ارائه الگوریتمی کارا جهت همترازی رشته‌های بتای پروتئین

مصطفی سبزه‌کار[۱]، مسعود بیجاری[۲]
[۱] استادیارگروه مهندسی کامپیوتر، دانشگاه صنعتی بیرجند، sabzekar@birjandut.ac.ir
[۲] فارغ التحصیل کارشناسی ارشد، گروه مهندسی کامپیوتر، دانشگاه آزاد واحد بیرجند ma3oudphp@yahoo.com

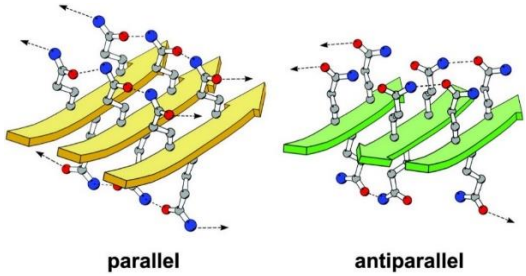
چکیده - پروتئین‌ها مولکول‌های بزرگی هستند که نقش عملکردی در موجودات زنده را بر عهده دارند. اسیدهای آمینه اجزای تشکیل دهنده پروتئین‌ها هستند. نحوه عملکرد یک پروتئین ارتباط مستقیمی با ساختار آن دارد. اسیدهای آمینه‌ای که با ترتیب خاصی کنار هم قرار گرفته‌اند (ساختار اول) با پیچ و تاب خوردن در رو و کنار یکدیگر زیر واحدهایی را می‌سازند (ساختار دوم) که با استفاده از آن ها ساختار نهایی و سه بعدی پروتئین تشکیل می‌شود (ساختار سوم). اجزای اصلی تشکیل دهنده ساختار دوم پروتئین، مارپیچ‌های آلفا و صفحات بتا هستند. تعیین ساختار صفحات بتای یک پروتئین از جمله مسائل پیچیده ی حوزه ی بیوانفورماتیک ساختاری در مسیر تعیین ساختار نهایی آن است. پیش‌بینی ساختار صفحات بتای پروتئین در سه گام تعیین نقشه تماس، همترازی رشته‌های بتا، تعیین ساختار صفحات بتا انجام می‌پذیرد. یکی از چالش‌های مهم در این مسیر، بالابودن زمان پیش‌بینی است. بنابراین یکی از بهبودهای مهم در مسیر پیش‌بینی این است که هر یک از مراحل در کوتاه‌ترین زمان ممکن انجام شود. در گام دوم (همترازی) تمایل هر زوج رشته ی بتا به عنوان اجزای تشکیل دهنده‌ی صفحات بتا برای تعامل با یکدیگر در هر یک از دو حالت موازی همسو و ناهمسو توسط روش پیشنهادی همترازی محاسبه می‌شود. با افزایش تعداد رشته‌های بتا، زمان لازم برای همترازی و در نتیجه زمان کل تعیین ساختار صفحات بتا افزایش می‌یابد. در این مقاله برای برخورد با این چالش، با سعی در تطبیق و به کار بردن الگوریتم *Hirschberg* در مسئله خاص همترازی رشته‌های بتا و هم چنین برنامه‌نویسی *OpenMP* برای اجرای هم‌زمان بخش‌هایی از الگوریتم پویا بر روی چند پردازنده برای دستیابی به سرعت بیشتر استفاده می‌کنیم.

کلید واژه- پروتئین، صفحات بتا، پیش بینی، همترازی.

۱- مقدمه

بر خلاف مولکول‌های DNA و RNA که اطلاعات ژنتیکی را درخود دارند، پروتئین‌ها نقش عملکردی در موجودات زنده را عهده‌دار هستند. پروتئین‌ها مواد مغذی اصلی سلول‌های زنده هستند. آنها مولکول‌های حیاتی جهت تعیین عملکرد و ساختار سلول‌ها می‌باشند. تمامی فعالیت‌هایی که در سلول انجام می‌شود به عهده پروتئین‌هاست. اسیدهای آمینه اجزای اصلی تشکیل دهنده ساختار پروتئین‌ها هستند که با توالی‌های مشخص خطی به طریق پیوند کووالانسی به یکدیگر متصل می‌باشند. اسیدهای آمینه با پیچ و تاب خوردن در فضا در نهایت ساختار پروتئین را مشخص می‌نمایند. بدلیل اینکه نحوه عملکرد پروتئین ارتباط مستقیمی با ساختار آن دارد، نیازمند مطالعه و شناخت ساختار هستیم. ساختارهای پروتئینی شامل از ده‌ها تا چندین هزار اسیدهای آمینه است[۱]. چهار سطح مجزا از ساختار پروتئین وجود دارد که شامل ساختار اول، ساختار دوم، ساختار سوم و ساختار چهارم می‌باشد. ساختار اولیه یک پروتئین به توالی خطی از اسیدهای آمینه در زنجیره پلی‌پپتیدی اشاره دارد. به نظم‌های موضعی گفته می‌شود که پروتئین در حین تاشدگی به خود می‌گیرد. ساختار دوم، ارتباط فضایی بین اسیدهای آمینه مجاور را در قطعات پلی‌پپتیدی نشان می‌دهد. توالی اسیدهای آمینه با پیچ‌وتاب خوردن ساختار دوم پروتئین را ایجاد می‌کنند. زیرساختارهای محلی منظم از صفحات بتا و مارپیچ‌های آلفا ساختار دوم پروتئین را تشکیل می‌دهند. برخلاف مارپیچ آلفا که از یک منطقه پیوسته ساخته‌شده است این ساختمان از ترکیب چندین منطقه زنجیر پلی‌پپتیدی تشکیل‌شده است. رشته‌های بتا معمولاً ۵ تا ۱۰ اسیدآمینه طول دارند و در مجاور یکدیگر ردیف می‌شوند، بطوریکه پیوندهای هیدروژنی می‌توانند بین گروه‌های کربونیل یک رشته بتا و گروه‌های آمینوی روی رشته مجاور و برعکس تشکیل شوند. رشته‌های بتا به دو طریق، موازی همسو و موازی ناهمسو می‌توانند برای تشکیل صفحات چین‌خورده با هم

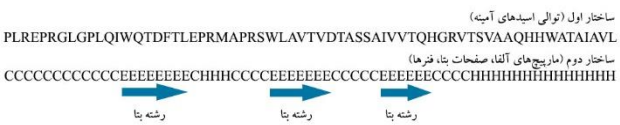
میان کنش داشته باشند. شکل ۱ قرارگیری موازی همسو و موازی ناهمسوی رشته‌های بتا در کنار یکدیگر را نمایش می‌دهد.



شکل ۱: قرارگیری موازی همسو و موازی ناهمسوی رشته‌های بتا در کنار یکدیگر.

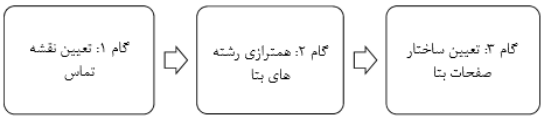
علاوه بر دو ساختار گفته‌شده، نوع دیگری بنام حلقه نیز وجود دارد که دارای اهمیت کمتری می‌باشد. به نواحی در زنجیره پروتئین که بین مارپیچ‌های آلفا و صفحات بتا قرارگرفته‌اند و طول و شکل سه‌بعدی دارند و درنهایت بر روی سطح ساختار قرار می‌گیرند. حلقه گفته می‌شود. نواحی از پروتئین که هیچ‌کدام از این موارد نباشد فنر یا حلقه تصادفی نام دارد. در فرهنگ واژگانی DSSP [۲] مارپیچ‌های آلفا را با H، صفحات بتا را با E و حلقه‌ها را با C نمایش می‌دهند. به حالت سه‌بعدی که پروتئین بعد از پیچش به خود می‌گیرد گفته می‌شود، ساختار سوم و به حالت قرارگیری چند پروتئین در فضا کنار یکدیگر، ساختار چهارم آن می‌گویند.

صفحات بتا یکی از اجزای مهم و پرتکرار در پروتئین‌ها هستند که از بخش‌های مجزایی به نام رشته‌های بتا تشکیل شده‌اند. بدلیل اینکه این رشته‌ها ناپایدار بوده و به یکدیگر نیرو وارد می‌کنند، رشته‌های بتا در فضا حرکت می‌کنند و با قرار گرفتن در کنار یکدیگر، ساختارهای پایداری بنام صفحات بتا ایجاد می‌کنند. پیش‌بینی ساختار صفحات بتای پروتئین یکی از مهم‌ترین مسائل بیوانفورماتیک است که نقش مهمی در تشخیص و درمان بسیاری از بیماری‌ها از جمله آلزایمر دارد [۳]. ورودی این مسئله ساختار اول و ساختار دوم پروتئین هستند. همان‌طور که در شکل ۲ مشاهده می‌شود، یک پروتئین می‌تواند شامل چندین رشته‌ی بتا باشد پس از پردازش ورودی مساله و جداسازی رشته‌های بتا همترازی برای هر جفت رشته بتا آغاز می‌گردد.



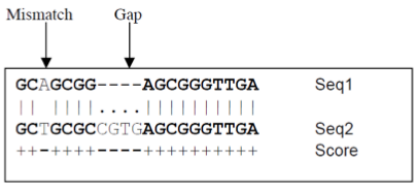
شکل ۲: ورودی مسئله شامل توالی اسیدهای آمینه که هر کدام با یک حرف مشخص شده‌اند. همچنین ساختار دوم که محل رشته‌های بتا در آن مشخص شده است.

مراحل پیش‌بینی ساختار صفحات بتای پروتئین در شکل ۳ نمایش داده شده است.



شکل ۳: مراحل پیش‌بینی ساختار صفحات بتا

در گام اول احتمالات تماس اسیدهای آمینه در دو رشته بتا را که نقشه تماس می‌نامند، توسط روش‌هایی پیش‌بینی می‌نمایند. در گام دوم، که هدف این مقاله بر آن بنا شده، مرتب‌کردن رشته‌های بتا به نحوی است که نواحی شبیه به هم آن ها را پیدا کنیم. زیرا رشته‌های مشابه به احتمال زیاد روابط تکاملی، ساختار و حتی عملکرد مشابهی دارند. به این گام همترازی می‌گوییم. در این مرحله رشته‌های بتای به‌دست‌آمده را دو به دو با یکدیگر به‌صورت موازی همسو و ناهمسو همتراز می‌کنیم. در هر مرحله امتیاز همترازی را محاسبه کرده، تا درنهایت نحوه قرار گرفتن رشته‌ها در صفحات بتا هم از لحاظ ترتیب قرار گرفتن در کنار یکدیگر و هم از لحاظ همسو یا ناهمسو بودن جهت رشته‌ها در یک صفحه مشخص گردد. به فواصلی که پس از مرحله همترازی در ابتدا یا انتهای هر رشته ممکن است به وجود آید گپ گفته می‌شود. در این مرحله امتیاز صفحات بتا محاسبه می‌گردد. شکل ۴ نحوه‌ی همترازی دو رشته را نمایش می‌دهد. سرانجام در گام چهارم براساس امتیازات به‌دست‌آمده برای تمام ساختارها، ساختار نهایی پروتئین به دست می‌آید. روش‌های مختلفی برای امتیازدهی به ساختارها وجود دارد.



شکل ۴: همترازی دو رشته.

با افزایش تعداد رشته‌های بتا، جستجو برای یافتن بهترین همترازی برای هر زوج رشته در دو جهت همسو و ناهمسو، پیچیدگی زمانی و فضایی بالایی را به دنبال دارد. حال به نظر می‌رسد اگر روشی بتواند زمان و فضای حالت مساله همترازی را کاهش دهد منجر به کاهش زمان در کل مساله پیش بینی ساختار صفحات بتا شودکه این از اهمیت زیادی برخوردار است.

۲- پیشینه

همترازی یکی از پرکاربردترین مراحل میانی در مسائل ژنومیک و پروتئومیک در بیوانفورماتیک است. هدف از همترازی یافتن نواحی مشابه در رشته‌های ژنی یا توالی‌های آمینواسیدی است. یافتن این نواحی اطلاعات ارزشمندی را در اختیار محققین قرار می‌دهد. همترازی می‌تواند بسته به این که کار همترازی بر روی بخشی از توالی انجام می‌شود با بر روی تمام آنها به دو صورت محلی و یا سراسری باشد[۴]. از دید دیگر نیز اگر فقط دو توالی را با یکدیگر همتراز کنیم، آن را همترازی دو به دو و اگر هم زمان چند توالی را همتراز کنیم آن را همترازی چندگانه می‌نامند. همچنین به منظور همترازی، دسته‌ای از روش‌ها با لغزاندن دو رشته در امتداد هم بهترین حالت همترازی را یافته‌اند و دسته‌ای دیگر، از روش‌های پایه‌ی همترازی مانند نیدلمن-وانچ [۵] استفاده کرده‌اند. دسته‌ی اول اجازه‌ی وجود شکاف در همترازی را نمی‌دهند. در حالی که وجود شکاف باعث انعطاف بیشتر و دقت بالاتر در همترازی شده و به کمک آن می‌توان وجود جهش در رشته‌ها که به علت تکامل رخ می‌دهد را بیان نمود. فلسفه‌ی وجود شکاف در همترازی این است که امکان تطبیق عناصر مشابه فراهم شده و امتیاز و در نتیجه دقت همترازی افزایش یابد.

روش برنامه‌نویسی پویا با الگوریتم نیدلمن-وانچ می‌تواند در همترازی سراسری و با الگوریتم اسمیت واترمن [۶] در همترازی محلی استفاده شود. نیدلمن-وانچ مساله خود را از طریق پیدا کردن بیشترین شباهت حل می‌کند اما روش دیگر به حداقل رساندن فاصله بین رشته‌هاست که به وسیله لونشتین [۷] معرفی شد. پیتر سلرز در سال ۱۹۷۴ نشان داد[۸] که این دو مساله معادل هستند. الگوریتم هیرشبرگ [۹] یک الگوریتم برنامه نویسی پویا است که همترازی بهینه بین دو رشته را با فاصله Levenshtein اندازه گیری می‌کند. همان طور که قبلا گفته شد در این پژوهش به دنبال بهینه‌سازی همترازی دو رشته بتا در حالت موازی همسو و ناهمسو هستیم، بنابراین روش‌های همترازی دوبدو را بررسی می‌کنیم. این روش‌های را می‌توان به سه دسته‌ی روش‌های ماتریس نقطه، کلمه و روش‌های مبتنی بر برنامه‌نویسی پویا تقسیم‌بندی کرد.

رویکرد ماتریس–نقطه، که یک خانواده از هم‌ترازی‌ها برای مناطق هر توالی تولید می‌کند، از نظر مقداری و مفهومی ساده است، با این وجود برای آنالیز در یک مقیاس بزرگ، زمان بر است. در غیاب نویز، می‌توان به سادگی برخی از ویژگی‌های توالی را مانند درج‌ها، حذف‌ها، تکرارها، یا تکرارهای معکوس شده به صورت دیداری در یک نمودار ماتریس-نقطه‌ای تشخیص داد. برای ساخت یک طرح ماتریس-نقطه، دو توالی در امتداد بالاترین سطر و چپ‌ترین ستون از یک ماتریس دو بعدی نوشته می‌شود و سپس در هر مکانی که دو کاراکتر، سطر و ستون آن‌ها با یکدیگر منطبق بود، یک نقطه گذاشته می‌شود. بعضی از پیاده‌سازی‌های این روش، اندازه و شدت هر نقطه را بسته به درجه تشابه دو کاراکتر تغییر می‌دهند تا جانشینی‌های حفظ شده را معین کنند. در این روش، نمودار نقطه‌ای توالی‌های بسیار نزدیک به هم، به صورت یک خط در امتداد قطر اصلی ماتریس ظاهر می‌شود. مشکلات نمودارهای نقطه‌ای، به عنوان یک تکنیک نمایش اطلاعات، شامل مواردی چون: نویز، کمبود وضوح، عدم درک مستقیم وشهودی، و دشواری استخراج خلاصه آماری جورشده‌ها و مکان‌های جور روی دو توالی است. همچنین فضای هدر رفته بیشتری در جاییکه اطلاعات جورها در طول قطر تکثیرشده‌اند و اکثر فضای نمودار خالی است یا توسط نویز اشغال شده است، وجود دارد؛ و نهایتا، نمودارهای نقطه‌ای برای دو توالی محدود شده‌اند. هیچ‌کدام از این محدودیت‌ها در دیاگرام‌های هم‌ترازی Miropeats وجود ندارد اما آن‌ها نیز عیب‌های خاص خود را دارند.

روش های کلمه که به عنوان روش‌های k-tuple نیز شناخته می‌شوند، روش‌های ابتکاری هستند که پیدا کردن همترازی بهینه را تضمین نمی‌کنند اما به طور قابل توجهی از برنامه‌نویسی پویا کارآمدتر هستند. این روش ها بخصوص در جستجوی بانک‌های داده‌ای بزرگ مفید می‌باشند، جایی که بخش بزرگی از رشته نامزدهای احتمالی تطابق قابل توجهی با رشته مورد نظر ندارند. روش‌های کلمه بیشتر به دلیل پیاده‌سازی در ابزار جستجوی پایگاه‌داده FASTA و خانواده BLAST شناخته شده اند. این روش‌ها بیشتر برای جستجوی بانک داده‌ای و همچنین همترازی محلی مورد استفاده قرار می‌گیرند.

اما مهم‌ترین روش‌های همترازی، روش‌های مبتنی بر برنامه‌نویسی پویا هستند. یکی از محبوب‌ترین این روش‌ها، روش نیدلمن-وانچ [۵] است که در آن از یک ماتریس امتیازات بهره می‌برد. این الگوریتم در دو مرحله بهترین همترازی بین دو رشته را پیدا می‌کند. در گام اول که گام رو به جلو نامیده می‌شود با استفاده از امتیازات همترازی بین دو کاراکتر، ماتریس امتیازات محاسبه می‌شود و در گام عقب‌گرد بهترین همترازی یافت می‌شود. چنانچه طول رشته اول و دوم به ترتیب m و n باشد، پیچیدگی این روش $O(mn)$ خواهد بود.

الگوریتم اسمیت-واترمن [۶] برای انجام دادن یک همترازی محلی به کار گرفته می‌شود و برای مشخص کردن مناطق مشابه

بین دو توالی اسید نوکلئیک یا پروتین استفاده می‌شود. به جای در نظر گرفتن تمام توالی این الگوریتم سعی می‌کند که با در نظر گرفتن بخش‌های مختلف با همه طولهای ممکن میزان شباهت را بهینه کند. این روش در تعریف و استفاده از ماتریس امتیازات شباهت بسیار زیادی به نیدلمن-وانچ دارد. تفاوتی که با الگوریتم نیدلمن-وانچ دارد این است که در ماتریس امتیازدهی آن مقادیر منفی با صفر جایگزین می‌شوند.

الگوریتم هریشبرگ [۹] نیز در واقع نسخه‌ای از الگوریتم نیدلمن-وانچ است که با تقسیم و حل مسئله را حل می‌کند و مصرف حافظه بهتری دارد. در واقع مصرف حافظه این روش بر خلاف نیدلمن-وانچ که از مرتبه $O(mn)$ است، به صورت خطی می‌باشد.

از آنجا که برای حل مسئله‌ی تعیین صفحات بتای یک پروتئین، تمامی رشته‌های آن بصورت دوبدو می‌بایست همتراز شوند، به نظر می‌رسد موازی‌سازی بتواند ایده‌ی مناسبی باشد. ساختار الگوریتم هریشبرگ به نحوی است که می‌توان مسئله را به سادگی به قسمت‌های مختلفی شکست. در فصل سوم روش حل مسئله با برنامه‌نویسی OpenMP شرح داده خواهد شد.

۳- روش پیشنهادی

همانطور که دیدیم، محاسبه ماتریس امتیازدهی پیچیدگی زمانی و فضایی بالایی از مرتبه $O(n^2)$ دارد. بنابراین یکی از کارهای ما در این مقاله این است که ابتدا ماتریس امتیازدهی را بهینه کنیم. الگوریتم هیرشبرگ در واقع نسخه‌ای از الگوریتم نیدلمن-وانچ است که با تقسیم و حل مسئله را حل می‌کند و مصرف حافظه بهتری دارد. همین ساختار درختی الگوریتم هیرشبرگ امکان اجرای موازی بخش هایی از آن را فراهم می‌کند. گام دوم کار ما موازی‌سازی الگوریتم هیرشبرگ است.

۳-۱ برنامه‌نویسی OpenMP

OpenMP یک رابط برنامه‌نویسی است که از برنامه‌نویسی چندسکویی و چندپردازشی با حافظه اشتراکی در زبان‌های C، C++ و فرترن روی اکثر معماری‌های پردازنده و سیستم عامل‌های گوناگون نظیر لینوکس، ویندوز، مک اواس، سولاریس، AIX و HP-UX پشتیبانی می‌کند.Open MP از یک مدل پرتابل و مقیاس-پذیر که به برنامه‌نویس یک رابط کاربری ساده و انعطاف‌پذیر برای توسعه برنامه‌های موازی برای تبدیل کامپیوترهای دسکتاپ به ابرکامپیوترها ارائه می‌دهد، استفاده می‌کند. یک الگوریتم ساخته شده با مدل ترکیبی برنامه‌نویسی موازی می‌تواند بر روی یک خوشه کامپیوتری با استفاده از هر دو برنامه نویسی Open MP و MPI اجرا شود، به طوری که Open MP برای موازی سازی گره‌ها (چند هسته) استفاده شود در حالی که MPI برای موازی سازی روابط بین گره‌ها استفاده می‌شود.

۳-۲ روش کار

روش کار ما به این صورت است که ابتدا با موازی کردن ماتریس امتیازدهی و اجرای همزمان آن بر روی چند پردازنده زمان محاسبه ماتریس امتیازدهی را کاهش می‌دهیم. در مرحله بعدی الگوریتم هیرشبرگ که به صورت تقسیم و حل مساله را حل می-کند موازی کرده و در اجرای آن از ماتریس امتیازدهی موازی که در مرحله اول طراحی کردیم استفاده خواهیم کرد.

۳-۲-۱ موازی کردن ماتریس امتیازدهی

امتیاز هر سلول در ماتریس امتیازدهی از مقایسه امتیازات همسایه سمت چپ، بالا و بالا-سمت چپ هر درایه و افزودن امتیاز مناسب برای حالت تطابق، عدم تطابق و یا پرش به دست می‌آید. امتیازات را برای هر یک از سه حالت فوق محاسبه می‌کنیم. نتیجه امتیاز برای هر درایه بالاترین امتیاز سه حالت ذکر شده می‌باشد. شکل ۵ الگوریتم موازی کردن ماتریس را نمایش می‌دهد. با اجرای همزمان دو نخ موازی طراحی شده که در شکل ۵ قابل مشاهده است روی دو پردازنده زمان اجرا تقسیم بر دو شده بنابراین زمان اجرا کاهش قابل ملاحظه‌ای خواهد داشت.

۳-۲-۲ موازی کردن الگوریتم هریشبرگ

با توجه به ساختار الگوریتم هیرشبرگ امکان موازی‌سازی آن در قسمت فراخوانی توابع تقسیم و غلبه وجود دارد. الگوریتم پیشنهادی برای این کار ما در شکل ۶ نشان داده شده است. در فصل آینده به بررسی کارایی روش پیشنهاد شده خواهیم پرداخت.

۴- نتایج

در پیاده‌سازی برای مقایسه زمان از الگوریتم نیدلمن-وانچ، هیرشبرگ و الگوریتم پیشنهادی استفاده می‌کنیم. الگوریتم‌ها را برای دو رشته، چند رشته و بانک داده‌ای مورد استفاده تکرار می-کنیم. در این فصل ابتدا بانک داده‌ای مورد استفاده را تشریح می-کنیم، سپس نتایج پیاده‌سازی روش‌ها بوسیله معیار زمان با یکدیگر مقایسه می‌شوند.

```
function NFScore(X,Y)
F(۰,۰) = ۰
for i=۱ to length(X)
  F(i,۰) = F(i-۱,۰) + Del(Xi)
for j=۱ to length(Y)
  F(۰,j) = F(۰,j-۱) + Ins(Yj)
for k=۱ to min(length(X),length(Y))
thread ۱ :
  for i=k to length(X)
     scoreSub = F(i-۱,k-۱) + Sub(Xi,
Yk)
     scoreDel = F(i-۱,k) + Del(Xi)
     scoreIns = F(i,k-۱) + Ins(Yk)
     F(i,k) = max(scoreSub,
scoreDel, scoreIns)
  end
thread ۲ :
  for j=k+۱ to length(Y)
     scoreSub = F(k-۱,j-۱) + Sub(Xk,
Yj)
     scoreDel = F(k-۱,j) + Del(Xk)
     scoreIns = F(k,j-۱) + Ins(Yj)
     F(k,j) = max(scoreSub,
scoreDel, scoreIns)
  end
end
for j=۰ to length(Y)
  LastLine(j) = F(length(X),j)
```

شکل ۵: الگوریتم پیشنهادی برای موازی کردن ماتریس همترازی.

۴-۱ بانک داده‌ای

بانک داده‌ای استفاده شده در این پژوهش، BetaSheet۹۱۶ [۱۰] می‌باشد. این بانک داده‌ای از پروتئین‌ها استخراج گردیده است. در این بانک داده‌ای زنجیره‌های شامل اسیدهای آمینه ناشناخته یا غیر استاندارد، تعلیق‌های زیرساخت و زنجیره‌های با طول کمتر از ۵۰ اسید آمینه درنظر گرفته نشده‌اند و در آن از استاندارد DSSP جهت اختصاص ساختار دوم و مقادیر دسترسی حل شده به هر اسیدآمینه استفاده گردیده است. بانک داده‌ای نهایی که β-sheet ۹۱۶ نام دارد شامل ۹۱۶ زنجیره محتوی ۱۸۷۵۱۶ اسیدآمینه می‌باشد. ۲۶ درصد از این تعداد یعنی ۴۸۹۹۶ اسیدآمینه موجود در رشته‌های بتا می‌باشند که ۳۱۶۳۸ تای آن‌ها در جفت‌شدن اسیدهای آمینه شرکت دارند. این بانک داده شامل ۱۰۷۴۵ رشته بتا با میانگین طول ۴/۶ اسیدآمینه و تعداد ۸۱۷۲ زوج رشته بتا می‌باشد که شامل ۴۵۱۹ جفت موازی ناهمسو و ۲۲۱۴ جفت موازی همسو و ۱۴۳۹ جفت محتوی پل‌های بتای مجزا می‌باشد که این جفت رشته‌ها از ۲۵۳۳ صفحه بتا می‌باشند.

```
function PHirschberg(X,Y)
   Z = ""
   W = ""
   if length(X) == ۰
     for i=۱ to length(Y)
       Z = Z + '-'
       W = W + Yi
     end
   else if length(Y) == ۰
     for i=۱ to length(X)
       Z = Z + Xi
       W = W + '-'
     end
   else if length(X) == ۱ or
length(Y) == ۱
     (Z,W) = NW(X,Y)
   else
     xlen = length(X)
     xmid = length(X)/۲
     ylen = length(Y)

thread ۱
     ScoreL = NFScore(X۱:xmid, Y)
Thread ۲
     ScoreR =
NFScore(rev(Xxmid+۱:xlen), rev(Y))

     ymid = arg max ScoreL +
rev(ScoreR)

thread ۳
     t۱ = PHirschberg(X۱:xmid,
y۱:ymid)
thread ۴
     t۲ = PHirschberg(Xxmid+۱:xlen,
Yymid+۱:ylen)
```

شکل ۶: الگوریتم موازی کردن الگوریتم هریشبرگ.

۴-۲ بررسی و مقایسه نتایج

برای پیاده‌سازی و اجرای الگوریتم ها از یک سیستم با پردازنده چهار هسته‌ای استفاده شد که معماری چند نخی توسط سیستم عامل بر روی پردازنده اجرا می‌شود و امکان دارد بخش‌هایی از برنامه توسط سیستم عامل موازی اجرا شود. پس بهتر است از سیستمی استفاده کرد که دارای پردازنده‌ی دو هسته ای باشد و سیستم عامل آن به صورت خودکار عملیات بهینه‌سازی را انجام ندهد. ما در اینجا به کمک کدهای Open MP تا حد امکان جلوی موازی‌سازی خودکار را گرفته‌ایم. مقایسه را برای تمام رشته‌های بتای موجود در ۹۱۶ پروتئین بانک داده‌ای مورد استفاده انجام می‌دهیم. تعداد رشته‌های بتای موجود برابر ۱۰۷۴۵ رشته و تعداد

همترازی‌های انجام شده برابر با ۱۴۶۲۹۶ همترازی می‌باشد. زمان لازم برای همترازی رشته‌های موجود در این پروتئین در روش نیدلمن-وانچ برابر ۴۶۸ ثانیه، در روش هیرشبرگ برابر ۱۲۰ ثانیه و در روش پیشنهادی برابر با ۵۵ ثانیه می‌باشد. نتایج مقایسه روش‌های مختلف در جدول ۱ نشان داده شده است.

جدول ۱: مقایسه زمان اجرای روش پیشنهادی

روش	Needleman–Wunsch	Hirschberg	روش پیشنهادی
زمان	468s	120s	55s

۵- نتیجه‌گیری

همترازی به عنوان یک گام میانی برای کل مساله پیش‌بینی ساختار از اهمیت بالایی برخوردار است. با توجه به اینکه همترازی رشته‌های بتا یک گام میانی جهت پیش‌بینی ساختار پروتئین‌ها می‌باشد، کاهش زمان در این مرحله منجر به کاهش زمان کل مساله می‌شود. در مسائل با اندازه ورودی بسیار بزرگ یکی از مهم-ترین چالش‌ها بحث زمان اجرا می‌باشد. بنابراین یکی از چالش‌های مهم در مساله همترازی بالابودن زمان مراحل اجرای آن است. در این پژوهش با یافتن الگوریتم هیرشبرگ که قابلیت موازی‌سازی داشت فضای مساله را به صورت خطی بهینه نگه داشتیم. سپس با موازی‌سازی الگوریتم محاسبه ماتریس امتیازدهی و اصل الگوریتم هیرشبرگ و استفاده از برنامه‌نویسی OpenMP زمان مساله همترازی را به میزان قابل توجهی کاهش دادیم. به عنوان کار آینده می‌توان گفت در ماتریس امتیازدهی که مورد بررسی قرارگرفت برخی داده‌های اضافی وجود دارد که در عملیات همترازی مورد استفاده قرار نمی‌گیرد. اگر بتوان الگوریتمی ارائه داد که این داده-های اضافی را محاسبه و نگهداری نکند هم در زمان و هم در فضای مساله بهبود اتفاق می‌افتد.

مراجع

[1] L. Brocchieri, and S. Karlin, *Protein length in eukaryotic and prokaryotic proteomes*, Nucleic Acids Res, vol. ۳۳, pp. ۳۳۹۰-۴۰۰, ۲۰۰۵.

[2] W. Kabsch and C. Sander, *Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features*, vol. ۲۲, pp. ۲۵۷۷-۲۶۳۷, ۱۹۸۳.

[3] M. Sabzekar, M. Naghibzadeh and J. Sadri, Efficient dynamic programming algorithm with prior knowledge for protein β-strand alignment, Journal of Theoretical Biology, vol. ۴۱۷, pp. ۴۳-۵۰, ۲۰۱۷.

[4] V. Polyanovsky, M. Roytberg and V. Tumanyan, *Comparative analysis of the quality of a global algorithm and a local algorithm for alignment of two sequences*, Algorithms Mol Biol, vol. ۶, pp. ۲۵, ۲۰۱۱.

[5] S. Needleman and C. Wunsch, *A general method applicable to the search for similarities in the amino acid sequence of two proteins*, J Mol Biol, vol. ۴۸, p. ۴۴۳-۵۳, ۱۹۷۰.

[6] T. Smith and M. Waterman, *Identification of common molecular subsequences*, J Mol Biol, vol. ۱۴۷, pp. ۱۹۵-۷, ۱۹۸۱.

[7] V. Levenshtein, *Binary codes capable of correcting deletions, insertions, and reversals*, pp. ۷۰۷-۷۱۰, ۱۹۶۶.

[8] P.H. Sellers, *On the Theory and Computation of Evolutionary Distances*, SIAM Journal on Applied Mathematics, vol. ۲۶(۴): p. ۷۸۷-۷۹۳, ۱۹۷۴.

[9] Hirschberg, D.S., A linear space algorithm for computing maximal common subsequences. Commun. ACM, ۱۹۷۵. ۱۸(۶): p. ۳۴۱-۳۴۳.

[10] "BetaSheet۹۱۶ Set," http://www.ics.uci.edu/~baldig/betasheet_data.html, ۲۰۰۹Burkoff, N.S., Várnai, C., Wild, D.L., ۲۰۱۳. Predicting protein β-sheet contacts using a maximum entropy-based correlated mutation measure. Bioinformatics ۲۹ (۵),۵۸۰ –۵۸۷.

[11]

[12] A. Author ۱ and B. Author ۲, *Title of the Book*. John Wiley & Sons, pp. ۱۰۰-۱۰۵, ۲۰۰۲.

[13] A. Author ۱ and B. Author ۲, "Title of the conference paper," *Proc. Int. Conf. on Power System Reliability*. Singapore, pp. ۱۰۰-۱۰۵, ۱۹۹۹.

[14] A. Author ۱ and B. Author ۲, "Title of the journal paper" *IEEE Trans. Antennas and Propagation*, Vol. ۵۵, No. ۱, pp. ۱۲-۲۳, ۲۰۰۷.