
بسم الله الرحمن الرحیم

دانشگاه صنعتی اصفهان

دانشکده برق و کامپیوتر

بررسی الگوریتم‌های موجود برای مسئله کوتاه ترین ابرتوالی مشترک و ارائه الگوریتمی بهبود یافته برای آن

پایان‌نامه کارشناسی ارشد مهندسی کامپیوتر- هوش مصنوعی

فاطمه بحری

استاد راهنما

دکتر سید رسول موسوی

۱۳۸۹

دانشگاه صنعتی اصفهان

دانشکده برق و کامپیوتر

پایان‌نامه‌ی کارشناسی ارشد رشته‌ی مهندسی کامپیوتر- هوش مصنوعی خانم فاطمه بحری تحت عنوان

بررسی الگوریتم‌های موجود برای مسئله کوتاه ترین ابرتوالی مشترک و ارائه الگوریتمی بهبود یافته برای آن

در تاریخ ۱۳۸۹/۱۲/۲۵ توسط کمیته‌ی تخصصی زیر مورد بررسی و تصویب نهایی قرار گرفت.

۱- استاد راهنمای پایان‌نامه	دکتر سید رسول موسوی
۲- استاد داور	دکتر مازیار پالهنگ
۳- استاد داور	دکتر عبدالرضا میرزائی
سرپرست تحصیلات تکمیلی دانشکده	دکتر سید محمود مدرس هاشمی

نخستین سپاس به پیشگاه حضرت دوست، که هر چه داریم از اوست.
صمیمانه‌ترین سپاس‌ها را تقدیم می‌کنم به همسر مهربانم به خاطر همدلی‌ها، همراهی‌ها و کمک‌های بی دریغش ، به پدر و مادر عزیزم که تا ابد وام‌دار محبتشان هستم و به برادران بسیار عزیزم، علیرضا و امیرمحمد.
از استاد راهنمای بزرگوارم جناب آقای دکتر موسوی که همواره در مراحل انجام پژوهش از راهنمایی‌های ارزنده ایشان برخوردار بوده‌ام،کمال تشکر و سپاس را دارم.
از اساتید گران‌قدری که در دوران تحصیل افتخار شاگردی‌شان را داشته-ام، به‌ویژه جناب آقای دکتر صدری متشکرم.
از همه دوستانم که در طول مقطع کارشناسی ارشد در محضرشان بودم و این مدت از زندگیم را به دورانی دوست‌داشتنی تبدیل کردند، به‌ویژه خانم‌ها مهندس عموزادی، پاکیزه، احمدی، طباطبا، رزاقی، ایمانی، بابائی و رمضانی کمال تشکر را دارم.
و در پایان از همدلی دوستان همیشه همراهم فاطی، ملیحه، شکوه، فرزان، نیلوفر،مهناز، مطهره و بهناز سپاسگزارم.

کلیه‌ی حقوق مادی مترتب بر نتایج مطالعات، ابتکارات و نوآوری‌های ناشی از تحقیق موضوع این پایان‌نامه (رساله) متعلق به دانشگاه صنعتی اصفهان است.

تقدیم به پدر و مادر عزیزتر از جانم

به پاس مهر بی‌دریغشان

و تقدیم به

مهربانم، همراهم، عزیزم

همسرم

محمدم

فهرست مطالب

عنوان **صفحه**

فصل ششم: نتیجه‌گیری و پیشنهادات

ده

چکیده

مسئله کوتاه‌ترین ابرتوالی مشترک (SCS) از مسائل مهم بهینه‌سازی ترکیبیاتی است. مجموعه‌ای از رشته‌ها به عنوان ورودی به مسئله SCS داده می‌شود و این مسئله به دنبال رشته ای می‌گردد که ابرتوالی تمام رشته‌ها بوده و دارای کوتاه‌ترین طول ممکن باشد. ابرتوالی یک رشته، از درج صفر یا تعداد بیشتری نماد در هر مکان از رشته مورد نظر به دست می‌آید. این مسئله در زمینه‌های مختلفی از جمله فشرده‌سازی داده، بهینه‌سازی پرسش در پایگاه داده ها، طرح‌ریزی و زیست‌اطلاعات کاربرد دارد. یکی از کاربردهای مهم این مسئله در زیست‌اطلاعات، کاهش زمان، هزینه و خطا در ساخت یکی از انواع ریزآرایه‌های DNA می باشد. ریزآرایه‌های DNA ابزارهای مفیدی هستند که در دسته بندی ژن‌ها، تشخیص ژن‌ها، تصویر سازی بیان ژنی و آشکارسازی SNP، کاربرد دارند.

تا به حال الگوریتم‌های دقیق برنامه‌نویسی‌پویا و شاخه‌وحد برای حل مسئله SCS ارائه شده‌اند، اما این مسئله هنگامی که تعداد رشته ها بیشتر از ۲ باشد یک مسئله NP-Hard می باشد، بنابراین تاکنون راه حل بهینه ای که در زمان چندجمله ای قابل اجرا باشد برای آن ارائه نشده است و در عمل الگوریتم های دقیق برای نمونه های بزرگ این مسئله قابل اجرا نیستند. به همین دلیل، اکثر الگوریتم-های موجود برای حل این مسئله غیر دقیق می‌باشند، بدین معنی که این الگوریتم‌ها از شرط بهینگی پاسخ چشم‌پوشی کرده و سعی می کنند که جوابی نزدیک به بهینه در زمانی معقول به دست آورند. تا به حال الگوریتم های غیردقیق مختلفی از انواع الگوریتم های تقریبی، مکاشفه ای ساده و فرا مکاشفه ای برای مسئله SCS ارائه شده اند. الگوریتم های تقریبی الگوریتم هایی هستند که جواب حاصل از آنها نسبت به جواب بهینه از حدی که با نسبت تقریب مشخص می شود، بدتر نمی‌باشد. الگوریتم های مکاشفه‌ای نیز، به الگوریتم‌هایی گفته می شود که بر مبنای یک مکاشفه عمل می کنند. این الگوریتم ها به دو دسته مکاشفه‌ای ساده و فرامکاشفه‌ای تقسیم می شوند. برای حل مسئله کوتاه‌ترین ابرتوالی مشترک تا به حال از الگوریتم‌های مکاشفه‌ای ساده و فرامکاشفه ای مختلفی استفاده شده است. تعدادی از این الگوریتم‌های مکاشفه‌ای به طور خاص برای رشته های DNA طراحی شده اند.

در این پایان نامه یک الگوریتم دقیق A* ارائه شده و برای آن توابع مکاشفه ای مختلفی بررسی شده است و در نهایت یک تابع مکاشفه‌ای ترکیبی ارائه شده که با تخمین حدی دقیق‌تر، از سایر توابع مکاشفه‌ای بهتر عمل می کند. در این الگوریتم برای کاهش فضای جستجو و در نتیجه کاهش زمان اجرا از راهبرد شاخه و حد و همچنین ایده هرس غالبانه استفاده شده است. با مقایسه این الگوریتم با دیگر الگوریتم‌های دقیق موجود، مشخص شد که الگوریتم ارائه شده منجر به کاهش قابل ملاحظه‌ای در فضای جستجو شده و دارای زمان اجرای بهتری می باشد.

همچنین یک الگوریتم مکاشفه ای به همراه یک رویه پس پردازش در این پایان نامه ارائه شده است. این الگوریتم مکاشفه ای یک الگوریتم جستجوی پرتوی محلی است که مکاشفه‌ای مبتنی بر روابط بازگشتی را به کار گرفته و با استفاده از برنامه‌نویسی پویا یک آرایه از مقادیر احتمالاتی به صورت برنامه‌نویسی‌پویا تولید می‌کند که برای محاسبه سریع‌تر مقادیر مکاشفه‌ای مورد استفاده قرار می-گیرد. همچنین این الگوریتم با استفاده از ایده هرس غالبانه فضای جستجو را هرس می‌کند. این الگوریتم با سه الگوریتم برتر اخیر بر روی رشته‌های تصادفی و واقعی مقایسه شده و نتایجی بهتر هم از نظر زمان اجرا و هم طول جواب به دست آورده است. رویه پس پردازش ارائه شده را نیز می توان بر روی هر پاسخ غیر بهینه از یک نمونه مسئله SCS، اجرا کرد. این رویه با کاهش طول جواب موجود در حلقه‌ای تکرارشونده، جواب کوتاه‌تری که همچنان ابرتوالی مشترک رشته های ورودی می‌باشد، ارائه می‌کند. نتایج به دست آمده مشخص می‌کند که این رویه پس‌پردازش نسبت به رویه پس‌پردازش موجود، در زمان اجرای یکسان، دارای کیفیت جواب بهتری می باشد.

کلمات کلیدی: ۱- کوتاه‌ترین ابرتوالی مشترک ۲- الگوریتم A* ۳- الگوریتم‌های مکاشفه‌ای ۴- جستجوی پرتوی محلی

فصل اول
مقدمه

۱-۱ مقدمه

مسئله کوتاه ترین ابر توالی مشترک[1](SCS) از جمله مسائل مهم بهینه سازی ترکیباتی[2] است. این مسئله بایستی کوتاه ترین رشته ای را بیابد که ابرتوالی هر کدام از رشته های ورودی باشد. ابرتوالی رشته x رشته ای است که با اضافه کردن صفر یا تعداد بیشتری نماد، در هر مکان دلخواه از رشته x به دست می آید. برای مثال کوتاه ترین ابرتوالی مشترک سه رشته $CGATG$، $TCGCA$ و $GATTC$، رشته $TCGCATTGC$ می باشد.

مسئله SCS برای ۲ رشته ورودی در زمان چند جمله ای به صورت بهینه [۱] حل می شود، اما هنگامی که تعداد رشته های ورودی بیشتر از ۲ باشد، این مسئله NP-Hard می باشد[۲-۴]. این مسئله حتی در حالت های محدودشده ای که اندازه الفبا ۲ می باشد و یا طول همه رشته ها ۲ می باشد، نیز NP-Hard باقی می ماند[۲-۷]. تا به حال الگوریتم های بهینه ای که برای مسائل NP-Hard ارائه شده اند، بر حسب اندازه ورودی مسئله، دارای زمان اجرایی نمایی می باشند [۸] که این موضوع در مورد مسئله SCS نیز صادق است. الگوریتم های دقیقی که تا به حال برای مسئله SCS ارائه شده اند، الگوریتم برنامه نویسی پویا [۴, ۹] و الگوریتم های شاخه و حد، می باشند. برای حل این مسئله، الگوریتم های غیر دقیق متعددی از انواع تقریبی، مکاشفه ای ساده و فرامکاشفه ای ارائه شده است که در فصل دوم به تفصیل، مورد مطالعه قرار می گیرند.

---

[1] Shortest Common Supersequence
[2] Combinatorial optimization

۲-۱ تعریف مسئله SCS

اگر s یک رشته باشد، طول آن با $m = |s|$ نمایش داده می شود. $s(k)$ برای نمایش k امین نماد رشته s استفاده می شود، که k مقداری بین ۱ و m می باشد. رشته ای که از نماد j ام تا k ام رشته s به دست می آید به طوری که $j \le k$ ، با نماد $s(j:k)$ نمایش داده می شود، بنابر این برای نمایش رشته شامل k نماد ابتدایی s، از نماد $s(1:k)$ استفاده می شود. برای نمایش الحاق دو رشته از نماد + استفاده می شود، برای مثال رشته x از الحاق دو رشته x_1 و x_2 به دست می آید: $x = x_1 + x_2$.

فرض کنیم s_1 و s_2 دو رشته باشند و $A_1 = \{i \mid i \in \mathrm{N}, i \le |s_1|\}$ و $A_2 = \{i \mid i \in \mathrm{N}, i \le |s_2|\}$ که N مجموعه اعداد صحیح بزرگتر از صفر است. رشته s_2 ابرتوالی رشته s_1 است یا به بیان نمادین $s_1 \prec s_2$، اگر یک تابع یک به‌یک همانند g از A_1 به A_2 وجود داشته باشد به طوری که :

$$\begin{aligned} &\forall k \in A_1, s_1(k) = s_2(g(k)) \\ &\forall k, k' \in A_1, k < k' \Rightarrow g(k) < g(k') \end{aligned} \qquad ۱-۱$$

تابع g لزوما یکتا نیست و احتمال دارد چندین تابع که این شروط را برآورده کند، وجود داشته باشد. هر رشته، ابر توالی رشته تهی -که رشته ای به طول صفر است- محسوب می شود. همچنین در صورتی که رشته s_2 ابرتوالی رشته s_1 باشد، می توان گفت که رشته s_1 زیرتوالی رشته s_2 می باشد.

فرض می شود که x یک رشته و S یک مجموعه غیر تهی از رشته ها می باشد. اگر $\forall s_i \in S, s_i \prec x$، آن‌گاه x ابرتوالی مجموعه رشته S است یا به بیانی نمادین $S \prec x$. مسئله کوچکترین ابرتوالی مشترک به این صورت تعریف می شود؛ با توجه به یک مجموعه رشته S داده شده، باید رشته x با طول کمینه یافت شود به طوری که $S \prec x$. اگر تنها یک رشته به عنوان ورودی داده شود، منظور این است که S دارای فقط یک رشته است. الفبایی که رشته های ورودی بر روی آن تعریف شده اند با Σ نمایش داده می شود؛ فرض می کنیم که $|\Sigma| > 1$. از n برای نمایش تعداد رشته های ورودی استفاده می کنیم، به عبارتی دیگر $n = |S|$. از آنجایی که برای $n = 2$ مسئله SCS در زمان خطی حل می شود، فرض می کنیم که $n > 2$ می باشد.

هر کدام از رشته های ورودی مسئله را با حرف s کوچک نشان می دهیم که با زیرنویس ۱ تا n مشخص شده اند یعنی $S = \{s_1, ..., s_n\}$. از نماد m_i برای نمایش طول s_i استفاده می شود یعنی $m_i = |s_i|$. فرض می شود که $m_i > 0, i = 1, ..., n$ یعنی هر رشته ورودی حداقل شامل یک نماد می باشد. نماد m نشان دهنده مقدار بیشینه طول رشته های ورودی است یا به عبارتی $m = \max\{m_i, i = 1, ..., n\}$. از نماد x (شاید همراه با زیرنویس) برای نمایش یک جواب نامزد[1] استفاده می شود. جواب نامزد x، جواب نهایی[1] نامیده می شود

[1] Candidate solution

اگر x ابرتوالی تمام رشته های ورودی باشد، در غیر این صورت جواب جزئی[۲] نامیده می شود. یک جواب نامزد نهایی در صورتی بهینه است که هیچ جواب نامزد نهایی دیگری با طولی کوتاه تر وجود نداشته باشد.

اگر x یک جواب نامزد جزئی باشد، $p_i(x)$ نشان دهنده ماکزیمم عدد صحیح ممکن k است به طوری که $s_i(1:k) \prec x$، به بیانی دیگر $s_i(1:k)$ نشان دهنده بلندترین زیررشته از ابتدای رشته s_i است که توسط x پوشش داده شده است و $p_i(x)$ نشانگر طول این رشته است. رشته ای که از حذف کردن $p_i(x)$ کاراکتر اول رشته s_i به دست می آید، $r_i(x)$ یا به عبارتی دیگر باقیمانده رشته s_i نامیده می شود. $R(x)$ مجموعه تمام باقیمانده های رشته‌های ورودی است، یعنی $\{r_i(x), i=1,...,n\}$. شکل ۱-۱ مثالی برای مشخص شدن نماد $p_i(x)$ و $r_i(x)$ ارائه می‌کند.

x = b c a b

s_1 = b c a $\underbrace{\text{d c f d c}}_{r_1(x)}$ s_2 = c a b $\underbrace{\text{a a f d c}}_{r_2(x)}$ s_3 = b a $\underbrace{\text{c d d f c d}}_{r_3(x)}$

$p_1(x)=3$ $p_2(x)=3$ $p_3(x)=2$

شکل ۱-۱- مثالی از $p_i(x)$ و $r_i(x)$ برای جواب جزئی x و سه رشته ورودی s_1، s_2 و s_3

۱-۳ کاربردهای مسئله SCS

مسئله SCS در زمینه های مختلفی از جمله فشرده سازی داده[۳] [۱۰]، بهینه سازی پرسش[۴] در پایگاه داده ها [۱۱, ۱۲]، طرح‌ریزی[۵] [۱۳] و زیست‌اطلاعات[۶] [۱۴-۱۸]کاربرد دارد. زیست‌اطلاعات، شاخه‌ای از علم است که دو شاخه‌ی علوم کامپیوتر و زیست‌شناسی را به یکدیگر متصل می‌کند. یکی از کاربرد های این مسئله در زمینهٔ زیست‌اطلاعات، مربوط به تولید نوعی ریزآرایهٔ DNA[۷] می باشد. ریزآرایه های DNA شامل هزاران مکان هستند، که هر کدام از آن ها حاوی تعداد زیادی از یک توالی خاص DNA است و در دسته بندی ژن ها[۸]، تصویر سازی بیان ژنی[۹]، آشکارسازی SNP[۱۰]، تشخیص ژن ها[۱۱] و تشخیص بیماری ها کاربرد دارند[۱۹, ۲۰]. ریزآرایه

---

[1] Final solution
[2] Partial solution
[3] Data compression
[4] Query optimization
[5] Planning
[6] Bioinformatics
[7] DNA microarray
[8] Gene clustering
[9] Gene expression profiling
[10] Single Nucleotide Polymorphism detection
[11] Gene identification

هـای DNA، بـه دو نـوع ریزآرایـهcDNA و ریزآرایـه اولیگونوکلئوتیـد[1] تقسـیم مـی شـوند. ریزآرایـه هـای اولیگونوکلئوتید هزینه ساخت بیشتری دارند، اما در آن ها پیوند متقاطع[2] کمتر رخ مـی دهـد [۲۱] و بـه همـین دلیل دقیق تر می باشند. این نوع ریزآرایه برای تشخیص چندریختی[3] می تواند به کار رود[۱۹].

ریزآرایه های اولیگونوکلئوتید معمولا به روش لیتوگرافی نوری[4] تولید می شوند[۲۲]. در این روش، در هر قدم، یک نوکلئوتید به بعضی از مکان های ریزآرایه اضافه می شود و به انتهای توالی آن مکان، الحـاق مـی شـود. نوکلئوتید واحد سازنده اسیدهای نوکلئیک می باشد و بنا به نوعش با یکی از نمادهای A^{5}، C^{6}، G^{7} یا T^{8} نمایش داده می شود. برای اضافه کردن نوکلئوتید در هر دور، یک نقاب[9] برای محافظـت از مکـان هـایی از ریزآرایـه کـه نبایستی اشعه فرابنفش را دریافت کنند، استفاده می شود. اشعه فرابنفش در مکان های بدون نقاب نفـوذ کـرده و اولیگونوکلئوتید را در آنجا فعال می کند[۱۹]. هزینه ساخت نقاب های مورد استفاده در این روش بالا می باشد و بنابراین کاهش مراحل تولید ریزآرایه، به کاهش هزینه ساخت و البته کاهش زمان مورد نیاز برای ساخت ریزآرایه منجر می شود. به دلیل این که پوشاندن مکان های ریزآرایه با نقاب، بدون خطا نمی باشد، با کاهش مراحـل مـی توان به کاهش خطا در ساخت ریزآرایه نیز کمک کرد. مسئله یـافتن کمتـرین مراحـل مـورد نیـاز بـرای سـاخت ریزآرایه ای از تعدادی توالی DNA را می توان با مسئله SCS مدل کرد، به این صورت که هـر تـوالی DNA بـه عنوان یک رشته ورودی در نظر گرفته شود. کوتاه ترین ابر توالی مشترک ایـن رشـته هـا، ترتیـب اضـافه کـردن نوکلئوتیدها را (که منجر به کمترین مراحل ساخت می شود) نمایش می دهد. بنـابراین اسـتفاده از مسـئلهٔ SCS موجب کاهش هزینه، زمان و خطا در ساخت ریزآرایه های اولیگونوکلئوتید می شود[۱۵-۱۷].

به عنوان مثالی دیگر از کاربرد SCS می توان به طراحی خط تولیدی که در آن قطعـات مختلفـی تولیـد می‌شوند، اشاره کرد. برای تولید یک قطعه لازم است دنباله ای از عمل ها که به نوع آن قطعه بستگی دارد بر روی آن صورت بگیرد. خط تولید نیز بایستی به صورت دنباله ای از دستگاه ها طراحی شود، کـه هـر دسـتگاه قابلیـت انجام فقط یک عمل را بر روی قطعه ها دارد. این مسئله با استفاده از مسئله SCS مدل می شود، به ایـن صـورت که دنبالهٔ عملیات مورد نیاز برای هر قطعه، یک رشته ورودی در نظر گرفتـه مـی شـود و کوتـاه تـرین ابـر تـوالی

[1] Oligonucleotide
[2] Cross hybridization
[3] Polymorphism
[4] Photolithography
[5] Adenine
[6] Cytosine
[7] Guanine
[8] Thymine
[9] Mask

مشترک این رشته‌ها، کوتاه‌ترین دنباله از دستگاه ها را مشخص می کند که تمام قطعات با این دنباله از دستگاه ها می توانند تولید شوند.

۱-۴ ساختار پایان‌نامه

در این تحقیق، یک الگوریتم دقیق A* پیشنهاد شده است و به منظور هرس فضای جستجو وکاهش زمان اجرا، در آن از راهبرد شاخه و حد و نیز هرس غالبانه[1] استفاده شده است. در این الگوریتم از توابع مکاشفه ای مختلفی استفاده شده است و در نهایت یک تابع مکاشفه ای ترکیبی ارائه شده است که نسبت به بقیه توابع مکاشفه ای نتایج بهتری به دست می آورد. الگوریتم A* مطرح شده، نسبت به سایر الگوریتم های دقیق از جمله برنامه نویسی‌پویا دارای زمان اجرای بهتری می باشد.

همچنین در تحقیق ارائه شده، یک الگوریتم غیر دقیق مکاشفه ای به نام IBS-SCS برای مسئله SCS ارائه شده است. الگوریتم IBS-SCS یک الگوریتم جستجوی پرتوی محلی[2] می باشد که از مکاشفه ای جدید مبتنی بر روابط بازگشتی و نیز هرس غالبانه برای کاهش فضای جستجو استفاده می کند. این الگوریتم در مقایسه با بهترین الگوریتم های غیر دقیق برای SCS (که الگوریتم های DR [۲۳]،PBS [۲۴] و MA-BS [۲۵] می باشند) نتایج بهتری در مدت زمان اجرای کمتر به دست می‌آورد. همچنین یک رویه پس پردازش پیشنهاد شده است که با اجرای این رویه پس پردازش بر روی جواب به دست آمده از هر الگوریتم غیر دقیق حل SCS، جواب کوتاه تری به دست می آید. رویه پس پردازش پیشنهاد شده از مرحله کاهش در الگوریتم DR [۲۳] الهام گرفته شده است و نسبت به آن در زمان اجرای یکسان، نتایج بهتری کسب می کند.

ساختار این پایان نامه بدین ترتیب است: در فصل دوم مروری بر الگوریتم هایی که تا به حال برای مسئله SCS ارائه شده اند، صورت می گیرد. سپس در فصل سوم الگوریتم A* پیشنهادی ارائه می گردد. الگوریتم مکاشفه ای IBS-SCS و رویه پس پردازش پیشنهادی در فصل چهارم به تفصیل بیان می گردند. در فصل پنجم ابتدا نتایج الگوریتم A* با الگوریتم های دقیق پیاده سازی شده مقایسه شده، سپس نتایج الگوریتم IBS-SCS با بهترین الگوریتم های موجود مقایسه شده و در پایان مقایسه ای بین نتایج رویه پس پردازش پیشنهادی و رویه کاهش الگوریتم DR صورت می گیرد. در فصل ششم نیز جمع بندی این تحقیق و پیشنهاداتی برای آینده ارائه می شود.

---

[1] Dominance pruning
[2] Local Beam Search

فصل دوم
مروری بر الگوریتم های موجود برای مسئله SCS

۲-۱ مقدمه

الگوریتم های موجود برای مسئلهSCS را می توان به دو دسته دقیق و غیر دقیق تقسیم کرد، که در این فصل مورد مطالعه قرار می گیرند. الگوریتم های دقیق، الگوریتم‌هایی هستند که مساله را به صورت دقیق حل کرده و پاسخ بهینه مسئله را می یابند. این الگوریتم‌ها بر روی نمونه‌های بزرگ از مسائل NP-Hard، قابل اجرا نمی‌باشند. به همین علت الگوریتم های دقیق برای مسئله SCS به ندرت مورد بررسی قرار گرفته اند.در بخش ۲-۲، این الگوریتم های دقیق مطالعه می شوند.

الگوریتم‌های غیردقیق با برداشتن شرط بهینگی، در زمان نسبتا مناسب، جوابی نزدیک به بهینه به دست می‌آورند. الگوریتم های غیردقیق دو دسته هستند. دسته اول الگوریتم‌های تقریبی می باشند. این الگوریتم‌ها، تضمین می‌کنند که جواب آنها نسبت به جواب بهینه از حدی که نسبت تقریب[1] نام دارد، بدتر نیست. الگوریتم های تقریبی ارائه شده برای مسئله SCS در بخش ۲-۶ مورد مطالعه قرار می گیرند. دسته دوم، الگوریتم‌های مکاشفه‌ای نام دارند. این الگوریتم‌ها، هیچ نسبت تقریبی را تضمین نمی‌کنند، اما معمولاً جواب‌های خوبی در زمانی مناسب به دست می‌آورند. الگوریتم‌های مکاشفه‌ای به دو دسته مکاشفه‌ای ساده و فرا مکاشفه‌ای تقسیم می شوند. الگوریتم‌های مکاشفه‌ای ساده معمولا از یک تابع حریصانه[2] استفاده می‌کنند و با نام حریصانه نیز شناخته

[1] Approximation ratio
[2] Greedy

می‌شوند. تا به حال تعدادی الگوریتم مکاشفه ای ساده برای مسئله SCS ارائه شده است، که از آن جمله می توان به الگوریتم های مکاشفه ای ساده ای که برای رشته های DNA طراحی شده اند، اشاره کرد. این الگـوریتم هـای مکاشفه ای ساده و مکاشفه ای مخصوص رشته های DNA به ترتیب در بخش هـای ۲-۳ و ۲-۴، مـورد بررسـی قرار گرفته اند. الگوریتم‌های فرامکاشفه‌ای روش هایی محاسباتی سطح بالایی هستند که به صورت تکراری سـعی در بهبود جواب مسئله دارند، ولی یافتن جواب بهینه را تضمین نمی کنند. این الگوریتم ها در بخش ۲-۵ تشـریح شده اند.

علاوه بر نسخه اصلی مسئله SCS، تا به حال نسخه های خاص از مسئله SCS نیز مورد توجه قرار گرفته اند. در [۲۶] برای دو نسخه خاص از مسئله SCS، الگوریتم های تقریبی ارائه شـده اسـت. در نسـخه اول، فـرض شده است که تمام رشته ها دارای طول ۲ می باشند. در نسخه دوم نه تنها فرض شده است همه رشته هـا دارای طول ۲ می‌باشند، بلکه هر نماد از الفبا در S (مجموعه رشته های ورودی) حداکثر ۳ بار تکرار شده است و بـرای آن یک الگوریتم با نسبت تقریب ۷/۶ ارائه شده است. در [۲۷] نیز حالت محدود شده مسئله SCS که همه رشته ها دارای طول ۲ می‌باشند، مورد بررسی قرار گرفته است.

۲-۲ الگوریتم های دقیق

در [۴] الگوریتم برنامه نویسی پویا برای حل مسئله SCS بیان شده اسـت. همچنـین در [۹] بـه بررسـی راه‌حل‌های دقیق مسئله SCS با استفاده از الگوریتم برنامه نویسی پویا و الگوریتم های شاخه و حد پرداخته شده است. الگوریتم های برنامه نویسی پویا فقط بر روی تعداد کمی رشته داده شده موفق عمل می کنند و در صـورت زیاد شدن تعداد رشته ها، زمان و فضای حافظه بسیار زیادی نیاز دارند.کوتاه ترین ابرتوالی مشـترک دو رشـته بـا طول m توسط برنامه نویسی پویا با پیچیدگی زمانی و مکانی $o(m^2)$ محاسبه می شود. این الگوریتم مسـئله ای با n رشته با طول m را با پیچیدگی زمانی و مکانی $o(m^n)$ محاسبه می کند[۴]. الگوریتم های شاخه و حد نیز به زمان زیادی برای محاسبه جواب بهینه نیازدارند، مگر آن که اندازه الفبا بسیار کوچک باشد [۹]. در این بخـش، ابتدا الگوریتم برنامه‌نویسی پویا برای حل مسئله SCS با دو رشته ورودی و سپس این الگوریتم برای حل مسـئله SCS با n رشته ورودی بر اساس [۴] تشریح می شود.

۲-۲-۱ الگوریتم برنامه نویسی پویا برای حل مسئله SCS با دو رشته ورودی

کوتاه ترین ابر توالی مشترک دو رشته x و y توسط برنامه نویسی پویا، در زمان چند جمله ای به دست می‌آید. در صورتی که $length(i,j)$ نشان دهنده طول کوتاه ترین ابر توالی مشترک دو رشته $x(1:i)$ و $y(1:j)$ باشد، رابطه بازگشتی ۲-۱ زیر برای یافتن کوتاه ترین ابرتوالی مشترک دو رشته x و y به کار می رود[۴].

$$length(i,j)=\begin{cases} j & if\ i=0 \\ i & if\ j=0 \\ length(i-1,j-1)+1 & if\ i,j\succ 0\ ,\ x(i)=y(j) \\ \min\{length(i,j-1)+1, length(i-1,j)+1\} & if\ i,j\succ 0, x(i)\neq y(j) \end{cases} \qquad ۲\text{-}۱$$

شبه کدهای ۲-۱ و ۲-۲ طریقه یافتن کوتاه ترین ابر توالی مشترک دو رشته x و y را به وسیله رابطه بازگشتی فوق نشان می دهد.

```
find-SCS(x,y)
{       for(i=0:|x|)
                length(i,0)=i
        for(j=0:|y|)
                length(j,0)=j
        for(i=1:|x|)
                for(j=1:|y|)
                {       if(x(i)==y(j))
                        {       length(i,j)=length(i-1,j-1)+1
                                Previous(i,j)=1
                        }
                        else
                        {       if(length(i,j-1)<length(i-1,j)
                                {       length(i,j)=length(i,j-1)+1
                                        Previous(i,j)=2
                                }
                                else
                                {       length(i,j)=length(i-1,j)+1
                                        Previous(i,j)=3
                                }
                        }
                }
back-tracking(|x|,|y|)
}
```

شبه کد ۲-۱- تابع find-SCS برای یافتن SCS دو رشته x و y با برنامه نویسی پویا [۴]

در تابع find-SCS به وسیله برنامه نویسی پویا، طول ابرتوالی مشترک x و y در ماتریس $length$ ذخیره می‌شود. البته ماتریس دیگری به نام $previous$ هم با اندازه ماتریس $length$ در نظر گرفته شده است. در عنصر $previous(i,j)$ مشخص می شود که برای به دست آوردن $length(i,j)$، از کدام یک از عناصر $length(i-1,j-1)$ و $length(i-1,j)$ و یا $length(i,j-1)$ استفاده شده است. در پایان که تمام عناصر ماتریس $length$ محاسبه شد، تابع $back-tracking$ با پارامترهای $|x|$ و $|y|$ فرا خوانده می شود.

```
back-tracking(i,j)
```

```
{       if(i==0&j==0)
                return SCS:
        If(i==0)
        {       SCS=y(0:j)+SCS
                return SCS
        }
```

شبه کد ۲-۲- تابع back-tracking برای به دست آوردن SCS با استفاده از ماتریس previous [۴]

```
        If(j==0)
        {       SCS=x(0:i)+SCS
                return SCS
        }
        switch(previous(i,j))
        {       case1:
                {       SCS=x(i)+SCS // or SCS=y(j)+SCS
                        back-tracking(i-1,j-1)
                }
                case2:
                {       SCS=y(j)+SCS
                        back-tracking(i,j-1)
                }
                case3:
                {       SCS=x(i)+SCS
                        back-tracking(i-1,j)
                }
        }
}
```

ادامه شبه کد ۲-۲- تابع back-tracking برای به دست آوردن SCS با استفاده از ماتریس previous [۴]

تابع $back-tracking$ یک تابع بازگشتی است که رشته ابرتوالی x و y را با استفاده از مقادیر ماتریس $previous$، به دست می آورد. با توجه به مقدار $previous(i,j)$، در فراخوانی $back-tracking(i,j)$ نماد $x(i)$ و یا $y(j)$ به SCS اضافه شده و تابع $back-tracking$ با پارامترهای جدید فرا خوانی می شود. هنگامی که j برابر با صفر شود، SCS ابرتوالی رشته y است، بنابراین قسمت باقیمانده از رشته x به SCS الحاق می شود یعنی $SCS = x(0:i) + SCS$ و مقدار SCS بازگردانده می شود. همین طور هنگامی که i برابر با صفر شود، SCS ابرتوالی رشته x است، بنابراین قسمت باقیمانده از رشته y به SCS الحاق می شود یعنی $SCS = y(0:j) + SCS$ و مقدار SCS بازگردانده می شود. و در صورتی که i و j هر دو برابر صفر باشند، SCS ابرتوالی هر دو رشته x و y می باشد و بازگردانده می شود. در بخش بعدی، یافتن ابرتوالی n رشته با روش برنامه نویسی پویا بیان می شود.

۲-۲-۲ الگوریتم برنامه نویسی پویا برای حل مسئله SCS با n رشته ورودی

در [۴] به صورت خلاصه، یافتن ابرتوالی مشترک n رشته با روش برنامه نویسی پویا بیان شده است که در اینجا تشریح می شود. ابتدا بعضی تعاریف و نمادها بیان می شود. فرض می شود که هر یک از رشته های s_i،