راه حل DNA Computing جهت مسیریابی در شبکه های کامپیوتری

علیرضا نیکیان

عضو هیئت علمی دانشگاه آزاد اسلامی واحد نجف آباد

Nikian_a@yahoo.com

مرتضی روحانی

دانشگاه آزاد اسلامی واحد نجف آباد – دانشکده فنی ۱ – گروه کامپیوتر

E-mail: rohani.morteza@googlemail.com

خلاصه – در سال ۱۹۹۴ لئونارد آدلمن روشی را معرفی کرد که در آن از مولکول های *DNA* که بنیان حیات در موجودات زنده است برای انجام محاسبات استفاده شد. وی برای اولین بار یکی از مسائل دسته *NP* را که هنوز حل مناسبی با پیچیدگی زمانی خطی برای آن کشف نشده را حل کرد. پس از آن ایده ساخت کامپیوتر های مولکولی مورد توجه بسیاری از دانشمندان قرار گرفت. در این مقاله بر آنیم تا با ترکیب حل مجموعه های احاطه گر و مساله گراف همیلتون با استفاده از مدل مولکولی آدلمن- لیپتون که هر دو از رده مسائل *NP* هستند به یک روش ابتکاری جهت مسائل مسیریابی در شبکه های کامپیوتری دست یابیم. این روش پیچیدگی زمانی این مسائل را که حل آنها با کامپیوتر های پیشرفته امروزی ممکن است سالها طول بکشد را به میزان بسیار قابل توجهی پائین می آورد.

کلما ت کلیدی- مسیریابی (Routing) , Dominating Sets , Adleman- Lipton , DNA Computing .

۱- مقدمه

محاسبات مولکولی، محاسبات از طریق انجام واکنش های شیمیایی بر روی رشته های DNA میباشد[2]. در این محاسبات ابتدا مساله با رشته های DNA کد می شود، سپس الگوریتمی برای پیدا کردن جواب ارائه می شود. انجام محاسبات DNA به خاطر قابلیت پردازش موازی آن با اهمیت است[4]. با قرار دادن تعداد کافی رشته DNA برای حل مساله بعلت انجام همزمان عملیات روی تمام رشته ها میتوان در یک زمان کم مساله را حل کرد. این امر باعث کمتر شدن پیچیدگی زمانی الگوریتم های مولکولی نسبت به الگوریتم های معمولی میشود. بررسی های شیمیایی مشخص میکند که DNA توالی از نوکلئوتید هاست. هر نوکلئوتید از سه قسمت اصلی شامل: یک قند ، یک گروه فسفات معدنی و یکی از چهار باز آلی نیتروژن دار حلقوی آدنین (Adenine) ، گوانین (Guanine) ، سیتوزین (Cytosine) و تیمین (Thymine) تشکیل شده است. این چهار باز آلی به اختصار با حروف A، G، C و T نمایش میدهند. دو نوکلئوتید از طریق یک پیوند فسفودی استر به هم متصل می شوند به این صورت که گروه هیدروکسیل متصل به کربن '۳- قند یک نوکلئوتید با گروه فسفات متصل به کربن'۵- قند نوکلئوتید دیگر واکنش داده و پیوند فسفودی استر را بوجود می آورد. قند موجود در نوکلئوتید، یک قند کربنی است، کربن های موجود در قند یک نوکلئوتید را شماره گذاری می کنند و آنها را با اعداد '۱، '۲، '۳، '۴ و'۵ نمایش می دهند. نوکلئوتید ها بدین صورت به هم متصل می شوند و زنجیره پلی نوکلئوتید که همان DNA است را بوجود می آورند.تمامی نوکلئوتید ها در این

زنجیره دارای جهت یکسان هستند. تقریبا" بیست سال پس از کشف ماهیت شیمیایی DNA دو دانشمند بنامهای واتسون و کریک یک مدل سه بعدی از ساختمان DNA ارائه کردند که شرح آن اینگونه است [3]: DNA یک مارپیچ دو رشته ایست که رشته های آن بدور یک محور مرکزی (معمولا راستگرد) پیچ میخورند. طبق مدل واتسون کریک دو رشته DNA در مقابل هم قرار دارند و پیوند های هیدروژنی باز آدنین از یک رشته با باز تیمین از رشته دیگر و باز گوانین رشته با باز سیتوزین رشته مقابل باعث اتصال این دو رشته DNA به هم میشود و تشکیل مارپیچ را میدهد. از دلایل اصلی استفاده از زیست شناسی مولکولی برای جواب دادن به مسائل محاسباتی میتوان به چگالی بالای ذخیره سازی DNA (یک بیت در یک نانومتر مربع و در رسانه ذخیره سازی مانند TAPE ، 10^{12} نانو متر مربع ذخیره میشود [11].) و پردازش در سطح موازی اشاره کرد. حال مقایسه را با یک مثال ناچیز از مسأله TSP (مثلا برای ۲۰ شهر) ارائه میکنیم [11] : با معماری Neumann ابتدا باید درخت جستجو ایجاد شود و سپس همه راه های ممکن تولید شود که معادل 45 Million GBytes نیاز است. (!۱۸ مسیر با کلمات ۷ بایتی) که با کامپیوتری با سرعت 100 MIPS ، دو سال زمان برای تولید همه راه های ممکن نیاز است. این در حالی است که با یک کامپیوتر DNA ، 10^{15} فقط یک نانو مول است و همچنین هیچ جستجویی برای پیدا کردن کوتاه ترین مسیر نیاز نیست.

۲- مدل محاسباتی مولکولی

مفهوم مدل و محاسبات مولکولی به معنای نحوه نمایش مساله و تعیین عملیات مولکولی است که در یک چارچوب خاص در حل مسائل و نوشتن الگوریتم آنها بکار برده میشود. بنابراین اولین قدم در نوشتن هر الگوریتم مولکولی تعیین مدل محاسباتی مولکولی است که الگوریتم در آن مدل نوشته میشود.

۲-۱- مدل آدلمن - لیپتون

این مدل توسط آدلمن و لیپتون ارائه شده است و از جمله مدل هایی است که بسیاری از الگوریتم ها در آن نوشته شده است [1,5,6,7,12] .

۲-۲- عملیات مدل آدلمن - لیپتون

حال به شرح تعدادی از عملیات مدل آدلمن- لیپتون که در این مقاله بکار برده ایم می پردازیم :

الف) ساخت : این عمل عبارتست از ساختن یک DNA تک رشته ای با طول مطلوب که حاوی نوکلئوتید ها با ترتیب خاص باشد. این عمل توسط ماشین سازنده DNA انجام میشود. در این ماشین مولکول DNA مورد نظر نوکلئوتید به نوکلئوتید بدون استفاده از الگو ساخته می شود.

ب) استخراج : لوله آزمایش T و تک رشته DNA ، S مفروض است. این عمل دو لوله آزمایش T_1 و T_2 را از روی لوله آزمایش T تولید میکند بطوری که T_1 شامل تمام رشته هایی از T است که رشته S بعنوان زیر رشته در آنها وجود دارد و T_2 مولکول های DNA ای از T را دارا میباشد که شامل زیر رشته S نباشند. این عمل با تابع $Extract$ نمایش داده می شود. $Extract(T,T_1,T_2,S)$

شکل گسترش یافته این عمل نیز می تواند بصورت : $Extract(T,T_1,T_2,i,S)$ تعریف شود که در آن رشته هایی که در موقعیت I ام آنها زیر رشته S وجود دارد در T_1 جدا می شوند.

ج) جدا سازی : این عمل لوله آزمایش T و عدد مثبت L را بعنوان طول یک رشته دوتایی DNA می گیرد و آزمایش جدید T_1 را تولید می کند که T_1 شامل همه رشته های DNA از T است که طول آنها برابر L باشد. . این عمل با تابع Separate نمایش داده می شود. $Separate(T,T_1,l)$ شکل دیگری از این عمل میتواند بصورت $SeparateMin(T,T_1)$ باشد که کوتاه ترین رشته موجود در T را درون T_1 قرار میدهد. از نظر عملی این عمل امکان انتخاب یک DNA با طول خاص را از میان DNA هایی با طول های مختلف به ما می دهد. طول بر حسب تعداد نوکلئوتید محاسبه می شود، برای این کار از روشی به نام ژل الکتروفورز (Gel Electrophores) استفاده می شود. در این روش از صفحه ای که بر روی آن یک ژل مانند پلی اکریلامید قرار دارد و دارای چاهک بطور عمودی است، استفاده می شود نمونه های DNA بطور جداگانه وارد چاهک های ژل می شوند سپس یک میدان الکتریکی به ژل اعمال می شود. مولکول های

DNA کوچکتر با سرعت بیشتر و مولکول های DNA بزرگتر با سرعت کمتر از درون چاهک ها شروع به حرکت به داخل ژل می نمایند. بعد از مدت زمانی مولکول های DNA بر اساس اندازه در مکان های متفاوت در طول ژل قرار می گیرند و بصورت نوار هایی با باند مجزا تقسیم می شوند.

د) مخلوط کردن : این عمل دو لوله آزمایش T_1 و T_2 را بعنوان ورودی می گیرد و لوله آزمایش T را که شامل رشته های T_1 و T_2 است می سازد. این عمل با تابع Merge انجام می شود. $Merge(T_1,T_2,T)$

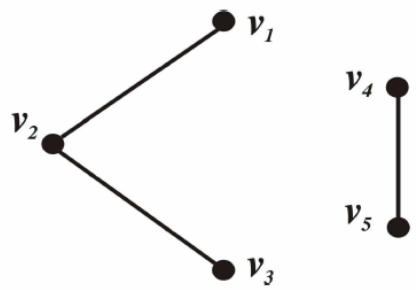
ن) هیبریداسیون : این عمل باعث می شود تک رشته ای های یک لوله آزمایش T با رشته مکملشان که در T قرار دارند هیبرید شود و به دو رشته ای تبدیل گردند. این عمل با تابع *Hybrid(T)* نشان داده می شود.

و) تشخیص : لوله آزمایش T مفروض است. این عمل با بررسی لوله آزمایش در صورتی که حداقل یک رشته DNA در آن وجود داشته باشد جواب مثبت و در غیر اینصورت جواب منفی برمی گرداند. این عمل با تابع *Detect(T)* نشان داده می شود.

ه) ازدیاد رشته : این عمل در واقع با روش PCR انجام می شود آنگونه که آنزیم DNA-Polymerase را برای ساختن نسخه های زیادی از یک رشته DNA بکار می برد. بدین ترتیب که اگر در یک لوله آزمایش تعدادی DNA داشته باشیم و روی یک یا چند تا از آنها عمل ازدیاد انجام گیرد، تعداد توالی های ازدیاد شده به قدری می شود که عملا رشته هایی را که برای آنها این عمل انجام نشده است، می توان نادیده گرفت. این عمل با تابع *PCR(T,x,y)* نشان داده می شود که در آن x,y آغازگرهای مورد استفاده هستند.

۳- مجموعه احاطه گر

زیر مجموعه U از مجموعه رئوس گراف G را در نظر بگیرید. U را مجموعه احاطه گر گراف G گوییم ، هرگاه هر راس گراف G با یکی از اعضای این مجموعه مجاور باشد[13]. برای مثال در گراف داده شده G در شکل ۱ مجموعه رئوس $U=\{v_1,v_3,v_4\}$ یک مجموعه احاطه گر گراف G می باشد. زیر مجموعه U از رئوس گراف G را کوچکترین مجموعه احاطه گر گوییم، هرگاه U دارای کمترین تعداد اعضاء در بین مجموعه های احاطه گر G باشد. برای مثال مجموعه $U=\{v_2,v_4\}$ کوچکترین مجموعه احاطه گر گراف G نشان داده شده در شکل ۳-۱ است.



شکل ۱: گراف G .

مساله مجموعه احاطه گر در یک گراف از جمله مسائل NP – سخت می باشد[12,13]. در مدل مولکولی آدلمن – لیپتون الگوریتمی توسط شانگ و ژئو [8] برای این مساله ارائه شده است که در اینجا پس از شرح مساله به بیان کاربردی آن در مسائل مسیریابی در شبکه می پردازیم.

گراف G=(V,E) که در آن |V|=n و |E|=m مفروض است. برای نمایش مجموعه های احاطه گر در این گراف می توان از یک عدد دودویی n بیتی استفاده نمود و هر یک از مجموعه های احاطه گر گراف را کد کرد. V^1 به عنوان مجموعه احاطه گر گراف G مفروض است. هر یک از بیت های عدد متناظر با مجموعه V^1 متناظر با یکی از رئوس گراف G است. اگر بیتی ۱ باشد نمایانگر حضور رأس متناظر در V^1 است و صفر بودن بیت نمایانگر عدم حضور آن رأس در V / V^1 خواهد بود. برای مثال عدد دودویی ۰۱۰۱۰ مجموعه احاطه گر مینیمم $\{v_2,v_4\}$ را در گراف شکل ۱ کد می کند.

حال برای نمایش مجموعه احاطه گر کافیست با رشته های DNA اعداد n بیتی را کدگذاری نمود. برای تولید عدد n بیتی X بصورت، $x_n,x_{n-1},...,x_1$ برای هر بیت i دو رشته x_i^0 و x_i^1 در نظر گرفته میشود که x_i^0 نشانگر مقدار صفر

در بیت i ام و x_i^1 نشانگر مقدار یک در این بیت است.برای تولید تمام 2^n عدد باینری که با n بیت می توان تولید کرد از روش لیپتون استفاده شده است. به این صورت که به هر x_i^0 یک رشته تصادفی ۲۰ تایی DNA به نام O_i^0 و به هر x_i^1 یک رشته تصادفی ۳۰ نوکلئوتیدی O_i^1 نسبت داده می شود. نیمه پایین O_i^0 یا O_i^1 با c_i و نیمه بالایی آنها با d_i مشخص می شود. در واقع O_i^0 یا O_i^1 برابر است با رشته $c_i\ d_i$. بین دو بیت مجاور x_i و $x_j = x_{i+1}$ یک یال در نظر گرفته می شود و برای هر یال از بیت i به بیت j رشته $d_i\ c_i \downarrow$ تولید می شود. با توجه به روش نمایش بالا برای تولید تمام مجموعه های احاطه گر رشته های زیر توسط عمل ساخت بوجود می آید و داخل لوله آزمایش T_0 ریخته میشود :

۱) به ازای هر بیت i کپی های زیادی از رشته $c_i\ d_i$.

۲) به ازای هر یال از بیت i به j کپی های زیادی از رشته $d_i\ c_i \downarrow$.

۳) کپی های زیادی از رشته مکمل نیمه اول رشته مربوط به بیت آغازی ، x_n، و همچنین کپی های زیادی از نیمه دوم بیت پایانی ، x_1.

این رشته ها با استفاده از عمل اتصال با هم مخلوط می شوند و رشته های دوتایی n بیتی ، که تمام 2^n حالت ممکن تولید می شود.

```
1.  RoutingSolution(T0, n, m)
2.  begin
3.    for i=1 to n do begin
4.        Extract(T0, T0, T1, x_i^1)
5.        for each v_j adjacent v_i do begin
6.            Extract(T1, T2, T1, x_j^1)
7.            Merge(T0, T2, T0)
8.        end for
9.    end for
10.   SeparateMin(T1, T2)
11.   if ( Detect(T2) = = 'yes' ) then
12.       begin
13.           Merge(T2, x_s^1)
```

شکل الف-۲:الگوریتم مسیریابی در شبکه های کامپیوتری

```
14.           Merge(T2, x_E^1)
15.           for i=1 to n do begin
16.               init(T3, T2, ↓ d_i c_{i+1})
17.           end for
18.           Hybrid(T3)
19.           PCR(T4, x_S^1, x_E^1)
20.           SeparateMin(T4, T5)
21.           if ( Detect(T5) = = 'yes' ) then
22.               return(T5)
23.           end if
24.       end if
25.       else report 'no dominating set'
26.   end RoutingSolution.
```

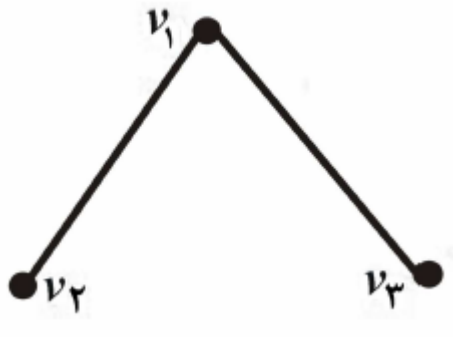
شکل ب-۲:الگوریتم مسیریابی در شبکه های کامپیوتری

حال الگوریتم نشان داده شده در شکل ۲ ابتدا از بین مجموعه رشته های تولید شده بالا مجموعه احاطه گر مینیمم را که عبارتست از مجموعه کمترین تعداد رئوسی که با تمام رئوس گراف G همسایه باشد را تولید می کند. با اجرای حلقه اول الگوریتم به ازای هر یک از رئوس گراف ابتدا بر اساس حضور رشته متناظر با رأس i ام لوله آزمایش T_0 به دو لوله T_0 و T_1 تقسیم میشود. T_0 شامل رشته هایی است که در آنها x_i^1 و T_1 رشته هایی که در آنها x_i^0 وجود دارند. سپس در مرحله بعد در یک حلقه به ازای هر یک از رئوس مجاور v_i یک بار عملیات زیر اجرا می شود : در ابتدا رشته هایی که در آنها x_i^1 و x_j^1 وجود دارد در T_2 قرار داده میشود و رشته هایی که در آنها x_i^0 و x_j^0 وجود داشته باشد در T_1 ریخته می شود و بعبارت دیگر دور ریخته می شوند. به این معنا که رشته هایی که نمی توانند کد شده یک مجموعه احاطه گر باشند مرحله به مرحله از لوله آزمایش اولیه بیرون کشیده می شوند و رشته هایی که در T_2 قرار می گیرند آنهایی هستند که یک احاطه گر معتبر را نشان می دهند. در نهایت در خط ۱۰ رشته ای که کمترین طول را دارد با استفاده از عمل جداسازی بر حسب طول انتخاب می شود که این رشته همان مجموعه احاطه گر مینیمم است. در صورتی که مجموعه احاطه گر مینیمم در لوله آزمایش T_2 باشد، الگوریتم از این پس (خط ۱۱) اقدام به ساخت همه راه های ممکن از رئوس جاری می پردازد و در غیر اینصورت پیغام no dominating set را

مبنی بر عدم وجود مجموعه احاطه گر برمی گرداند. حال به بررسی چگونگی ایجاد مسیرهای ممکن جهت مسیریابی می پردازیم : برای این کار از آنجا که مـی دانـیم مطمئنـا همـه رئوس گراف مفروض با مجموعه احاطه گر بدسـت آمـده تـا این مرحله در همسایگی قرار دارند ابتدا رأس مبـدأ (x_S^1) و مقصد (x_E^1) را به این مجموعه اضافه می کنـیم (هـر چنـد ممکن است از قبل وجود داشته باشـند.) و سـپس در یـک حلقه در صورت وجود یال بـین رئـوس i ام و i+1 ام رشـته هـــــای معـــــادل آن ($\downarrow d_i c_{i+1}$) در تـــــابع $init(T_3, T_2, \downarrow d_i c_{i+1})$ تولیـد میـشود. پـس از آن تـابع هیبرید رشته های داخل T_3 را هیبریـد مـی کنـد تـا تـابع PCR بتواند آنها را تکثیر کند. تابع تنها رشته هایی را که بـا x_S^1 آغاز و به x_E^1 خـتم میـشوند را تکثیـر مـی کنـد زیـرا آغازگر های مورد استفاده x_S^1 و x_E^1 هستند.

در نهایت تابع $SeparateMin(T_4, T_5)$ کوتاه ترین رشته موجود در T_4 را درون لوله آزمایش T_5 قرار میدهد کـه در حقیقت همان پاسخ مورد نظر ما می باشـد. در صـورتی کـه درون لوله آزمایش T_5 رشته ای یافته باشد آنرا توسط تـابع $Detect(T_5)$ تشخیص میدهد.

حال بعنوان مثال اجرای این الگوریتم بر روی گراف شکل ۳ بیان می شود. ابتدا لوله آزمایش T_0 با ۸ رشته های ۰۰۰ و ۰۰۱ و ۰۱۰ و ۰۱۱ و ۱۰۰ و ۱۰۱ و ۱۱۰و ۱۱۱ پـر مـی شود. حلقه اول الگوریتم به ازای هر یک از رئـوس گـراف، ۳ بار اجرا می شود. با اجرای مرحله اول حلقه دو لوله آزمایش T_0 و T_1 تولید می شود. لوله T_0 شامل رشته های ۱ * * (* می تواند ۰ یا ۱ باشد.) و لولـه T_1 شـامل رشـته هـای ۰ * * است. از آنجا ئیکه تعداد رئوس مجاور v_1 دو تاسـت حلقه دوم دو بار اجرا خواهد شد. یک بار بـه ازای رأس v_2و یک بار به ازای رأس v_3. در اولین اجرای حلقـه داخلـی دو لوله آزمایش بدست می آید، لوله T_2 که شامل رشـته هـای ۱۰* است و لوله T_1 که شامل رشته هـای ۰۰* اسـت. بـا توجه به تعریف مجموعه احاطه گر لوله T_2 می تواند حاوی رشته های معرف مجموعه احاطه گر باشد در نتیجه در لوله T_0 ریخته می شود. در اجرای دوم حلقه داخلی بـاز هـم دو لوله آزمایش تولید می شود، لوله T_2 که تنها شـامل رشـته ۱۰۰ است و لوله T_1 که حاوی رشـته ۰۰۰ مـی باشـد. بـا توجـه بـه تعریـف مجموعـه احاطـه گـر T_2 حـاوی رشـته مجموعه احاطه گر است، در نتیجه داخـل لولـه T_0 ریختـه می شود و محتویات لوله T_2 دور ریخته می شود. در نهایت کوتاه ترین رشته تشخیص داده می شـود و ایـن رشـته کـه برابر ۰۰۱ است و نشان دهنده مجموعه احاطه گـر مینـیمم است به مرحله بعد یعنی مسیر یابی تحویل داده می شود.



شکل ۳: نمونه گراف برای مجموعه احاطه گر .

۴- نتیجه‌گیری

همان طور که مشاهده شد در این مقاله ایـن ایـده ابتکـاری ارائه شد که ابتدا از یک گراف، مجموعه احاطـه گـر آن کـه عبارت بود از حد اقل مجموعه رئوسـی کـه بـا همـه رئـوس گراف همسایه باشند، استخراج شـود و سـپس راس مبـدأ و مقصد که مورد نظر ما در مسیر یابی است را به این مجموعه اضافه کردیم و در نهایت مـسیرهای بهینـه ای کـه در ایـن مجموعه رئوس بود را استخراج نمودیم. با توجه به دو حلقه تودرتو در بخش مجموعه احاطه گر این الگوریتم که هر یک از درجه n است پیچیدگی زمانی الگوریتم تا این بخش برای هر گـراف بـا n رأس $O(n^2)$ اسـت، پـس از آن در بخـش بعـدی الگـوریتم شـامل یـک حلقـه بـا تکـرار n اسـت کـه پیچیدگی زمانی آن $O(n)$ می باشد. در نتیجه پیچیـدگی زمـانی نهـایی الگـوریتم برابـر بـا مجمـوع ایـن دو اسـت $(O(n^2)+O(n))$. که می توان آنرا بـصورت $O(n^2)$ در نظر گرفت. و با توجه به اینکه 2^n رشـته DNA بـرای حـل مسأله نیاز است که ضریبی از مسیر هـا در گـراف نیـز مـی باشد، پیچیدگی حافظه آن $O(2^n)$ است. همان طـور کـه می دانید برای مسأله مجموعه احاطه گر که یکـی از کـاربرد های آن بصورت ترکیبـی در ایـن مقالـه مـورد بحـث قـرار گرفت، تا بحال حل مناسبی در رتبه زمانی چنـد جملـه ای،

ارائه نشده و حتی با کامپیوتر های بسیار پیشرفته کنونی در مقادیر ورودی بالا ، برای الگوریتم بسیار زمان بر است و حتی غیر ممکن بودن ارائه حل مناسب برای آن نیز اثبات نشده است که این موضوع اهمیت پرداختن به الگوریتم های محاسبات مولکولی را بالا می برد.

۵- جمع بندی

در این مقاله حل یکی از مسائل ترکیباتی در چهار چوب مسائل گراف و ترکیب آن با یکی از پر کاربرد ترین مسائل روز دنیای کامپیوتر یعنی مسیر یابی در شبکه ها در مدل محاسباتی مولکولی ارائه گشت. مدل محاسباتی بررسی شده در این مقاله مدل ادلمن – لیپتون بود [1,5,6,7,12]. همان طور که مشاهده شد بسیاری از مسائل ترکیباتی در این مدل بصورت کارا می تواند انجام پذیرد و برای حل هر مسأله ابتدا روشی برای کد گذاری مصادیق مسآله با رشته های DNA ارائه شده و سپس با توجه به عملیات مجاز در آن مدل الگوریتمی برای تولید جواب های ممکن ارائه و در نهایت روشی برای مشخص کردن جواب اصلی تعیین می شود. بسیاری از الگوریتم ها بصورت عملی در آزمایشگاه نیز پیاده سازی و انجام شده اند و نتایج حاصل قابل قبول و امید وار کننده بوده است.

مراجع

[1] L.M. Adleman, Molecular computation of solution to combinatorial problems , Science 266 (1994), 1021- 1029

[2] R. Deaton, M. Garzon, J.A Rose, D.R. Franceschetti, and S.E. Stevens JR., Reliability and efficiency of a DNA based computation , Phys. Rev. Lett. 80 (1997), 417-420

[3] M. Kadkhodaie, Molecular algorithm for Hamilton cycle problem, M.Sc. dissertion, Faculty of Science University of Tehran, 2003.

[4] DNA computing A Primer by: Will Ryu

[5] L.M. Adleman, Computing with dna, Sci. Am.279 (1998), 54-61

[6] R.J. Lipton, DNA solution of hard computational problems, Science. 268 (1995), 542-545

[7] R.J. Lipton, Using DNA to solve NP-complete problems, Science. 268 (1995), 542-545

[8] M. Guo, M. Ho, and W.L. Chang , Fast parallel molecular solution to the dominating-set problem on massively parallel bio-computing, Parallel Comput. 30 (2004), 1109-1125

[9] G.P. Raja Sekhar, DNA Computing- Graph Algorithms , Department of Mathematics, Indian Institut of Technology

[10] N. Pisanti, DNA Computing : a survey, Department of Computer Science University of pisa corso Italia, 40 56125, Pisa Italy

[11] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to algorithms, 2nd ed., MIT Press, Cambridge, 2001.

[12] G. Kabirian Solving NP Problems with DNA Computing, M.Sc. dissertion, Faculty of Science University of Tehran,July 2006

[13] W.L Chang , M. Guo. Solving the Dominating-Set problem in Adleman lipton's Model. Department of computer software the university of Aizu, Aizu-wakamatsu City, Fukushima, Japan