then we can see if making the same change to the order of the letters in the other blocks produces anything which makes sense. Alternatively we can try to think of a word that is bound to be in the cryptogram. If we can do that then we can look for the letters in that word near to one another in the ciphertext and work out how to put them into right order .'

کلید بدست آمده توسط الگوریتم '6 3 2 9 8 10 7 1 4 5' و کلید صحیح '6 3 2 8 9 4 7 1 10 5' می‌باشد. همانطور که مشاهده می‌شود، فقط دو رقم دوم و پنجم را جابجا محاسبه نموده است. متن رمزگشایی شده نیز در زیر آمده است.

Decrypted text= 'Iu gnessingeth size tf ohe blo kscit helts po counh tte numbor ef lett rsein theycr ptograi Smnce tho tetal nurbem of lertets in tce hryptog amrmust bn ae exactlmu tiple tf ohe num erbof letsert in eabh clock tne humber lf oetters in each bkocl must ae b factof or the tltao numbef or letteis rn the pryctogrami F.nding hetchangeath t needo ts be mate do the erdor of letters in hace block is mainlyqa uestiof on expernmeitationn O.e thine wg can ds io to tate khe letsert in thi ferst bl ckoand tro ty make dorws out tf ohem. T enhwe canese if magink the s meachange to the or erdof thetle ters ih tne othel brocks puodrces anithyng whimh cakes sense. Altetnarively ce wan try to think af o word that is bo nduto be tn ihe crygtopram. Ie wf can dh toat thee wn can l okofor the letters tn ihat wond rear toeon anothir en the hipcertextdan work utohow totpu them onti rightdor er .'

با وجود اینکه دو رقم کلید صحیح نیست اما با یک نگاه به متن رمزگشایی شده می‌توان این اشتباه را برطرف کرد. مثلاً در متن بالا می‌توان به راحتی تشخیص داد که در همان ۱۰ حرف اول (چون طول کلید ۱۰ است، متن را در بلوکهای ۱۰تایی بررسی می‌کنیم)، حرف دوم و پنجم جابجا هستند و با اصلاح آن، خروجی نهایی بصورت زیر خواهد بود.

Decrypted text = 'In guessing the size of the blocks it helps to count the number of letters in the cryptogram Since the total number of letters in the cryptogram must be an exact multiple of the number of letters in each block the number of letters in each block must be a factor of the total number of letters in the cryptogram. Finding the change that needs to be made to the order of letters in each block is mainly a question of experimentation. One thing we can do is to take the letters in the first block and try to make words out of them. Then we can see if making the same change to the order of the letters in the other blocks produces anything which makes sense. Alternatively we can try to think of a word that is bound to be in the cryptogram. If we can do that then we can look for the letters in that word near to one another in the ciphertext and work out how to put them into right order .'

این روش در بدترین حالت نیاز به دو مورد جابجایی اجزا کلید، توسط کاربر دارد و استفاده از آن، فرآیند کشف کلید را بسیار سرعت می‌بخشد.

البته باید به این نکته توجه شود که با توجه به ماهیت تابع cost که بر اساس digram frequency و trigram frequency می‌باشد، در صورتی که متن رمز شده کوتاه باشد این روش پاسخگو نخواهد بود. رمزهایی که به این روش قابل کشف هستند باید حداقل حدود ۲۰۰۰ بایت باشند تا digram frequency و trigram frequency آنها به frequency های زبان انگلیسی نزدیک باشد.

به عنوان پیشنهادی برای کارهای بعدی روی این مقاله می‌توان برای کشف رمزهای کوتاه، روش R.Morelli و R.Walde[۴] را با روش انجام گرفته در این مقاله ترکیب کرد. آنها با یک روش مبتنی بر کلمه[1]، الگوریتم ژنتیک را برای کشف رمزهای کوتاه، مثلاً حدود ۱۰۰ کاراکتر، بکار برده‌اند.

مراجع

[1] E. Atashpaz-Gargari, C. Lucas, Imperialist Competitive Algorithm: An Algorithm for Optimization Inspired by Imperialistic Competition, IEEE ECE, 2007

[2] A. Gorodilov, V. Morozenko, Genetic Algorithm For Finding The Key's Length And Cryptanalysis Of The Permutation Cipher, International Journal "Information Theories & Applications" Vol.15, 2008.

[3] T. Jakobsen, A Fast Method for the Cryptanalysis of Substitution Ciphers, 1995.

[4] R. Morelli & R. Walde, AWord-Based Genetic Algorithm for Cryptanalysis of Short Cryptograms, FLAIRS, 2003.

[1] Word-based