توسعه یک الگوریتم نقطه مرزی برای حل مسائل برنامه‌ریزی خطی با جواب اولیه موجه

محمد نکوفر[۱]، محمدعلی موفق‌پور[۲،*]

اطلاعات مقاله

دریافت مقاله: ۱۳۹۴/۰۵/۱۸
پذیرش مقاله: ۱۳۹۷/۰۳/۰۹

واژگان کلیدی:
برنامه‌ریزی خطی،
روش‌های مجموعه فعال،
مرتبه زمانی حل چندجمله‌ای،
روش جهت موجه.

چکیده

در این تحقیق برای حل مسائل برنامه‌ریزی خطی، الگوریتم SALCHOW توسعه داده شده است که در هرگام در جهت گرادیان مقید تابع هدف حرکت می‌کند به نوعی که همواره روی مرز ناحیه موجه باقی می‌ماند. این نوع حرکت بر روی مرز ناحیه موجه متفاوت با رفتار الگوریتم سیمپلکس است که روی گوشه‌های فضای موجه حرکت می‌کند. از سوی دیگر با رفتار الگوریتم های نقاط درونی هم که از روی مرز فضای موجه جدا شده و وارد آن می‌شوند، نیز متفاوت است. در واقع SALCHOW با یافتن تدریجی ضرایب وزنی برای مجموعه‌ای از قیدها و افزودن این جمع وزن‌دار به گرادیان تابع هدف، گرادیان مقید تابع هدف را به روز رسانی می‌کند؛ تا در نهایت ضرایب لاگرانژ قیود فعال در نقطه بهینه مسئله برنامه‌ریزی خطی را محاسبه کند. نتایج محاسباتی بر روی مجموعه‌ای از مسائل نمونه تصادفی تولید شده و چند مسئله استاندارد از پایگاه کتابخانه تحقیق در عملیات با اندازه کوچک نشان دهنده برتری زمانی SALCHOW نسبت به سیمپلکس در این مثال‌های محدود است. به این معنی که متوسط زمان حل الگوریتم توسعه داده شده برای مسائل نمونه تابعی از تعداد متغیرهای تصمیم مسئله است. این امر بر خلاف رفتار سیمپلکس است که زمان اجرای آن در حالت متوسط، تابعی از تعداد قیدهای مسئله است. وجود خطای محاسباتی ناشی از گردکردن اعداد در محیط برنامه نویسی MATLAB امکان قضاوت در مورد برتری قاطع SALCHOW بر سیمپلکس را در حل مسائل کوچک سلب می‌نمود.

۱- مقدمه

برای تقریبا ۴۰ سال روش سیمپلکس که دنتزیگ [۱] معرفی کرده بود، بی‌رقیب و بهترین الگوریتم حل مسائل برنامه‌ریزی خطی بود. تا اینکه این وضعیت با انتشار مقاله روش کارمارکار [۲] و با این ادعا که رفتار عملی این روش در زمانی چند جمله‌ای محدود می‌شود، تغییر کرد و برتری روش کارمارکار بر روش سیمپلکس مورد پذیرش جامعه برنامه‌ریزی خطی قرار گرفت. دسته مسائلی که کلی و مینتی [۳] توسعه دادند، به شدت کارایی روش سیمپلکس در بدترین حالت را زیر سوال بردند. البته این سوال که برتری قطعی متعلق به کدام روش است هنوز مورد قضاوت همه جانبه قرار نگرفته است؛ اگر چه تصور کلی این است که روش جدید در واقع ممکن است فقط برای دسته خاصی از مسائل با ساختار خاص بزرگ [۴] برتر باشد. در هر صورت، اعلام چشمگیر این روش جدید علاقه وافری را برای به‌کارگیری برنامه‌ریزی خطی در حوزه‌های کاربردی برانگیخت. این انگیزش نه تنها به دلیل استفاده از روش کارمارکار بلکه به دلیل ایجاد علاقه به توسعه الگوریتم‌های جدید غیرسیمپلکسی برای حل مسائل برنامه‌ریزی خطی برانگیخته شد. انگیزه بیشتر در جستجو برای روش‌های غیرسیمپلکسی ممکن را می‌توان در نتایج تحقیقات زلنی [۵] و میترا و همکاران [۶] مشاهده کرد. به طور خاص زلنی [۵] نیاز به توسعه الگوریتم‌های بهینه‌سازی موازی را مورد اشاره قرار می‌دهد. محرک این پیشنهاد ظهور کامپیوترهای

*. پست الکترونیک نویسنده مسئول: movafaghpour@jsu.ac.ir
۱. مربی، گروه ریاضی، دانشگاه آزاد اسلامی واحد اندیمشک
۲. استادیار، دانشکده مهندسی مکانیک، دانشگاه صنعتی جندی شاپور دزفول، دزفول، ایران

موازی پیشرفته در آن سال‌ها بود که امکان به‌کارگیری چند هسته کامپیوتری برای انجام مراحل موازی حل یک مسئله بهینه‌سازی بود. تا سال‌های ۱۹۹۰ محققین تحقیق در عملیات تلاش خود را برای توسعه الگوریتم‌های پی‌درپی[1] اختصاص داده بودند زیرا استفاده از کامپیوترهای موازی نوظهور، نیاز به طراحی الگوریتم‌های جدید با معماری کاملا متفاوت داشت [۷].

در این تحقیق یک روش جهت موجه غیرسیمپلکسی ارائه شده است که در زمره روش های تعقیب مرز[2] قرار می‌گیرد. این روش تلاش برای شناسایی مجموعه‌ی محدودیت‌های فعال که گوشه بهینه را تشکیل می‌دهند را از طریق شناسایی دنباله‌ای از نقاط بر روی مرز ناحیه موجه عملی می‌کند. علاوه بر این، الگوریتم جدید را می‌توان در ساختار جدولی نیز پیاده‌سازی کرد تا آموزش و فهم پایه های تئوریک آن تسهیل کردد.

در یک تکرار معمولی از روش توسعه داده شده در این تحقیق، از یک نقطه مرزی موجه در جهت گرادیان مقید حرکت می‌کنیم تا به یک نقطه مرزی دیگر برسیم. جهت گرادیان مقید به‌روز شده، همان جهت گرادیان تابع هدف است که فقط یک ترکیب خطی از گرادیان محدودیت‌های فعال در نقطه مرزی جاری به آن افزوده می شود تا حفظ حرکت در یک جهت موجه تضمین گردد.

۲- تحقیقات پیشین

دسته الگوریتم‌های الحاق لاگرانژی به نوعی مشابه رویکرد مورد نظر این تحقیق را مورد استفاده قرار می‌دهند. به عنوان نمونه [۱۲] الگوریتمی را بر پایه گرادیان تصویر شده طیفی[3] را برای بهینه‌سازی مسائل برنامه‌ریزی غیرخطی توسعه داده‌اند. البته دسته الگوریتم‌های گرادیان تصویر شده قبلاً برای مسائل برنامه‌ریزی خطی مثل [۱۳] و مسائل غیرخطی مثل [۱۴] به کار گرفته شده‌اند و بعد از آن [۱۵] در مورد مرتبه پیچیدگی محاسباتی آنها و شرایط لازم برای همگرایی نتایجی را منتشر کرده‌اند. البته روش گرادیان تصویر شده قبلا توسط روزن [۹] و [۱۶] برای مسائل غیرخطی توسعه داده شده بودند. اخیراً نورمحمدی [۱۷] تلاش کرد تا روش زوتندیک را برای برنامه‌ریزی خطی بازگویی کند، اما وی تحلیلی در مورد رفتار زمانی الگوریتم توسعه داده‌اش ارائه نداد.

آندرانی و همکاران [۱۹] برای حل مسئله مکمل خطی و غیرخطی[4] یک الگوریتم برای بیشینه کردن طول گام در یک روش نقطه داخلی توسعه دادند که از تکنیکهای گرادیان تصویر شده استفاده می‌کند. از آنجا که الگوریتم آنها در دسته روش‌های نقطه درونی قرار می‌گیرد، همه نقص‌هایی که دسته روش‌های نقطه داخلی وارد می‌شود ([۲۵-۲۴])، به الگوریتم توسعه داده شده آنها هم وارد است. از سوی دیگر برای حل مسائل مکمل خطی یکنوا[5] که جواب موجه داشته باشند، نیازی به استفاده از ماژول تصویر کردن گرادیان نیست و الگوریتم آنها به یک روش نقطه داخلی تقلیل می‌یابد.

برای بهینه‌سازی چندهدفه نیز گرانا دروموند [۲۰] یک الگوریتم مبتنی بر گرادیان تصویر شده توسعه داده است.

اهمیت روش‌های نقطه داخلی نه تنها به دلیل کاربردهایشان در برنامه‌ریزی خطی است، بلکه در برنامه‌ریزی غیرخطی غیرمحدب نیز کاربرد دارند ([۲۱]). این کاربردها علاوه بر برنامه‌ریزی محدب درجه دوم تا حوزه‌های همچون برنامه‌ریزی نیمه‌معین و حالت‌های عمومی‌تری در بهینه‌سازی مخروطی نیز گسترده شده‌اند ([۲۲]).

بیشتر روش‌های جهت موجه[6] قدیمی مثل روش زوتندیک [۸]، روش تصویر کردن گرادیان روزن [۹]، روش گرادیان تقلیل یافته ولف [۱۰] و روش سیمپلکس محدب زنگویل [۱۱] همگی به عنوان روش‌های تعقیب مرز[7] شناخته می‌شوند و به تجربه ثابت شده است که همگی این روش‌ها در مقایسه با کارایی روش سیمپلکس از رقابت حذف می‌شوند، البته در عوض این مزیت را دارند که نسخه‌های تعدیل شده آنها می‌تواند برای حالت عمومی‌تر برنامه‌ریزی غیرخطی به کار گرفته شوند [۷].

اسنایمن [۷] یک روش جهت‌های موجه[8] توسعه داد که از روی مرز فضای موجه حرکت نمی‌کرد و یا به عبارتی روی نقاط داخلی حرکت می‌کرد و جزو روش‌های تعقیب مرز[9] قرار نمی‌گرفت؛ الگوریتم پیشنهادی وی در هر تکرار با رسیدن به یک محدودیت که مانع پیشرفت در راستای بردار

[1] sequential algorithm
[2] boundary following
[3] Spectral Projected Gradient Method
[4] Linear/Nonlinear Complementarity Problem
[5] monotone Linear Complementary Problem (LCP)
[6] feasible direction
[7] boundary following
[8] feasible direction method
[9] boundary following

تصویر شده تابع هدف می‌شد، در راستای متعامد بر قیدهایی که به تازگی فعال شده‌اند (و با طول گام معینی)، وارد فضای موجه می‌شد و مجدداً در راستای تابع هدف حرکت می‌کرد تا تکرار بعدی انجام شود. مشکلی که این روش داشت تعیین طول مناسب گام برای هم حرکت بود به نحوی که هم به جواب غیرموجه دچار نشود و هم سرعت همگرا شدن به نقطه بهینه.

روش حل مورد استفاده در این تحقیق یافتن مجموعه‌ای از محدودیت‌ها به عنوان محدودیت فعال است؛ این رویکرد قبلاً به نام استراتژی مجموعه فعال شناخته شده و توسط محققین بسیاری مثل هینترمولر و همکاران [۱۸] مورد استفاده قرار گرفته است.

مسئله بهینه‌سازی یک تابع محدب روی فضای موجهی که فقط با کران‌های بالا و پایین تعریف می‌شود توسط بیرجین و مارتینز [۲۷] مورد بحث قرار گرفته است. آنها یک الگوریتم گرادیان تصویر شده[1] ارائه دادند. یکی از بهترین و مشابه‌ترین تحقیقات گزارش شده توسط زیائو و همکاران [۲۳] انجام شده است. ایده اصلی آنها گرادیان طیفی تصویر شده[2] است. بدین معنی که در این روش با استفاده از ضرایب طیفی الگوریتم گرادیان تصویر شده تسریع می‌شود. الگوریتم آنها همانند روشی که ما در اینجا توسعه داده‌ایم در نقطه بهینه مجموعه محدودیت‌های فعال را شناسایی می‌کرد بدون اینکه برقراری شرایط مکمل لنگی را بررسی کند. مزیت این روش سرعت بالاتر بود که اجازه می‌داد مسائل بزرگ مقیاس را حل کنند؛ همچنین مزیت دیگر آن توانایی حل مسائل تبهگن بود. مهم‌ترین ضعف الگوریتم پیشنهادی آنها این بود که برای حل مسائل بهینه‌سازی خطی توسعه داده شد که محدودیت‌های فنی آنها فقط از نوع کران بالا و پایین بودند و اینکه فقط توانستند اثبات کنند تحت شرایط خاصی الگوریتم آنها همگرا است. در نتایج محاسباتی که توسط زیائو و همکاران [۲۳] منتشر شد، در مقایسه الگوریتم خودشان با روش بیرجین و مارتینز [۲۷] برتری قاطعی گزارش نشد. آندریتا و همکاران [۲۸] بعدها این الگوریتم را برای حالت محدودیت‌های خطی عمومی گسترش دادند. مهم‌ترین ضعف الگوریتم آنها این بود که رخی پارامترها باید متناسب با شرایط مسئله مورد حل تنظیم می‌شدند.

[1] gradient projection
[2] projected spectral gradient

۳- بیان مسئله بهینه‌سازی

یک مسئله برنامه‌ریزی ریاضی خطی به صورت زیر را در نظر بگیرید:

$$\text{Min } z = \sum_{j=1}^{n} c_j x_j \qquad (۱)$$

Subject to

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \quad i = 1,2,...,m \qquad (۲)$$

$$x_j \ge 0, \quad j = 1,2,...,n \qquad (۳)$$

که در آن x یک بردار n بعدی است که متغیرهای تصمیم نامیده می‌شوند. تعداد متغیرهای تصمیم n و تعداد محدودیت‌های مستقل نامساوی m است. مسئله بیان شده در عبارات (۳-۱)، در دسته مسائل بهینه‌سازی مقید قرار می‌گیرد.

یک راه معنادار مقایسه الگوریتم‌های نقطه درونی با روش سیمپلکس این است که ویژگی‌های تئوریکی‌شان در مورد پیچیدگی محاسباتی بررسی شود. همان طور که گفته شد کارمارکار [۲] ثابت کرد که نسخه اصلی الگوریتمش از نظر زمان صرف شده یک الگوریتم چند جمله‌ای است، یعنی زمان مورد نیاز برای حل کردن هر مساله برنامه‌ریزی خطی می‌تواند از بالا توسط یک تابع چندجمله‌ای از اندازه مسئله، محدود شود. مثال‌های نقض ساخته شده توسط کلی و مینتی [۳] نشان دادند که روش سیمپلکس این ویژگی را ندارد، پس با اطمینان می‌توان گفت که سیمپلکس از نظر زمان صرف شده، یک الگوریتم غیر چندجمله‌ای است؛ یعنی زمان مورد نیاز آن نمی‌تواند از بالا توسط یک تابع چندجمله‌ای از اندازه مسئله، محدود شود. تفاوت در عملکرد بدترین حالت سیمپلکس و الگوریتم‌های نقطه داخلی، ارزش توجه کردن را دارد. البته در کاربرد واقعی چیزی که مهم‌تر است، مقایسه آنها در متوسط رفتار به ازای مسائل رایج است. دو عامل اصلی که عملکرد یک الگوریتم را تعیین می‌کند، میانگین زمان (یا تعداد عملیات) در هر تکرار و تعداد تکرارها است. الگوریتم‌های نقطه درونی بسیار پیچیده‌تر از روش سیمپلکس هستند. محاسبات بسیار گسترده‌تری برای هر تکرار نیاز است تا جواب موجه بعدی به دست آید. پس زمان مورد نیاز کامپیوتر در هر تکرار برای یک الگوریتم نقطه درونی چندین برابر طولانی‌تر از زمان

برای روش سیمپلکس است؛ و این موضوع، استفاده از روش‌های نقطه داخلی را برای مسائل رایج که نیاز به تعداد تکرارهای اندک دارند را غیرکاربردی می‌کند؛ سیمپلکس برای این نوع مسائل کارایی بسیار قابل قبولی ارائه می‌دهد. برای مسائل کوچک، تعداد تکرارهای مورد نیاز برای یک الگوریتم نقطه درونی و روش سیمپلکس تا حدّی قابل مقایسه‌اند. برای مثال، در یک مسئله با ۱۰ قید کارکردی، تقریبا ۲۰ تکرار برای هر نوع الگوریتم، عادی خواهد بود. در نتیجه، برای مسائلی با اندازه مشابه، زمان کلی کامپیوتر برای یک الگوریتم نقطه درونی چندین برابر طولانی‌تر از روش سیمپلکس است. از سوی دیگر، یک مزیت کلیدی الگوریتم‌های نقطه درونی این است که برای مسائل بزرگ به تکرارهای بسیار بیشتری نسبت به مسائل کوچک نیاز ندارند. برای مثال، یک مسئله با ۱۰ هزار قید کارکردی احتمالا به کمتر از ۱۰۰ تکرار از الگوریتم‌های نقطه درونی نیاز خواهد داشت. حتی با توجه به زمان کامپیوتری بسیار زیاد برای هر تکرار که برای یک مسئله این اندازه‌ای نیاز است، یک چنین تعداد کم تکرارها مسئله را کاملا قابل کنترل می‌کند. در مقابل، روش سیمپلکس ممکن است به ۲۰ هزار تکرار نیاز داشته باشد و بنابراین ممکن نیست در یک مقدار زمان کامپیوتری معقول تمام شود. پس الگوریتم-های نقطه درونی معمولا سریع‌تر از روش سیمپلکس برای مسائل بزرگ است (هیلیر و لیبرمن [۲۶]).

دلیل تفاوت بسیار بزرگ در تعداد تکرارها برای حل مسائل بزرگ، تفاوت در زنجیره جواب‌های تولید شده طی مراحل حل است. در هر تکرار، روش سیمپلکس از جواب گوشه موجه کنونی به یک جواب گوشه موجه بعدی (که در طول یک لبه بر مرز منطقه موجه همسایه‌اند) حرکت می‌کند. مسائل بسیار بزرگ تعداد بسیار زیادی از جواب‌های گوشه موجه دارند. مسیر پیشرفت الگوریتم از جواب اولیه گوشه موجه تا یک جواب بهینه نیز می‌تواند یک مسیر بسیار غیرمستقیم و پرپیچ و کند در امتداد مرز باشد، و در هر مرحله فقط گام کوچکی به سوی جواب گوشه موجه مجاور بعدی پیش برود، پس یک تعداد بسیار زیادی تکرار نیاز است تا یک جواب بهینه به دست آید. در مقابل یک الگوریتم نقطه درونی با دور زدن همه این محدودیت‌ها و شوت کردن از درون منطقه موجه به سمت یک جواب بهینه، کل این را دور می‌زند. اضافه کردن قیدهای کارکردی بیشتر، مرزهای قید بیشتری به منطقه موجه اضافه می‌کند اما تأثیر کمی بر تعداد جواب‌های آزمایشی مورد نیاز روش-های نقاط داخلی بر این مسیر درونی دارد. این امر اجازه می‌دهد الگوریتم‌های نقطه درونی مسائل با تعداد زیاد قیدهای کارکردی را به سادگی حل کنند. یک مقایسه کلیدی نهایی به توانایی انجام انواع تحلیل‌های پسابهینگی مختلف مرتبط می‌باشد. روش سیمپلکس و نسخه‌های مشتق شده از آن، برای انجام این نوع تحلیل‌ها بسیار مناسب هستند و به طور گسترده‌ای برای آن به کار برده می‌شوند. اما متأسفانه، رویکرد نقطه درونی در حال حاضر توانایی محدودی در این زمینه دارد. با توجه به اهمیت زیاد تحلیل پسابهینگی، این یک ایراد مهم الگوریتم‌های نقطه درونی است.

مهم‌تر از همه اینها، اخیرا دزا و همکاران [۲۴] بیان کردند: مسیر داخلی (که از میان فضای موجه عبور می‌کند و آن را مسیر مرکزی[1] نامیدند) که الگوریتم‌های نقاط درونی طی می‌کنند، در حالتی که تعدادی نمایی[2] قید نامساوی زاید[3] به مسئله استاندارد مکعب اعوجاج‌دار[4] کلی - مینتی افزوده شود، از همه گوشه‌های فضای موجه عبور می‌کند. لذا الگوریتم‌های نقاط درونی هم در این گونه مسائل نیاز به تعداد گام‌های غیر چندجمله ای (نمایی) دارند. دزا و همکاران [۲۵] کران بالای تعداد این گام ها را $O(2^n-1)$ تعیین کردند که n تعداد متغیرهای تصمیم است. البته شایان ذکر است که اگر تعریف بزرگی و ابعاد یک مسئله برنامه‌ریزی خطی به جای اینکه فقط بر اساس تعداد متغیرهای تصمیم n باشد، در برگیرنده تعداد قیدهای فنی m هم باشد، ایراد رشد نمایی زمان اجرای الگوریتم برطرف می‌شود. بدین معنی که در تحلیلی که دزا و همکاران [۲۴] انجام دادند، ابعاد مسئله m به صورت نمایی رشد کرد و زمان حل نیز به صورت یک تابع چندجمله‌ای از ابعاد مسئله (که خود تابعی از m و n است) رشد کرد.

۳-۱- مفاهیم کلیدی

از جبر خطی می‌دانیم که هر بردار C را می‌توان به صورت دو مولفه دلخواه $C^{\perp}$ و $C^{\parallel}$ تجزیه کرد که بر یکدیگر متعامدند:

[1] Central Path
[2] exponential
[3] redundant inequalities
[4] squashed

$$C = C^{\parallel} + C^{\perp};\ C^{\parallel} \bullet C^{\perp} = 0$$

به طوری که اگر بردار $C^{\parallel}$ موازی ابرصفحه دلخواه H باشد اصطلاحاً تصویر C روی H خوانده می‌شود و آن را به صورت $C^{\parallel_H}$ نمایش می‌دهیم:

$$\forall C \in \mathbb{R}^n, H = \left\{ X \in \mathbb{R}^n \mid a^i X = b_i, a^i \in \mathbb{R}^n, b_i \in \mathbb{R} \right\}:$$
$$\exists C^{\parallel_H}, C^{\perp_H} \in \mathbb{R}^n \rightarrow C = C^{\parallel_H} + C^{\perp_H}; a^i \bullet C^{\parallel_H} = 0$$

با شروع حرکت از هر نقطه $X_{(1)}$ عضو ابرصفحه H، با حرکت در راستای $C^{\parallel_H}$ و با هر طول گام دلخواه به نقطه $X_{(2)}$ می‌رسیم که همچنان نقطه $X_{(2)}$ نیز عضو ابرصفحه H است. به عبارت دیگر:

$$\forall X_{(1)} \in H \Rightarrow a^i X = b_i$$

چون $a^i \bullet C^{\parallel_H} = 0 \xrightarrow{\substack{X_{(1)} \in H, \\ X_{(2)} = X_{(1)} + \lambda C^{\parallel_H}}} a^i \bullet \left(X_{(1)} + \lambda C^{\parallel_H} \right)$

$$= a^i \bullet X_{(1)} + a^i \bullet \lambda C^{\parallel_H} = b_i + \lambda \left(a^i \bullet C^{\parallel_H} \right)$$
$$= b_i + \lambda(0) = b_i \Rightarrow a^i \bullet X_{(2)} = b_i \Rightarrow$$
$$X_{(2)} = \left(X_{(1)} + \lambda C^{\parallel_H} \right) \in H \qquad (۴)$$

بر این اساس برای محاسبه $C^{\parallel_H}$ و $C^{\perp_H}$ از یک داده شده و نسبت به H داریم:

$$C = C^{\parallel_H} + C^{\perp_H} \overset{\exists \lambda \in \mathbb{R}}{\Rightarrow}$$
$$C^{\parallel_H} = C - \lambda a^i, C^{\perp_H} = C + \lambda a^i, \qquad (۵)$$
$$\overset{a^i \bullet C^{\parallel_H} = 0}{\Rightarrow} \exists \lambda \in \mathbb{R} : a^i \bullet \left(C - \lambda a^i \right) = 0 \qquad (۶)$$

و در نتیجه:

$$\overset{a^i \bullet C^{\parallel_H} = 0}{\Rightarrow} a^i \bullet \left(C - \lambda a^i \right) = 0$$

$$\Rightarrow C.a^i - \lambda . a^i . a^i = 0 \Rightarrow \lambda = \frac{C.a^i}{a^i . a^i} \qquad (۷)$$

$$C^{\parallel_H} = C - \frac{C.a^i}{a^i . a^i} . a^i \qquad (۸)$$

مقدار λ در رابطه (۷) را ضریب تعدیل می‌نامیم.

محاسبات فوق را می‌توان برای تصویر کردن $C \in \mathbb{R}^n$ بر روی مجموعه ای از r ابر صفحه با بردار نرمال‌های $a^1, a^2, \ldots, a^r$ در فضای R^n نیز انجام داد به طوری که:

$$a^i \bullet C^{\parallel} = 0, \quad \forall i = 1, 2, \ldots, r \qquad (۹)$$

محاسبه ضرایب تعدیل λ_i که منجر به دستگاه r معادله - r مجهولی (۹) می‌گردد به سادگی میسر است.

۳-۲- مجموعه محدودیت‌های فعال و جواب‌های پایه سیمپلکس اولیه

در حل مدل برنامه‌ریزی خطی (۳-۱) مبنای ایجاد یک جواب موجه پایه در روش سیمپلکس و همه روش‌های مشتق شده از آن، افراز کردن مجموعه همه متغیرهای مسئله (شامل متغیرهای تصمیم و کمکی) به دو دسته پایه و غیرپایه است. در اینجا این دسته الگوریتم‌ها را مبتنی بر افراز متغیرها می‌نامیم. مبنای ساخت یک جواب پایه در این دسته از روش‌ها، شناسایی دسته‌ای m تایی از کل متغیرها به عنوان متغیر پایه با مقداری بزرگ‌تر یا مساوی صفر و نسبت دادن مقدار صفر به سایر متغیرها (به عنوان متغیر غیرپایه) است.

از سوی دیگر به طور معادل می‌توان به جای تمرکز بر دسته‌بندی متغیرها به دو افراز "پایه" و "غیرپایه" بر دسته‌بندی محدودیت‌ها به دو دسته "فعال" و "غیرفعال" تمرکز کرد. این رویکرد منجر به توسعه دسته جدیدی از الگوریتم‌ها شده که به جای متغیرهای پایه به دنبال مجموعه محدودیت‌های فعال یا به طور معادل "مجموعه فعال[1] " هستند. در اینجا به این دسته الگوریتم‌ها، دسته مبتنی بر افراز محدودیت‌ها می‌نامیم. این دسته روش‌ها برای حل مسائل غیرخطی به طور فراگیری مورد استفاده محققین قرار دارند. در این تحقیق روش توسعه داده شده در دسته دوم روش‌ها (مبتنی بر افراز محدودیت‌ها یا به طور معادل روش‌های مجموعه فعال[2]) برای یافتن جواب گوشه قرار می‌گیرد.

در روش‌های مبتنی بر افراز محدودیت‌ها مبنای تعیین مختصات یک نقطه گوشه، محل تقاطع n قید از مجموعه کل قیدها ($m+n$ قید شامل: m قید کارکردی و n قید غیرکارکردی یا نامنفی بودن متغیرها) در فضای R^n است. این زیرمجموعه از قیدها به نام محدودیت‌های فعال در آن نقطه گوشه شناخته می‌شوند.

در هر جواب گوشه تصویر گرادیان تابع هدف بر روی فضای موجه را با نام $C^{\parallel}$ نشان می‌دهیم و به صورت ترکیب خطی

[1] Active Set
[2] Active set methods

نامنفی از قیدهای فعال به همراه تابع هدف قابل بازنویسی است.

$$C^{\parallel} = C - \sum_{i \in ActiveSet} \lambda_i . a^i , \lambda_i > 0. \qquad (۱۰)$$

در مسئله‌ای به فرم (۳-۱)، هر گاه از یک نقطه عضو فضای موجه (درون فضا، روی مرز غیرگوشه‌ای فضا و یا روی گوشه‌ها) مانند X_0 در جهت گرادیان تابع هدف C با طول گام α ($\forall \alpha \in \mathbb{R}$) حرکت کنیم و به نقطه‌ای مانند X_1 برسیم:

$$X_1 = X_0 + \alpha . C \qquad (۱۱)$$

برای هر محدودیت iام مسئله با بردار سطری ضرایب فنی a^i یکی از سه وضعیت ذیل روی خواهد داد:

$$a^i C > \xrightarrow[X_1 = X_0 + \alpha . C,\ \alpha > 0]{a^i X_0 \le b_i} a^i X_1 > b_i \qquad (۱۲)$$

$$a^i C = 0 \xrightarrow[X_1 = X_0 + \alpha . C,\ \alpha > 0]{a^i X_0 \le b_i} a^i X_1 = b_i \qquad (۱۳)$$

$$a^i C < 0 \xrightarrow[X_1 = X_0 + \alpha . C,\ \alpha > 0]{a^i X_0 \le b_i} a^i X_1 < b_i \qquad (۱۴)$$

مهم‌ترین کاربرد عبارات (۱۲-۱۴) در اینجا این است که از بین همه محدودیت‌های a^i در نقطه کنونی X_0 که فعال نیستند، با حرکت کردن در امتداد C، تنها محدودیت‌هایی از نوع $A_i C > 0$ ممکن است به مجموعه فعال افزوده شوند و سایر قیود به هیچ عنوان کاندید ورود به مجموعه فعال نخواهند بود و بررسی نخواهند شد.

برای توسعه یک الگوریتم نقطه مرزی برای مسئله (۳-۱) به جای شناسایی جواب‌های گوشه (جواب‌های شامل m متغیر غیرصفر یا به طور معادل گوشه‌های شامل n قید فعال) ما به دنبال شناسایی r قید فعال ($r \le n$ برای جواب‌های غیرتبهگن) و افزایش تدریجی مقدار r هستیم. برای شناسایی قیدهای فعال در هر تکرار از الگوریتم توسعه داده شده، ما در راستای بردار گرادیان تابع هدف تصویر شده بر فضای موجه $C^{\cdot}$ حرکت می‌کنیم تا به یک نقطه مرزی جدید از ناحیه موجه برسیم. در این نقطه مرزی جدید محدودیتی که حرکت را متوقف کرده (قید با $a^i C > 0$ که از وضعیت $a^i X_0 > b_i$ به وضعیت $a^i X_1 = b_i$ تبدیل شده است) باید به مجموعه قیدهای فعال افزوده گردد و برخی قیدهایی که شرط فعال بودن را از دست داده‌اند (قید با $A_i C < 0$ که از وضعیت $a^i X_0 = b_i$ به وضعیت $a^i X_1 < b_i$ تبدیل شده است)، باید از مجموعه فعال خارج شوند. از سوی دیگر برخی قیدها که $a_i C = 0$ بوده‌اند و از وضعیت $a^i X_0 = b_i$ به وضعیت $a^i X_1 = b_i$ تغییر کرده‌اند، همچنان در مجموعه فعال باقی می‌مانند در نقطه مرزی نقطه مرزی بهینه و صفرشدن مقدار تابع هدف به‌روز شده، تکرار می‌گردد.

۳-۳- ارتباط ضرایب تعدیل و ضرایب لاگرانژ

در این تحقیق برای متوقف شدن الگوریتم توسعه داده شده، لازم است تا تصویر بردار گرادیان تابع هدف صفر گردد و دیگر امکان هیچ گونه بهبودی در مقدار تابع هدف مقید وجود نداشته باشد. این بدان معناست که بردار تابع هدف تصویر شده که قبلا در (۱۰) تعریف شد، برابر صفر گردد. به بیان شفاف تر

$$C' = C - \sum_{i \in ActiveSet} \lambda_i . a^i \qquad (۱۵)$$

$$C - \sum_{i \in ActiveSet} \lambda_i . a^i = 0 \qquad (۱۶)$$

برقراری این رابطه در واقع شرط لازم کمینه شدن تابع لاگرانژی مسئله است (البته می توان اثبات کرد که سایر شرط‌ها نیز به صورت ضمنی تامین گردیده‌اند)، که در این صورت ضرایب λ_i همان ضرایب لاگرانژ مسئله در نقطه بهینه خواهند بود.

$$C - \sum_{i \in ActiveSet} \lambda_i . a^i + \sum_{i \notin ActiveSet} 0 . a^i = 0 \qquad (۱۷)$$

اما در سیر پیشرفت الگوریتم حل و رسیدن به نقطه بهینه همان‌طور که دیدیم رابطه (۹) در هر جواب مرزی یک مقدار λ برای هر قید فعال ارائه می‌دهد. پس می‌توان نتیجه گرفت که الگوریتم ارائه شده در این تحقیق در هر تکرار ضریب تعدیلی را برای هر قید فعال ارائه می‌دهد که نقش آن تعدیل تابع هدف به بردار جدیدی است که حرکتش در امتداد مرز موجه فضای حل قرار گیرد؛ و در واقع ما در اینجا به دنبال محاسبه گام به گام ضرایب لاگرانژ از طریق همگرا شدن به مجموعه فعال با کمک حرکت در جهتی که تابع لاگرانژی را کمینه می‌کند، هستیم. الگوریتم توسعه داده‌شده در این تحقیق "محاسبه متوالی ضرایب لاگرانژ با حفظ حرکت روی مرز فضای جواب[1] " یا به

[1] Successively Acquiring Lagrange Coefficients by HOlding Wayside

اختصار SLACHOW نامیده می‌شود.

الگوریتم SALCHOW

گام ۰. دریافت ورودی ها: بردار تابع هدف ماکزیمم سازی C ، ماتریس قیود فنی کوچکتر یا مساوی A و مقادیر سمت راست محدودیت‌ها b .

گام ۱. مقداردهی اولیه: $k=1,\ X_k=0,\ C_k=C$

گام ۲.

محاسبه حداکثر طول گام حرکت در جهت C_k:

$$\alpha_k=\min_{i|A_iC_k\neq 0}\left\{\frac{b-A_iX_k}{A_iC_k}\right\},$$

$$\theta_k=\arg\min_{i|A_iC_k\neq 0}\left\{\frac{b-A_iX_k}{A_iC_k}\right\}.$$

گام ۳. به روز رسانی مجموعه فعال:

$$ActiveSet_{k+1}=$$

$$\{i\,|\,1\le i\le m, A_i(X_k+\alpha_kC_k)=b_i\}$$

گام ۴. بررسی شرط توقف: اگر $\alpha_k=0$ برو به گام ۷.

گام ۵. محاسبه نقطه مرزی بعدی و جهت تابع هدف تصویر شده: $X_{k+1}=X_k+\alpha_kC_k$ جواب مرزی بعدی خواهد بود و بردار تابع هدف تصویر شده نیز $C_{k+1}=C_1-\sum_{i\in ActiveSet}\frac{A_i.C_1}{A_i.A_i}A_i$ خواهد بود.

گام ۶. تکرار: اگر $C_{k+1}=0$ برو به ۷ در غیر این صورت قرار بده $k=k+1$ و برگرد به گام ۲.

گام ۷. توقف: مجموعه فعال، جواب بهینه و مقدار تابع هدف را گزارش بده.

۴- نتایج محاسباتی

برای بررسی کارایی محاسباتی SALCHOW دو معیار زمان و دقت حل عددی مورد توجه قرار گرفت. چون کد SALCHOW در محیط نرم‌افزار MATLAB r2012a – 64 bit توسعه داده شده است، برای یکسان و نرمال‌سازی شرایط مقایسه، از تابع linprog موجود در MATLAB به عنوان روش مبنا برای حل مثال‌ها استفاده شد.

دو دسته مسئله با ابعاد کوچک به عنوان مسائل نمونه استفاده شد. در دسته اول مسائلی قرار دارند که بر اساس تعداد متغیرها و محدودیت‌های هر مسئله، پارامترهای مورد نظر به صورت تصادفی با توزیع یکنواخت تولید شدند. در دسته دوم از مسائل استاندارد موجود در پایگاه OR-Library استفاده شد.

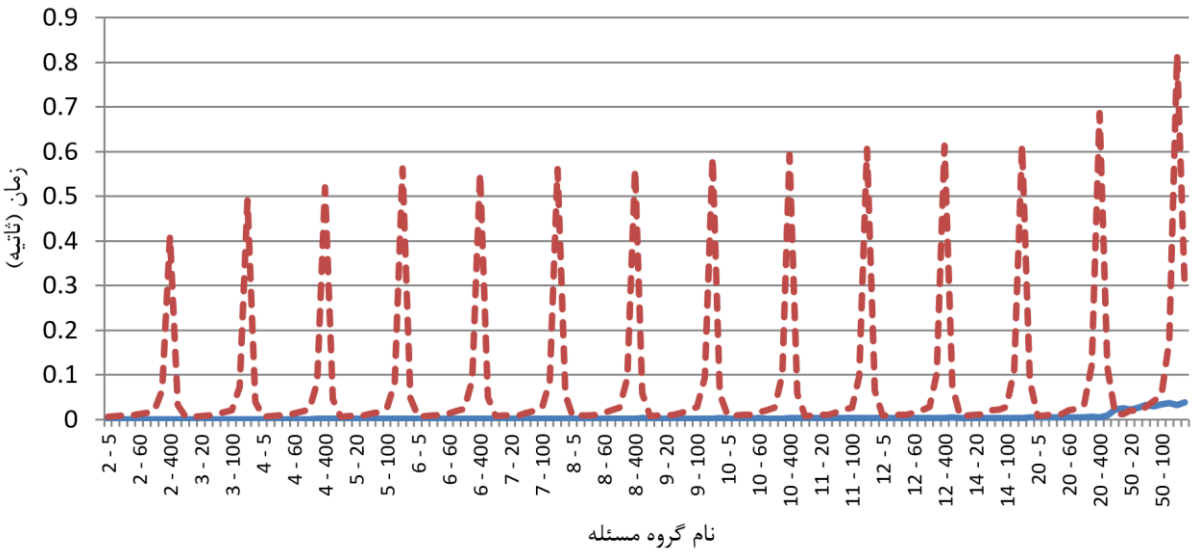
برای تولید مسائل نمونه تصادفی تعداد متغیرهای تصمیم و تعداد قیود کارکردی به عنوان معیار دسته‌بندی انتخاب شدند. بدین منظور مسائلی با ۲، ۳، ۴، ۵، ۶، ۷، ۸، ۹، ۱۰، ۱۱، ۱۲، ۱۴، ۲۰ و ۵۰ متغیر تصمیم ساخته شد. برای هر یک از این مسائل نیز تعداد ۵، ۱۰، ۲۰، ۳۰، ۶۰، ۸۰، ۱۰۰، ۲۰۰، ۴۰۰ و ۷۰۰ محدودیت فنی ساخته شد. در نتیجه جمعاً تعداد ۱۰×۱۴ گروه مسئله ساخته شد. به عنوان مثال گروه مسئله ۸۰-۱۴ دارای ۱۴ متغیر تصمیم و ۸۰ محدودیت فنی است. برای حذف اثرات تصادفی نیز در هر گروه تعداد ۵۰ مسئله تولید شد. یعنی جمعاً تعداد ۵۰×۱۴۰ مسئله نمونه تصادفی با دو الگوریتم SALCHOW و linprog حل شد.

مشاهده شد به دلیل بروز خطای عددی در الگوریتم که ناشی از خطای گرد کردن اعداد در محیط MATLAB بود، امکان استفاده از الگوریتم توسعه داده شده برای مسائل دارای تعداد متغیر بیشتر به دلیل بروز خطای گرد کردن ممکن نگردید. البته در صورت استفاده از تکنیک‌های برنامه‌نویسی مثل کار با اعداد کسری به جای کار با اعداد ممیز شناور با دقت محدود، می‌توان این ضعف SALCHOW را حذف کرد و یا اینکه در دل SALCHOW به جای مقایسه مقادیر با عدد صفر از شرط‌های دیگری استفاده کرد. به همین دلیل بیشترین تعداد متغیر تصمیم که در یک مسئله می‌توان با SALCHOW بدون بروز خطای عددی چشمگیر حل کرد محدود بود. که این محدودیت منجر به حل مسائلی گردید که جزو مسائل بسیار کوچک محسوب می‌گردند. برای تعداد قیودی که SALCHOW توانایی تولید جواب موجه داشت محدودیتی مشاهده نگردید.

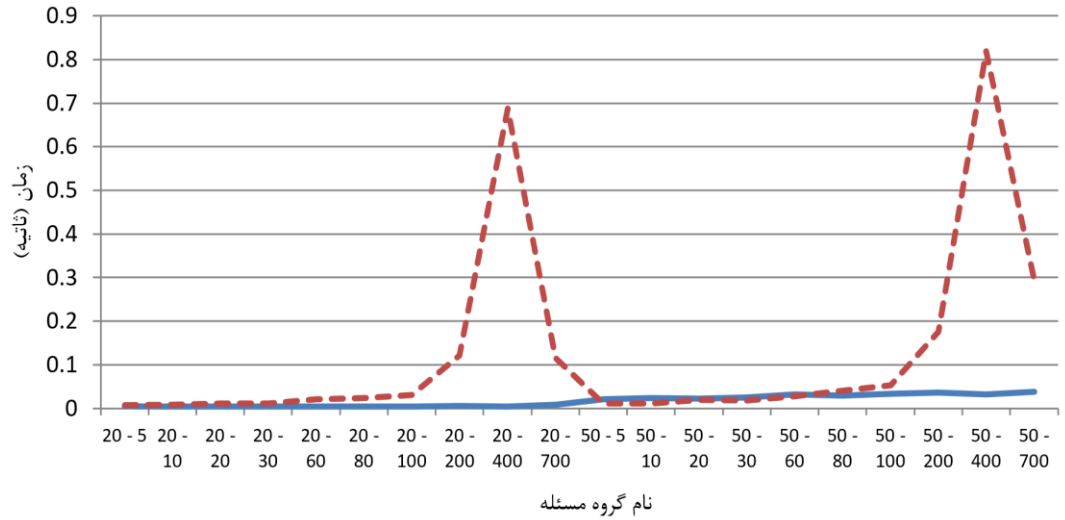
پارامترهای مورد نظر (ضرایب تابع هدف، ضرایب فنی محدودیت‌ها و ضرایب سمت راست قیدهای کوچک‌تر یا مساوی) به صورت اعداد صحیح تصادفی با توزیع یکنواخت تولید شدند. ضرایب تابع هدف با توزیع تصادفی یکنواخت در بازه ۱۰ تا ۲۰ تولید شدند. ضرایب فنی قیدها هم با توزیع تصادفی یکنواخت در بازه ۰ تا ۱۰ تولید شدند. ضرایب سمت راست همه قیدهای کوچک‌تر - مساوی با

توزیع تصادفی یکنواخت در بازه ۱۰ تا ۱۰۱۰ تولید شدند. یکی از مشکلاتی که این نسخه از SALCHOW با آن دست و پنجه نرم می‌کند، مشکل خطای عددی ناشی از محیط برنامه‌نویسی است که منجر به جواب‌های غیرموجه می‌گردد؛ به همین دلیل به بررسی رفتار خطای غیرموجه شدن و یا همگرایی به جواب های زیر بهینه یا فوق‌بهینه هم می‌پردازیم.

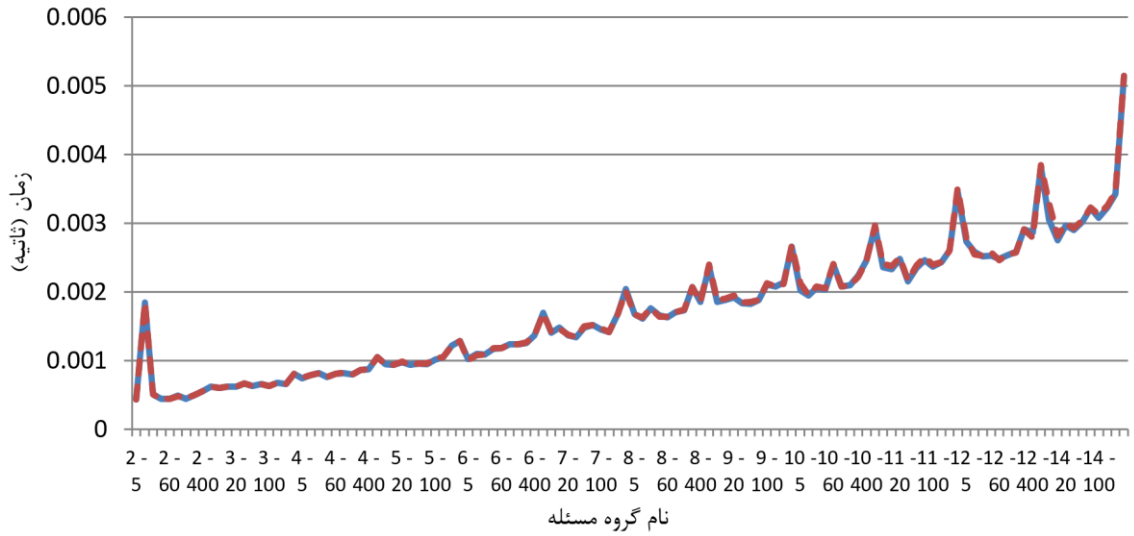
شکل (۱)، متوسط زمان حل هر دسته از مسائل را نشان می‌دهد. در شکل (۱) روی محور افقی نام دسته مسئله قید شده است؛ متوسط زمان حل هر گروه مسئله نیز به ازای ۵۰ مسئله نمونه درون هر گروه محاسبه شده است. الگوریتم مبنا (تابع linprog) مبتنی بر سیمپلکس است، لذا زمان اجرای آن به شدت به تعداد محدودیت‌ها وابسته است و با افزایش تعداد محدودیت‌ها و ثابت ماندن تعداد متغیرهای تصمیم، افزایش چشمگیری دارد. اما زمان حل توسط SALCHOW بیشتر تحت تأثیر تعداد متغیرهای تصمیم است. این موضوع با توجه به رفتار سیکل‌وار خط نقطه‌چین درون شکل (۱) خود را نشان می‌دهد. اولین سیکل مربوط به مسائل با ۲ متغیر تصمیم است؛ وقتی تعداد قیدهای فنی از ۵ به ۷۰۰ افزایش می‌یابد، زمان اجرای کد linprog (خط نقطه‌چین) به صورت نمایی افزایش پیدا می‌کند اما طی این تغییرات زمان اجرای SALCHOW با شیب خطی ناچیزی افزایش پیدا می‌کند. در سیکل دوم خط نقطه چین که مربوط به گروه مسائل با ۳ متغیر تصمیم است، مجدداً با شروع از مسائل با ۵ محدودیت فنی، زمان اجرا افت شدیدی دارد و با افزایش تعداد محدودیت‌های فنی، زمان اجرا نیز همان رفتار رشد نمایی را نشان می‌دهد. این رفتار رشد نمایی عیناً در ۱۲ گروه مسائل بعدی تکرار می‌گردد که نشان می‌دهد زمان اجرای الگوریتم سیمپلکس متأثر از تعداد قیدهای فنی است ولی زمان اجرای SALCHOW این‌گونه نیست. در آخرین گروه که مسائل با ۵۰ متغیر تصمیم قرار دارند، زمان اجرای SALCHOW اندکی رشد از خود نشان می‌دهد که بیانگر این نکته است که زمان اجرای آن از تعداد متغیرهای تصمیم تبعیت می‌کند. شیب افزایش زمان اجرای SALCHOW در طول کل مسائل حل شده، نسبت به تعداد محدودیت‌ها در مقایسه با linprog بسیار ناچیز و تقریبا خطی است. در دو سیکل آخر نمودار شکل (۱) به نظر می‌رسد منحنی متوسط زمان اجرای SALCHOW و linprog با هم تقاطع دارند. برای تدقیق بیشتر این قسمت از نمودار را در شکل (۲) دوباره رسم می‌کنیم. شکل (۲) همانند شکل (۱) متوسط زمان اجرا در هر گروه ۵۰ تایی از مسائل نمونه تصادفی را نشان می‌دهد، ولی فقط نتایج مربوط به مسائل با ۲۰ و ۵۰ متغیر تصمیم را در بر می‌گیرد. یعنی فقط دو سیکل سمت راست شکل (۱) در شکل (۲) قرار دارند. همان‌گونه که مشاهده می‌شود، زمان اجرای SALCHOW فقط برای دسته مسائل ۵-۵۰ تا ۶۰-۵۰ کمتر از ۰٫۰۱ ثانیه بیشتر از زمان linprog گزارش شده است.



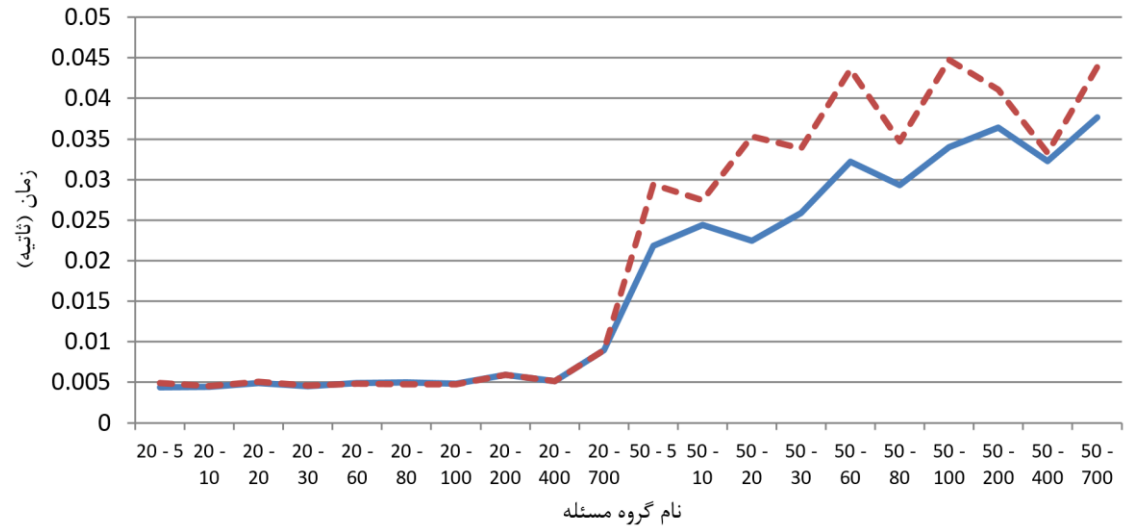
شکل ۱: میانگین زمان حل هر گروه از مسائل (خط نقطه‌چین: زمان الگوریتم مبنا، خط توپر: زمان SALCHOW)



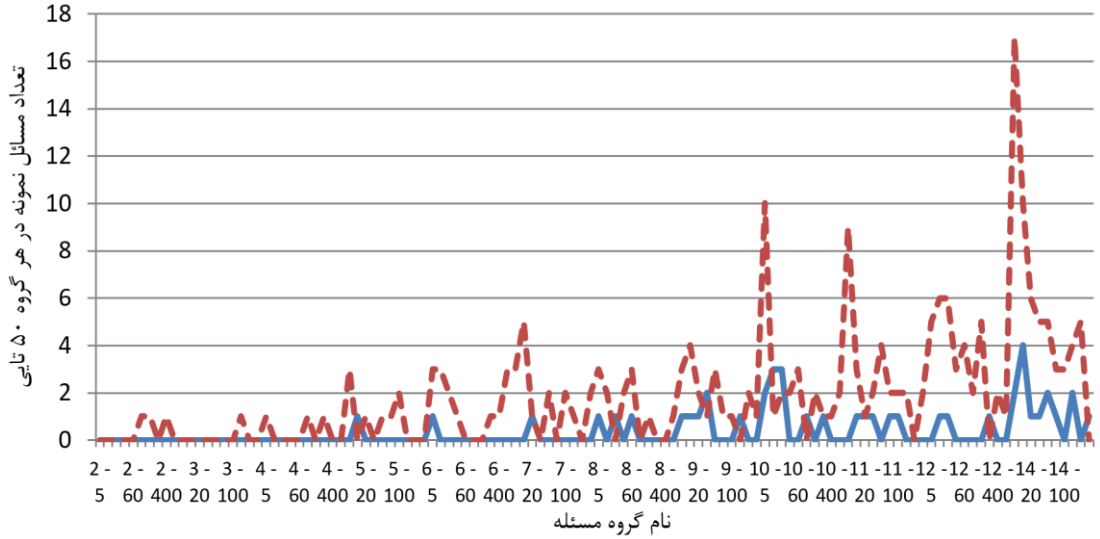
شکل ۲: میانگین زمان حل گروه مسائل با ۲۰ و ۵۰ متغیر تصمیم (خط نقطه چین: زمان الگوریتم مبنا، خط توپر: زمان SALCHOW)



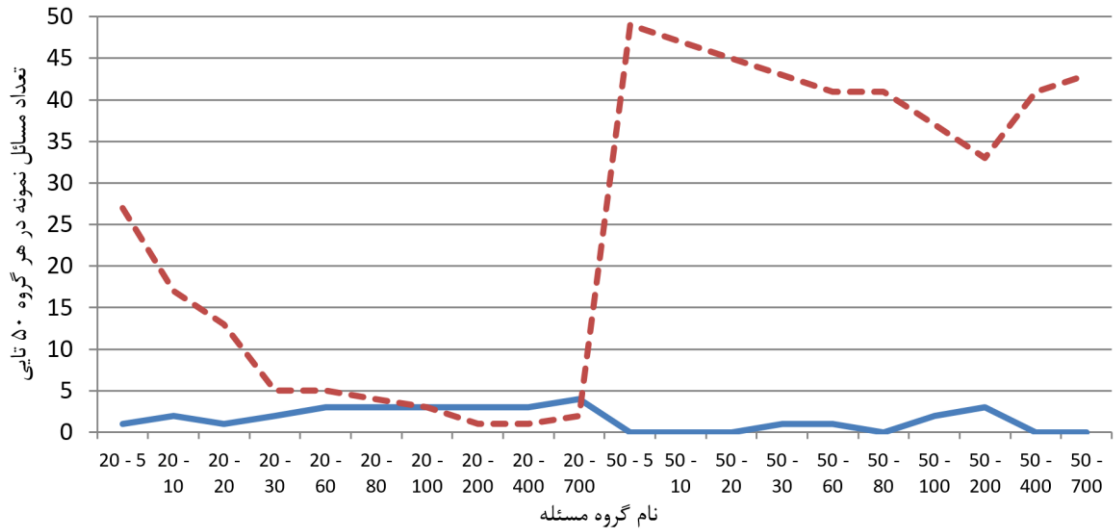
شکل ۳: الف- میانگین زمان حل گروه مسائل با ۲ الی ۱۴ متغیر تصمیم که بدون خطای همگرایی به جواب بهینه (خط نقطه چین: زمان SALCHOW برای مسائل بدون خطای همگرایی، خط توپر: زمان SALCHOW)



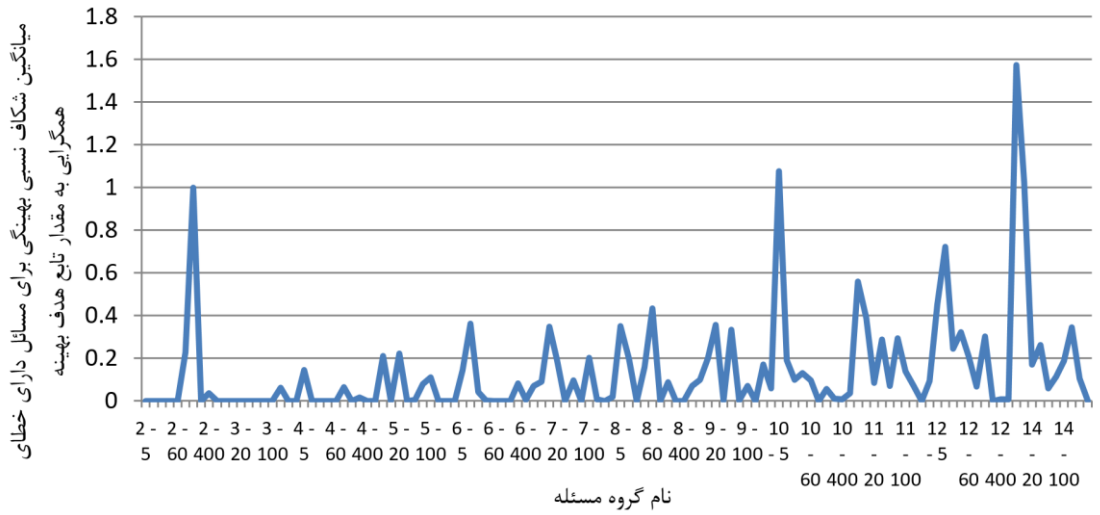
شکل ۳: ب- میانگین زمان حل گروه مسائل با ۲۰ و ۵۰ متغیر تصمیم که بدون خطای همگرایی به جواب بهینه (خط نقطه چین: زمان SALCHOW برای مسائل بدون خطای همگرایی، خط توپر: زمان SALCHOW)



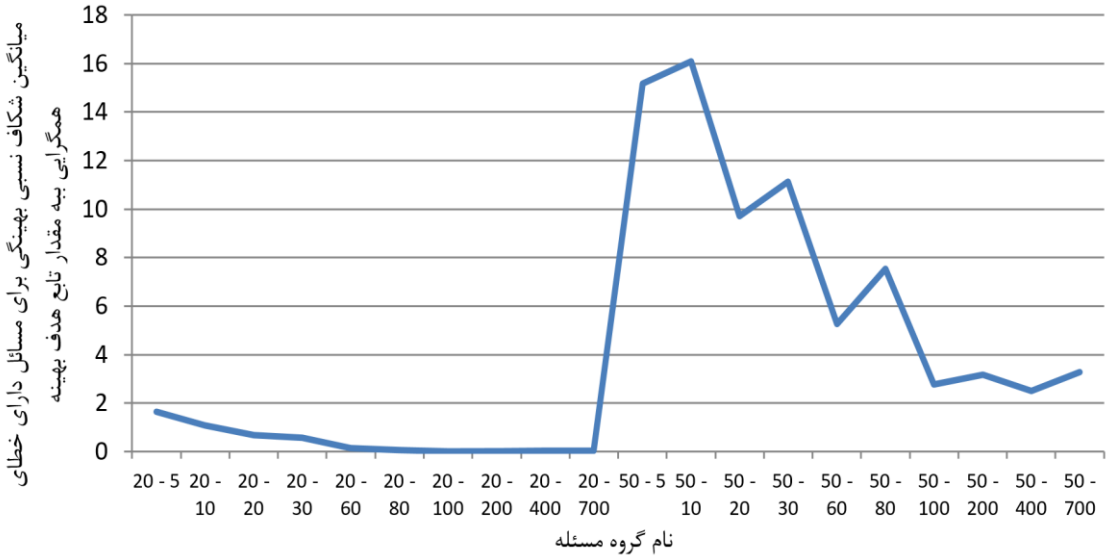
شکل ۴: الف- تعداد مسائل با ۲ الی ۱۴ متغیر تصمیم که به دلیل خطای همگرایی به جواب بهینه نرسیده‌اند (خط نقطه چین: تعداد مسائل با خطای همگرایی در مقدار تابع هدف، خط توپر: تعداد مسائل دارای تعدی از قیود)



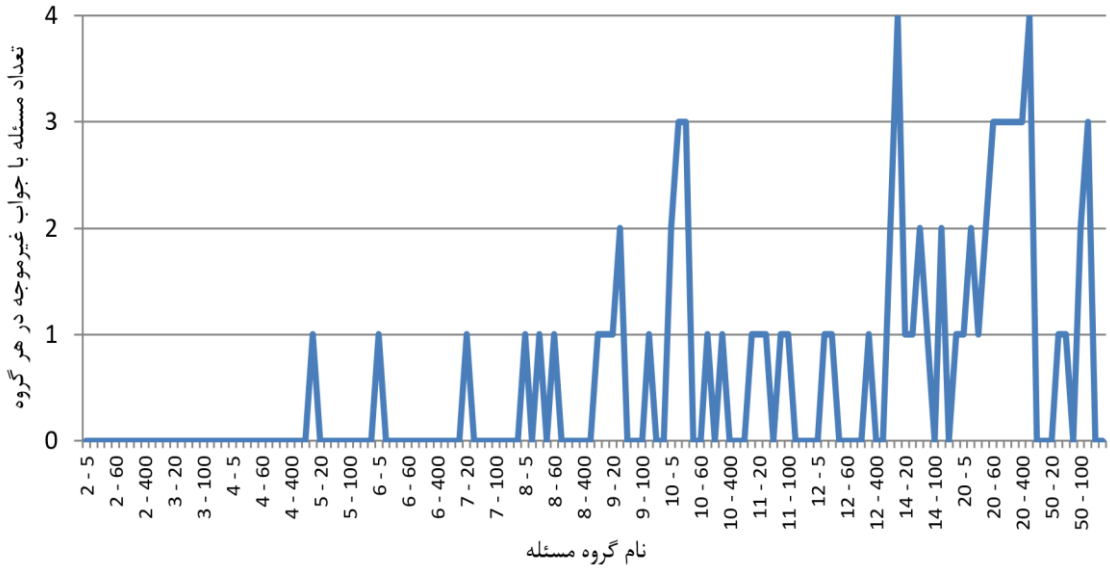
شکل ۴: ب- تعداد مسائل با ۲۰ و ۵۰ متغیر تصمیم که به دلیل خطای همگرایی به جواب بهینه نرسیده‌اند (خط نقطه چین: تعداد مسائل با خطای همگرایی در مقدار تابع هدف، خط توپر: تعداد مسائل دارای تعدی از قیود)



شکل ۵: الف- میانگین شکاف نسبی بهینگی برای مسائل با تعداد متغیر ۲ تا ۱۴ که دارای خطای همگرایی هستند



شکل ۵: ب- میانگین شکاف نسبی بهینگی برای مسائل با تعداد متغیر ۲۰ و ۵۰ که دارای خطای همگرایی هستند



شکل ۶: تعداد کل مسائلی در هر گروه که به جواب غیرموجه همگرا شده‌اند

جدول ۱: مقایسه کارایی الگوریتم به ازای مسائل استاندارد پایگاه OR-Lib

n	m	SALCHOW		linprog	مقدار زیربهینگی SALCHOW
		زمان اجرا	مقدار تابع هدف	زمان اجرا	
۶	۱۰	۰٫۰۰۵۳۶	۲۳۴	۰٫۰۲۲۸۹	۰
۱۰	۱۰	۰٫۰۰۵۴۴	۱۹۹۳٫۱۱	۰٫۰۱۱۷	۱۴۹٫۲
۱۵	۱۰	۰٫۰۰۳۷۹	۶۳۰٫۱۴۳	۰٫۰۰۹۹۷	۰
۲۰	۱۰	۰٫۰۰۷۷۵	۶۴۰٫۵	۰٫۰۱۳۲۸	۰
۲۸	۱۰	۰٫۰۱۷۴۶	۱۸۵۲٫۷۷	۰٫۰۲۰۸۸	۰
۳۹	۵	۰٫۰۲۱۴۲	۳۷۳۲٫۷۸	۰٫۰۱۲۶۱	۵۲۱٫۴
۵۰	۵	۰٫۰۳۳۵۶	۲۸۲۴	۰٫۰۱۵۲۴	۲۸۱٫۲

زمانی که SALCHOW به جواب‌های غیرموجه و یا فرابهینه همگرا می‌شود، از طریق شرط‌هایی که در کد الگوریتم قرار گرفته است، اجرای SALCHOW متوقف می‌شود. این امر ممکن است باعث شود زمان متوسط زمان اجرای الگوریتم برای کل گروه مسئله کمتر گزارش شود. برای رفع این ابهام، شکل (۳-الف) میانگین زمان حل هر گروه را فقط برای مسائلی (با ۲ الی ۱۴ متغیر تصمیم) که بدون خطای همگرایی به جواب بهینه رسیده‌اند گزارش می‌دهد؛ شکل (۳-ب) نیز همین اطلاعات را برای مسائل با ۲۰ و ۵۰ متغیر تصمیم را گزارش می‌دهد. همان‌گونه که مشاهده می‌شود در صورت چشم‌پوشی از اجراهایی که به جواب غیرموجه یا غیربهینه همگرا شده‌اند، متوسط زمان اجرای SALCHOW در هر گروه مسئله کمتر از ۰,۰۲ ثانیه افزایش می‌یابد. البته این اختلاف نیز طبق شکل (۳-ب) فقط برای مسائلی با تعداد ۵۰ متغیر تصمیم روی می‌دهد.

دو ایراد ممکن است در جواب نهایی که SALCHOW تولید می‌کند روی دهد. یکی همگرا شدن به مقادیر تابع هدف زیربهینه. دیگری تعدی از محدودیت‌های فنی است. شکل (۴-الف) و (۴-ب) تعداد مسائلی که به جواب زیربهینه و غیرموجه همگرا شده‌اند به ترتیب برای مسائل با ۲ تا ۱۴ متغیر تصمیم، و ۲۰ الی ۵۰ متغیر تصمیم را گزارش می‌دهند. همان‌گونه که در این نمودارها مشاهده می‌شود، معمولاً تعداد خطای همگرایی به جواب‌های زیربهینه بیشتر از بروز خطای تولید جواب غیرموجه است. نکته دیگر اینکه تعداد همگرایی به جواب‌های زیربهینه برای مسائل با ۵۰ متغیر تصمیم بیش از ۵۰ درصد موارد است. به عنوان نمونه برای دسته مسائل ۵۰ متغیر - ۵ محدودیت از ۵۰ مسئله تصادفی حل شده، در ۴۹ مورد SALCHOW به جواب‌های زیربهینه همگرا شده است. این واقعیت لزوم تکمیل مراحل توسعه SALCHOW و بهبود الگوریتم کامپیوتری آن را پررنگ‌تر می‌کند. البته شایان ذکر است در همه مسائلی که SALCHOW به مقدار بهینه تابع هدف همگرا شده، از هیچ قیدی تعدی نشده است. همچنین در مسائلی هم که از قیود تعدی شده، SALCHOW یا به تابع هدفی بدتر و به ندرت به مقادیری بهتر از مقدار بهینه واقعی همگرا شده است.

وقتی که گفته می‌شود SALCHOW خطای همگرایی به تابع هدف بهینه دارد منظور ما مقادیر بسیار بزرگ نیست. بلکه متوسط میزان زیربهینگی در مسائل با ۲ الی ۱۴ متغیر تصمیم شکل (۵-الف) کمتر از ۲ درصد مشاهده شد. شکل (۵-ب) نیز نشان می‌دهد که متوسط میزان زیربهینگی در مسائل با ۲۰ متغیر تصمیم همچنان کمتر از ۲ درصد و قابل چشم‌پوشی است. اما در همین شکل زمانی که مقدار شکاف زیربهینگی را برای مسائل با ۵۰ متغیر تصمیم بررسی می‌شود مقدار چشمگیر تا حدود ۱۶ درصد دیده می‌شود. این واقعیت لزوم تکمیل مراحل توسعه SALCHOW و بهبود الگوریتم کامپیوتری آن برای حل مسائل با بیش از ۲۰ متغیر تصمیم را پر رنگ‌تر می کند.

خطای دیگری که گفته شد ممکن است SALCHOW مرتکب شود، همگرا شدن به جواب‌های غیرموجه است. شکل (۶) تعداد مسائل با جواب نهایی غیرموجه که توسط SALCHOW تولید شده‌اند را نمایش می‌دهد. همان‌گونه که دیده می‌شود در هر دسته مسئله که شامل ۵۰ مسئله تصادفی تولید شده است، همواره کمتر از ۴ مسئله منجر به جواب غیرموجه شده‌اند. این موضوع بیان می‌کند که خطای گردکردن اعداد که منجر به خطا رفتن الگوریتم توسعه داده می شود، تأثیر کمتری روی تولید جواب‌های غیرموجه دارد.

در دسته دوم مسائل نمونه برای ارزیابی کارایی SALCHOW از مسائل نمونه پایگاه OR-Library استفاده شد. برای این کار از سری مسائل mknap1 استفاده کردیم. این سری مسئله در بر گیرنده ۷ سری مسئله کوله پشتی چندبعدی[1] است. مدل ریاضی مسئله کوله پشتی چند بعدی در (۱۸) آورده شده است. متغیر تصمیم مسئله کوله پشتی چندبعدی، باینری است ولی در اینجا برای استفاده محاسباتی از قید باینری بودن متغیرهای تصمیم چشم پوشی گردید.

همان گونه که در جدول ۱ دیده می شود برای مسائل با ۶ الی ۲۸ متغیر تصمیم زمان اجرای SALCHOW کمتر از الگوریتم مبنا است. در مواجهه با مسائل ۵-۳۹ و ۵-۵۰ همان گونه که در دو سطر پایین جدول ۱ دیده می شود SALCHOW مزیت نسبی در کوتاه بودن زمان حل نسبت به الگوریتم مبنا را از دست می دهد. این امر به دلیل زیاد بودن تعداد متغیرهای تصمیم این مسائل (۳۹ عدد و ۵۰

[1] Multidimensional knapsack problem

عدد) است. از دست دادن مزیت سرعت اجرا در کنار همگرایی به جواب‌های زیربهینه برای این دو مسئله مشهود است. البته همگرایی به جواب های غیربهینه برای مسئله ۱۰-۱۰ در سطر دوم نیز اتفاق افتاده است.

۵- بحث و تحلیل

در این تحقیق الگوریتم توسعه داده شده به عنوان SALCHOW نام‌گذاری گردیده است؛ علت این امر همگرا شدن به مجموعه قیدهای فعال در گوشه بهینه و محاسبه ضرایب لاگرانژی مربوط به هر یک است. چون پیشرفت الگوریتم به صورت گام به گام است، مجموعه قیدهای فعال به تدریج شناسایی می‌شوند. به طور هندسی، نقطه جواب معادل هر مجموعه فعال روی مرز فضای موجه قرار دارد. از این رو حرکت بر روی مرز فضای جواب صورت می‌گیرد، و نام این الگوریتم "محاسبه متوالی ضرایب لاگرانژ با حفظ حرکت روی مرز فضای جواب" انتخاب گردید. الگوریتم توسعه داده شده ابتدا با داشتن یک نقطه‌ی شروع موجه اقدام به آغاز فرایند حل می‌کند. نحوه کار الگوریتم توسعه داده شده به این صورت است که در جهت بردار گرادیان تابع هدف حرکت می‌کند تا به یک محدودیت برسد. پس از مواجهه با قید محدود کننده‌ی میزان پیشرفت تابع هدف، این قید به مجموعه قیدهای فعال افزوده می‌شود. پس از به روز رسانی مجموعه قیدهای فعال (حذف قیدهای قدیمی و افزودن قیدهای جدید) باید جهت گرادیان تابع هدف به نوعی به روز رسانی گردد تا علاوه بر تضمین حرکت در راستای افزایش تابع هدف اصلی مسئله، در محدوده موجه قیدهای فعال باقی بمانیم. این چرخه آن قدر تکرار می‌گردد تا جهت گرادیان تابع هدف به روز شده، صفر گردد. در واقع صفر شدن بردار گرادیان تابع هدف به روز شده، به معنی برقراری شرایط KKT و یا اکسترمم شدن تابع لاگرانژی مسئله است. مهم‌ترین مزیت استفاده از SALCHOW در مقایسه با الگوریتم سیمپلکس عبارت است از چندجمله‌ای بودن مرتبه زمانی حل نسبت به تعداد متغیرهای تصمیم مسئله. البته این موضوع در اینجا بدون اثبات و فقط بصورت تجربی مشاهده شد که مرتبه زمانی اجرای SALCHOW مستقل از تعداد محدودیت‌های کارکردی مسئله است.

حرکت بر روی مرز موجه مسئله این امکان را فراهم می‌آورد که بتوان در تحقیقات آتی از SALCHOW برای حل مسائل برنامه‌ریزی ریاضی مقید خطی و تابع هدف غیرخطی محدب استفاده کرد.

مزیت دیگر SALCHOW نسبت به الگوریتم‌های نقطه داخلی، ماهیت دقیق آن است که به صورت حدی به گوشه بهینه همگرا نمی‌گردد؛ بلکه به صورت دقیق و با شناسایی مجموعه قیدهای فعال در گوشه بهینه این کار را انجام می‌دهد. البته همگرایی SALCHOW موضوع در اینجا بدون اثبات و فقط به صورت تجربی مشاهده شد. و در ضمن زمان اجرای آن و حتی آماده‌سازی مسائل برای آنکه توسط این الگوریتم حل شوند به هیچ وجه همچون رقبای نقطه داخلی‌اش غیر کارا نیست. مهم‌ترین کاستی که حرکت بر روی مرز فضای موجه برای یک الگوریتم عددی می‌تواند ایجاد کند، خطای خروج از ناحیه جواب است که SALCHOW از این خطا مستثنی نیست. از سوی دیگر الگوریتم توسعه داده شده فقط قادر به حل مسائل با جواب موجه اولیه و فقط مسائل با قیدهای از نوع کوچک‌تر - مساوی است؛ که این موضوع، کاربرد آن را محدود می‌کند. هرچند بدون اثبات و فقط بصورت تجربی مشاهده شد زمان حل SALCHOW در حالت متوسط و یا حتی بدترین حالت ممکن، به صورت خطی و یا چندجمله‌ای از تعداد متغیرهای تصمیم افزایش می‌یابد، می‌توان امیدوار بود در صورت رفع اشکالات کدنویسی بتوان از آن برای حل مسائل برنامه‌ریزی خطی عمومی به عنوان جایگزینی برای سیمپلکس استفاده کرد. البته به دلیل حرکت بر روی مرز فضای موجه و محاسبه گسسته بردار گرادیان تابع هدف به سادگی می‌توان از این الگوریتم برای حل مسائل دارای قیود خطی و تابع هدف محدب غیرخطی استفاده کرد. عنوان موضوع دیگری که نویسندگان برای تحقیقات آتی به دنبال آن هستند، ارائه شکل جدولی برای پیاده‌سازی SALCHOW است تا ضمن حفظ مرتبه زمانی قابل قبول، آموزش و تفهیم الگوریتم را ساده‌تر گرداند. برای تحقیقات آتی مهم‌ترین سؤالی که در این تحقیق به آن پرداخته نشد، اثبات همگرایی الگوریتم SALCHOW است که البته با تطبیق دادن گام‌های آن با روش لاگرانژ، رویه اثبات سرراست خواهد بود. موضوع بعدی که در این تحقیق بدیهی فرض گردید، برقراری شرایط KKT در نقطه توقف الگوریتم و اثبات بهینگی جواب‌های نهایی تولید شده توسط SALCHOW است، که باید به صورت رسمی ارائه گردد. البته اثبات اینکه مرتبه زمانی حل نسبت به تعداد

متغیرهای تصمیم چندجمله‌ای است در اینجا به صورت شهودی مورد مشاهده قرار گرفت، اما برای اثبات تحلیل نیاز به تحقیق بیشتری احساس می‌گردد. همان‌طور که گفته شد، الگوریتم SALCHOW فقط برای مسائل خطی با قیود کوچک‌تر - مساوی و جواب اولیه داده شده توسعه داده شد. اما ارائه الگوریتم‌های منشعب از آن برای حل سایر انواع مسائل مثلا فاقد جواب اولیه موجه و یا سایر انواع قیدها (مساوی یا بزرگتر - مساوی) تا "تابع هدف غیرخطی و قیود خطی" و یا کلاً مسائل غیرخطی عمومی می‌تواند بسیار کاربردی و راهگشا باشد.

۶- جمع‌بندی و نتیجه‌گیری

در این مقاله یک الگوریتم برای حل مسائل برنامه‌ریزی خطی ارائه شد که برای رسیدن به بهینگی، برخلاف روش سیمپلکس بر روی نقاط رأسی فضای موجه حرکت نمی‌کند و برخلاف روش‌های نقطه داخلی هم وارد فضای موجه نمی‌شود. بلکه بر روی وجه‌های فضای موجه و در راستای تصویر گرادیان مقید تابع هدف حرکت می‌کند. علی‌رغم اینکه نتایج محاسباتی موجود برای مجموعه محدودی از مسائل نمونه کوچک رفتار زمانی بهتری نسبت به سیمپلکس نشان می‌دهد، اما عدم اثبات هم‌گرایی و وجود برخی خطاهای عددی در محاسبه مقدار جواب بهینه نهایی مواردی هستند که نیاز به تحقیقات بیشتر در این زمینه را طلب می‌کنند.

۷- مراجع

[1] G.B. Dantzig, Linear Programming and Extensions, Princeton Univ. Press, Princeton, N.J., 1963.

[2] N. Karmarkar, "A new polynomial-time algorithm for linear programming", Combinatorica Vol. 4, Issue 4, 1984, pp. 373–395.

[3] V. Klee, G.J. Minty, how good is the simplex algorithm in Inequalities III (Ed. O. Shisha), pp. 159-175, Academic Press, New York, 1972.

[4] G. Strang, the simplex method and Karmarkar's method, In Introduction to Applied Mathematics pp. 673-691. Wellesley-Cambridge Press, Wellesley, Mass, 1986.

[5] M. Zeleny, "An external reconstruction approach to linear programming", Computers and Operations Research, Vol. 13, Issue 1, 1986, pp. 95–100.

[6] G. Mitra, M. Tamiz, J. Yadegar, "Experimental investigation of an interior search method within a simplex framework", Commans ACM, Vol. 31, No. 12, 1988, pp. 1474–1482.

[7] J.A. Snyman, "An Interior Feasible Direction Method with Constraint Projections for Linear Programming", Computers & Mathematics with Applications, Vol. 20, Issue 12, 1990, pp. 43–54.

[8] G. Zoutendijk, Methods of Feasible Directions: A Study in Linear and Nonlinear Programming, Elsevier, Amsterdam and D Van Nostrand, Princeton, N.J., 1960.

[9] J.B. Rosen, "The gradient projection method for nonlinear programming, Part I, Linear constraints", SIAM J. Appl. Math, Vol. 8, No. 1, 1960, pp. 181–217.

[10] P. Wolfe, Methods of nonlinear programming. In Nonlinear Programming (Ed. J. Abadie), pp. 97–131. North-Holland, Amsterdam, 1967.

[11] W.I. Zangwill, "The Convex Simplex Method", Management Science, Vol. 14, No. 3, 1967, pp. 221–238.

[12] M.A. Diniz-Ehrhardt, M.A. Gomes-Ruggiero, J.M. Martinez, S.A. Santos, "Augmented Lagrangian Algorithms Based on the Spectral Projected Gradient Method for Solving Nonlinear Programming Problems", Journal of optimization theory and applications, Vol. 123, Issue 3, 2004, pp. 497–517.

[13] P.H. Calamai, J.J. Mori. "Projected Gradient Methods for Linearly Constrained Problems", Mathematical Programming, Vol. 39, Issue 1, 1987, pp. 93–116.

[14] D.P. Bertsekas, "Projected Newton Methods for Optimization Problems with Simple Constraints", Siam J. Control and Optimization, Vol. 20. No. 2, 1982, pp. 221–246.

[15] J.P. Dussault, G. Fournier, "Technical Note on the Convergence of the Projected Gradient Method", Journal of Optimization Theory and Applications, Vol. 77, No. 1, 1993, pp. 197–208.

[16] J.B. Rosen, "The Gradient Projection Method for Nonlinear Programming. Part II. Nonlinear Constraints", J. Soc. Indust. Appl. Math., Vol. 9, No. 4, 1961, pp. 514–532.

[۱۷] ح. نورمحمدی، "بررسی روش زوتندیک در مسایل برنامه‌ریزی خطی"، مجله ریاضیات کاربردی، سال ششم، شماره ۲۳، ۱۳۸۸، صفحه ۶۹-۷۲.

[18] M. Hintermuller, K. Ito, K. Kunisch, "The Primal-Dual Active Set Strategy as a Semismooth Newton Method", Siam J. Optim., Vol. 13, No. 3, 2003, pp. 865–888.

[19] R. Andreani, J.J. Júdice, J.M. Martínez, J. Patrício, "A projected–gradient interior–point algorithm for complementarity problems", Numerical Algorithms, Vol. 57, Issue 4, 2011, pp. 457–485.

[20] L.M. Grana Drummond, "A Projected Gradient Method for Vector Optimization Problems", Computational Optimization and Applications, Vol. 28, Issue 1, 2004, pp. 5–29.

[21] F.A. Potra, S.J. Wright, "Interior-Point Methods", Journal of Computational and Applied Mathematics, Vol. 124, Issues 1–2, 2000, pp. 281–302.

[22] J. Gondzio, "Interior Point Methods 25 Years Later", European Journal of Operational Research, Vol. 218, Issue 3, 2012, pp. 587–601.

[23] Y-H. Xiao, Q-J. Hu, Z. Wei, "Modified Active Set Projected Spectral Gradient Method for Bound Constrained Optimization", Applied Mathematical Modeling, Vol. 35, Issue 7, 2011, pp. 3117–3127

[24] A. Deza, E. Nematollahi, R. Peyghami, T. Terlaky, "The Central Path Visits All the Vertices of the Klee–Minty Cube", Optimization Methods and Software, Vol. 21, No. 5, 2006, pp. 851–865.

[25] A. Deza, E. Nematollahi, T. Terlaky, "How Good Are Interior Point Methods? Klee–Minty Cubes Tighten Iteration-Complexity Bounds", Mathematical Programming, Vol. 113, No. 1, 2008, pp. 1–14.

[26] F.S. Hillier, G.J. Lieberman, Introduction to Operations Research, 9th Edition, McGraw-Hill, New York, United States, 2010, pp. 142–143.

[27] E.G. Birgin, J.M. Martinez, "Large-Scale Active-Set Box-Constrained Optimization Method with Spectral Projected Gradients", Computational Optimization and Applications, Vol. 23, Issue 1, 2002, pp. 101–125.

[28] M. Andretta, E.G. Birgin, J.M. Martinez, "Partial spectral projected gradient method with active-set strategy for linearly constrained optimization", Numerical Algorithms, Vol. 53, No. 1, 2010, pp. 23–52.