

روش بهروزسانی متقاض از مرتبه-اول برای حل مسائل بهینه‌سازی مقیاس بزرگ

فرزین مدرس خیابانی^{۱*}، بهروز دانشیان^۲

۱- استادیار گروه ریاضی، واحد تبریز، دانشگاه آزاد اسلامی، تبریز، ایران

۲- دانشیار گروه ریاضی، واحد تهران مرکز، دانشگاه آزاد اسلامی، تهران، ایران

رسید مقاله: ۱ دی ۱۳۹۵

پذیرش مقاله: ۹ مرداد ۱۳۹۶

چکیده

جستجو جهت یافتن کمینه موضعی در مسائل بهینه‌سازی نامقید و یک نقطه ثابت از دستگاه گرادیان معادلات دیفرانسیل معمولی دو مساله نزدیک به هم می‌باشد، الگوریتم‌های با حافظه محدود به طور گسترده‌ای جهت حل مسائل مقیاس بزرگ استفاده می‌شوند؛ در حالی که روش‌های رانگ کوتا نیز برای حل عددی معادلات دیفرانسیل مورد استفاده قرار می‌گیرند. در این تحقیق با استفاده از ایده روش زیر فضا و طول گام ثابت و ادغام تکنیک‌های جستجوی خطی و ناحیه مطمئن، یک روش پیوندی مبتنی بر ODE برای حل مسائل بهینه‌سازی مقیاس بزرگ ارایه شده است. با توجه به اینکه روش‌های جستجوی خطی ممکن است نیازمند تکرارهای بیشتری برای همگرایی باشند؛ در حالی که روش‌های ناحیه مطمئن نیز نیازمند تکرارهای زیادی برای حل زیر مساله مقيید باشند، کلاس جدیدی از روش‌ها طوری پیشنهاد شده، که بتواند بهترین ویژگی‌های روش‌های ناحیه مطمئن و جستجوی خطی را با هم ترکیب کند، ویژگی اصلی روش پیشنهادی این است که دستگاه معادلات خطی فقط یکبار جهت به دست آوردن گام آزمایشی حل می‌شود. علاوه بر این، در صورتی که گام آزمایشی مورد قبول نگیرد این روش از جستجوی خطی بهره می‌جويد. نتایج یک سری از آزمون‌ها بر روی مسائل بهینه‌سازی نامقید استاندارد گزارش شده‌است. این نتایج عددی نشان دهنده مؤثر بودن الگوریتم جدید برای حل مسائل مقیاس بزرگ می‌باشد.

واژگان کلیدی: بهینه‌سازی نامقید، معادلات دیفرانسیل معمولی، روش‌های با حافظه محدود، جستجوی-خطی، ناحیه مطمئن

۱ مقدمه

در این مقاله، مسائل بهینه‌سازی نامقید به شکل کلی:

$$\underset{x \in R^n}{\text{Min}} f(x), \quad (1)$$

* عهده‌دار مکاتبات

آدرس الکترونیکی: f.modarres@iaut.ac.ir

در نظر گرفته می‌شود به طوری که $\nabla f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ یک تابع به‌طور پیوسته مشتق پذیر، $f: \mathbb{R}^n \rightarrow \mathbb{R}$ گرادیان f و $\nabla f: \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n}$ ماتریس هسیان f می‌باشند. در سراسر این مقاله، $\|\cdot\|$ نرم اقلیدسی روی \mathbb{R}^n را نشان می‌دهد.

الگوریتم‌های بهینه‌سازی با شروع از نقطه داده شده x دنباله‌ای از تکرارهای $\{x_k\}$ را تولید می‌کنند. این دنباله تولید شده زمانی متوقف می‌شود که یا هیچ بهبود در جواب مشاهده نشود و یا اینکه جواب تقریبی با دقت کافی به دست آید. در نحوه تصمیم‌گیری برای حرکت از تکرار x_k به تکرار بعدی، الگوریتم‌ها از اطلاعات f در x_k (و احتمالاً از اطلاعات تکرارهای قبلی x_1, \dots, x_{k-1}) استفاده می‌کنند. الگوریتم‌ها این اطلاعات را برای یافتن تکرار جدید x_{k+1} که مقدار کمتری در مقایسه با x_k دارد، مورد استفاده قرار می‌دهند.

الگوریتم ناحیه مطمئن^۱ و الگوریتم جستجوی خطی^۲ دو گروه عمده از الگوریتم‌ها می‌باشند که برای حرکت از نقطه فعلی x_k به نقطه جدید x_{k+1} ، مورد استفاده قرار می‌گیرند. روش‌های ناحیه مطمئن بر مبنای جستجویی هستند که یک تکرار جدید را در داخل یک ناحیه چند بُعدی محاط به نقطه فعلی تولید می‌کنند. تکرارهایی که توسط الگوریتم ناحیه مطمئن تولید می‌شوند در اغلب آن‌ها f به سرعت کاهش می‌یابد. روش‌های جستجوی خطی، هر تکرار را با جستجویی برای یک مقدار قابل قبول از x در راستای خطی که از تکرار قبلی می‌گذرد تکرار جدید تولید می‌کند. روش‌های ناحیه مطمئن به عنوان روش‌هایی با گام محدود نیز معروفند. این روش اولین بار برای حل مسایل غیرخطی کمترین مربعات پیشنهاد گردید. در هر تکرار از روش‌های ناحیه مطمئن، به حل زیر مسئله نیاز داریم.

هر دو استراتژی فوق تکرار جدیدی را با کمینه کردن یک مدل درجه دوم از تابع f تعریف می‌کنند. روش‌های ناحیه مطمئن و جستجوی خطی امکان دارد با مشکل مواجه شود نزدیک نقاطی که ماتریس مدل منفرد باشد. در چنین حالت‌هایی، روش‌های جستجوی خطی ممکن است نیازمند تکرارهای زیادی جهت همگرایی باشد از طرفی روش‌های ناحیه مطمئن نیز نیازمند تکرارهای زیادی جهت حل زیر مساله باشند. برای این منظور کلاس جدیدی از روش‌ها که بهترین ویژگی‌های روش‌های جستجوی خطی و ناحیه مطمئن را استفاده می‌کند، لحاظ شده است. روش‌های جستجوی-مطمئن همگرایی سریع‌تری در مقایسه با روش‌های ناحیه مطمئن به دست می‌دهند، و زیر مساله را با هزینه کمتری نسبت به روش جستجوی خطی حل می‌کنند. این روش‌ها برای حل مسایل بهینه‌سازی مقیاس بزرگ و دستگاه معادلات غیرخطی مناسب می‌باشند.

در طول سال‌های اخیر روش‌های مبتنی بر معادلات دیفرانسیل معمولی برای حل مسایل بهینه‌سازی غیرخطی توجه محققان زیادی را جلب نموده است (مراجعه شود به [۷، ۶، ۵، ۴، ۳، ۲، ۱]). براون و همکاران^۳ [۵] نشان دادند که وقتی روش‌های ODE در مسایل بهینه‌سازی نامقید به‌طور مناسب اجرا شود می‌توان مقایسه بسیار مطلوبی را بین

¹ Trust- region² Line-search³ Brown et al.⁴ Narrow curved valley

الگوریتم‌های نیوتن و شبیه نیوتن از لحاظ قابلیت اطمینان، دقت و کارایی به ویژه برای مسایل مینیمم‌سازی غیرخطی در قسمت‌های باریک منحنی^۱ انجام داد.

از بین تمامی روش‌های مبتنی بر ODE موفق‌ترین الگوریتم، الگوریتم IMPBOT می‌باشد [۶۸]. این روش برای مسایل در مقیاس کوچک قابل استفاده است؛ اما اگر تعداد متغیرها زیاد باشد حل دوباره دستگاه معادلات خطی می‌تواند به طور قابل ملاحظه‌ای هزینه محاسبات را افزایش دهد بنابراین هدف اصلی در طراحی مدل‌های مبتنی بر^۲ ODE، ساده‌سازی دستگاه معادلات خطی و اجتناب از حل دوباره آن در یک تکرار می‌باشد. در سال‌های اخیر روش‌های پیوندی در این راستا ارایه شده‌اند که روش‌های ناحیه مطمئن را با روش‌های جستجوی خطی ادغام می‌کنند [۸،۹]. یکی از مزایای روش‌های جدید این است که وقتی گام آزمایشی مورد قبول واقع نشد یک جستجوی خطی جهت یافتن یک نقطه تکرار به جای حل دوباره زیرمسئله ناحیه مطمئن انجام می‌شود؛ بنابراین این روش‌ها نیاز به محاسبات کم‌تری نسبت به روش‌های ناحیه مطمئن کلاسیک دارند. با این حال روشن است که وقتی f درجه دو یا به مقدار زیاد غیرخطی باشد پیدا کردن طول گام مناسب در جهت جستجو دشوار می‌باشد و لذا گام به دست آمده ممکن است خیلی کوچک باشد (برای مثال وقتی که نقطه فعلی در قسمت‌های باریک منحنی قرار بگیرد). برای غلبه بر این مشکل، روش طول گام ثابت معرفی شده است (مراجعه شود به [۱۰، ۱۱، ۱۲، ۱۳]).

لازم به ذکر است که روش‌های ناحیه مطمئن زیرفضا برای مسایل بهینه‌سازی نامقید معرفی شده‌اند [۱۴، ۱۵]، به طوری که در هر تکرار زیر مسئله ناحیه مطمئن در یک زیر فضای ساده حل می‌شود؛ بنابراین در کل، کار محاسباتی در فضای حافظه مورد نیاز به طور قابل توجهی کاهش می‌یابد [۱۶]. مشکلی که در این کلاس از روش‌ها وجود دارد این است که زیر فضای گام آزمایشی یا ناحیه مطمئن متعلق به آن انتخاب کرد. مفروضات بالا موجب می‌شود که یک روش پیوندی جدید مبتنی بر ODE برای بهینه‌سازی نامقید ارایه شود به طوری که در آن ایده الگوریتم IMPBOT با روش زیرفضا و طول گام ثابت ادغام می‌شود. اروی و مارسیا^۳ [۱۷] مسئله حل دستگاه معادلات خطی را با مؤلفه‌های با حافظه محدود در نظر گرفته و رویکرد جدیدی را بر مبنای اجرای عملی از نمایش فشرده برای معکوس این ماتریس‌های با حافظه محدود را ارایه دادند. نتایج عددی نشان داد که روش پیشنهادی در سرعت و دقت با سایر الگوریتم‌ها به خصوص اعضای خانواده برویدن قبل رقابت هست. برخلاف روش‌های موجود در متابع [۳، ۵، ۷، ۱۸، ۱۹، ۲۰]، دستگاه معادلات خطی با بعد کم را فقط یک بار حل می‌کند؛ بنابراین کار محاسباتی کاهش می‌یابد. مزیت دیگری که دارد این است که وقتی گام آزمایشی مورد قبول واقع نشد، روش مذکور دستگاه معادلات خطی را دوباره حل نمی‌کند؛ بلکه یک نقطه تکرار ایجاد می‌کند که طول گام آن توسط فرمولی محاسبه شود؛ بنابراین از انجام یک جستجوی خطی برای محاسبه طول گام اجتناب می‌شود. به علاوه لازم نیست که تقریب ماتریس هسیان (x) f معین مثبت یا نیمه‌معین مثبت باشد.

² Ordinary differential equations

³ Erway and Marcia

کارایی روش ارایه شده در محاسبات عملی و نیز آزمون‌های عددی انجام یافته، شرح داده شده است. بر این اساس می‌توان گفت ایده‌ی اصلی این مقاله استفاده از روش جستجوی مطمئن برای حل مسایل بهینه‌سازی مقیاس بزرگ مبتنی بر ODE می‌باشد.

ساختمار این مقاله به صورت زیر می‌باشد: در قسمت ۲، ابتدا استراتژی ناحیه مطمئن را بررسی کرده و سپس نحوه ادغام روش‌های ناحیه مطمئن با روش‌های جستجوی خطی مورد بحث قرار می‌گیرد. قسمت سوم، جستجوی-مطمئن پیشنهادی را ارایه می‌کند برای این منظور ابتدا نمایش فشرده‌ای از الگوریتم به روزرسانی متقارن از رتبه‌ی یک را خواهیم داشت و سپس الگوریتم پیشنهادی ارایه خواهد شد. در نهایت نتایج عددی به دست آمده از اجرای الگوریتم پیشنهادی بر روی مجموعه توابع آزمون استاندارد در قسمت ۴ ذکر می‌شود.

۲ روش ناحیه مطمئن

روش ناحیه مطمئن برای حل مسأله (۱) یک گام آزمایشی را با حل زیر مسأله :

$$\begin{aligned} \text{Min}_{d \in \mathbb{R}^n} g_k^T d + \frac{1}{2} d^T B_k d &\equiv \phi_k(d) \\ \text{s.t. } \|d\| &\leq \Delta_k, \end{aligned} \quad (2)$$

محاسبه می‌کند که $g_k = \nabla f(x_k)$ گرادیان تابع هدف در جواب تقریبی فعلی است و B_k یک ماتریس متقارن $n \times n$ است که تقریب هسیان f است و Δ_k شعاع ناحیه مطمئن می‌باشد. ناحیه مطمئن، ناحیه‌ای حول نقطه تکرار فعلی می‌باشد. انتخاب‌های متفاوت از ناحیه مطمئن و مدل‌های متفاوت، الگوریتم‌های ناحیه مطمئن متفاوتی را ارایه می‌دهد. یکی از مزایای روش‌های ناحیه مطمئن این است که در مقایسه با روش‌های جستجوی خطی B_k می‌تواند نامعین باشد و به وسیله فرمول شبه نیوتن به روز رسانی شود.

بعد از به دست آوردن گام آزمایشی d که یک جواب تقریبی یا دقیق زیرمسأله (۲) است، الگوریتم‌های

$$\text{ناحیه مطمئن با محاسبه نسبت} \frac{\text{کاهش واقعی}}{\text{کاهش پیش شده بینی}} = \frac{f(x_k) - f(x_k + d_k)}{\phi_k(0) - \phi_k(d_k)} = \rho_k \text{ شعاع ناحیه مطمئن} \Delta_k \text{ را}$$

مطابق با مقدار ρ_k به روز رسانی می‌کند. حال اگر گام d موفق نباشد؛ یعنی اگر $f(x_k + d_k) > f(x_k)$ گام پذیرفته نمی‌شود و $x_{k+1} = x_k$ قرار می‌دهد و شعاع ناحیه مطمئن را کاهش داده و دوباره زیرمسأله (۲) را حل می‌کند.

۱-۲ ادغام روش‌های ناحیه مطمئن با جستجوی خطی

حل زیرمسأله ناحیه-مطمئن برای مسایل با مقیاس کوچک کاملاً مناسب است. با این حال اگر تعداد متغیرها زیاد باشد حل دوباره زیرمسأله ناحیه مطمئن می‌تواند پرهزینه باشد؛ زیرا به یک یا بیش از یک دستگاه خطی به شکل $(B_k + \lambda I)d = -g_k$ نیاز دارد؛ ولی روش‌های جستجوی خطی به محاسبات خیلی کمتری برای تعیین یک نقطه آزمایشی جدید نیاز دارند.

بنابراین هدف این است که بتوانیم جستجوی خطی بازگشتی را در یک روش ناحیه مطمئن ادغام کنیم به طوری که از حل دوباره زیر مساله وقتی که گام مورد قبول قرار نمی‌گیرد، اجتناب کنیم. اگرچه ممکن است استفاده از روش‌های جستجوی خطی خواص همگرایی الگوریتم را تضعیف کند؛ بنابراین با دو حالت مشکل‌ساز رو به رو هستیم:

۱- وقتی که جهت جستجو در یک الگوریتم جستجوی خطی متعامد به جهت تندترین کاهش g_k - باشد یک طول گام خیلی کوچک می‌تواند گام مورد قبول را به دست آورد. در برخی حالت‌ها خطاهای گرد کردن ممکن است باعث شود که جستجوی خطی شکست بخورد. در موقع مشابه الگوریتم ناحیه مطمئن، ناحیه مطمئن را کاهش خواهد داد و گام آزمایشی جدید به جهت تندترین کاهش میل خواهد کرد. این خصیصه باعث می‌شود که الگوریتم نسبت به انتشار خطای گرد کردن بیشتر پایدار باشد.

۲- وقتی که جهت جستجو در یک الگوریتم جستجوی خطی یا گام آزمایشی در یک روش ناحیه مطمئن خیلی بزرگ باشد که ممکن است به وسیله یک ماتریس خیلی کوچک B_k ایجاد شود. در این حالت با کاهش ناحیه مطمئن، گام آزمایشی به دست آمده به جهت گام آزمایشی ناموفق قبلی نزدیک خواهد شد. در این حالت روش ناحیه مطمئن مانند روش جستجوی خطی بازگشتی رفتار خواهد کرد؛ اما هزینه محاسباتی افزایش خواهد یافت. پس در این حالت بهتر است یک جستجوی خطی بازگشتی انجام گیرد.

در نتیجه در روش بازگشتی باید جهت جستجو به حد کافی کاهشی باشد. می‌توان این روش را در امتداد یک خط مستقیم یا در امتداد یک مسیر منحنی وار انجام داد. پس یک راه حل برای مسئله (۲) پیدا کردیم که گام آزمایشی d_k - همیشه یک جهت به اندازه کافی کاهشی برای تابع هدف می‌باشد. این بدان معنی است که زاویه بین d_k و g_k - بیش از $\frac{\pi}{2}$ خواهد بود هرگاه g_k به صفر نزدیک شود و $\|d_k\|$ و $\|B_k\|$ از بالا کراندار باشند. این خصیصه باعث می‌شود نتایج همگرایی راضی کننده‌ای به دست آید.

۳ روش پیوندی جستجوی-مطمئن

در این قسمت، با استفاده از روش مبتنی بر ODE برای مسایل بهینه‌سازی نامقید، ارایه شده توسط مدرس و لیانگ^۱[۲۱]، روش زیر فضای ادغام تکنیک‌های جستجوی خطی و ناحیه مطمئن، الگوریتمی جدیدی را ارایه خواهیم کرد که جهت حل مسئله (۱) به کار گرفته خواهد شد و در ادامه ویژگی‌های روش پیشنهادی مورد بحث قرار خواهد گرفت. مدرس و لیانگ، روش پیوندی مبتنی بر ODE برای حل مسایل بهینه‌سازی نامقید، که از تکنیک رانگ-کوتا با مرتبه-کم استفاده می‌کنند، ارایه داده‌اند. الگوریتم ارایه شده در [۲۱] اساساً از استراتژی جستجوی خطی استفاده می‌کند. با توجه به مزايا و معایب هر یک از استراتژی‌های جستجوی خطی و ناحیه مطمئن، به ویژه موفقیت پیاده‌سازی روش SR1 به نگام استفاده از روش ناحیه مطمئن در این بخش الگوریتمی را ارایه خواهیم داد تا ضمن استفاده از مزاياي هر یک از اين روش‌ها، با استفاده از الگوریتم بازگشتی دو حلقه‌اي با

^۱Leong

حافظه محدود SR1 (L-SR1) آن را جهت حل مسایل بهینه‌سازی مقیاس-بزرگ استفاده کنیم. با توجه به اینکه هدف اصلی این تحقیق بررسی نتایج به دست آمده از پیاده‌سازی الگوریتم ادغامی جستجوی خطی و ناحیه مطمئن بر روی فرمول بهروزرسانی SR1 می‌باشد ابتدا در خصوص روش SR1 و مزایای استفاده از این روش مطالبی را مروار خواهیم کرد و در ادامه نمایش فشرده‌ای از این فرمول بهروزرسانی را ارایه خواهیم داد تا در پیاده‌سازی الگوریتم اصلی مورد استفاده قرار گیرد.

۱-۳ ارایه نمایش فشرده‌ای از فرمول بهروزرسانی متقارن از رتبه - یک

این روش از جمله روش‌های از مرتبه یک می‌باشد تغییراتی را از رتبه یک به تقریب هسیان قبلی؛ یعنی B_k می-دهد. بهروزرسانی SR1 از حل فرم کلی زیر به دست می‌آید:

$$B_{k+1} = B_k + u v^T; \quad s. t. \quad B_{k+1} s_k = y_k.$$

حل مساله فوق، فرمول معروف به روزرسانی SR1 را به صورت زیر به دست می‌آورد:

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k},$$

با استفاده از فرمول شرمن-موریسن می‌توان فرمول بهروزرسانی SR1 را جهت تقریب معکوس هسیان به صورت زیر محاسبه کرد:

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{y_k^T (s_k - H_k y_k)^T}.$$

از جمله مزایای این روش این است که تعداد تکرار کمتری را در مقایسه با سایر روش‌های شبه-نیوتن از خود نشان می‌دهد. علاوه بر این بسیار آسان‌تر برای پیاده‌سازی استفاده می‌شود؛ زیرا از ماتریس رتبه یک جهت به روزرسانی استفاده می‌کند. همچنین این روش در چارچوب تکنیک‌های ناحیه مطمئن عملکرد بسیار قوی دارد. یکی از ویژگی‌های جالب این روش اتمام آن در حداکثر n مرحله برای توابع درجه دوم محدب می‌باشد که تکرارها خاصیت به طور خطی مستقل را داشته باشند. از جمله معایب این روش این است که این فرمول به روزرسانی ناپایدار می‌باشد از این جهت که معین مثبت بودن B_{k+1} را تضمین نمی‌کند؛ حتی اگر ماتریس تقریب شده هسیان فعلی معین مثبت باشد. همچنین مخرج کسر در این روش می‌تواند صفر و یا نزدیک به صفر باشد که منجر به ناپایداری عددی می‌شود. به دست آوردن تقریبات معین مثبت به ماتریس هسیان تابع هدف خلاهایی را دارد که ریسک از دست دادن اطلاعات با ارزش به دست آمده از تکرارهای کاهشی قبلی را به همراه دارد؛ بنابراین باید استراتژی‌هایی را اتخاذ کنیم که علاوه بر حفظ معین مثبت بودن B_{k+1} اطلاعات به دست آمده قبلی را از دست ندهیم.

روش‌های نیوتن و شبه-نیوتن دست کم به n^2 مکان حافظه جهت ذخیره یک ماتریس $n \times n$ نیازمند هستند، از آنجا که در روش‌های شبه نیوتن تقریب‌ها معمولاً برای هسیان یا معکوس آن متراکم شده‌اند، به طور مستقیم برای مسایل بهینه‌سازی بزرگ قابل اجرا نمی‌باشند. روش‌های شبه-نیوتن با حافظه محدود به عنوان روشی

مؤثر و برای حل دسته خاصی از مسایل بهینه‌سازی نامقید در مقیاس بزرگ طراحی گردیده است که در آن بیشتر مکان‌های حافظه $O(n)$ به ذخیره بردارهای n -بعدی نیاز دارند؛ لذا در این قسمت ارایه فشرده‌ای از فرمول بهروزرسانی SR1 را خواهیم داشت.

برای بهروزرسانی ماتریس $B_{i,k}(\lambda)$ که تقریبی برای ماتریس هسیان است از روند ذیل استفاده می‌کنیم:

$$B_{i,k+1}(\lambda) = B_{i,k}(\lambda) + \frac{(Y_{i,k}(\lambda) - B_{i,k}(\lambda)s_k)(Y_{i,k}(\lambda) - B_{i,k}(\lambda)s_k)^T}{(Y_{i,k}(\lambda) - B_{i,k}(\lambda))^T s_k},$$

و معکوس تقریب ماتریس هسیان نیز می‌تواند توسط رابطه‌ی زیر به دست می‌آید:

$$H_{i,k+1}(\lambda) = H_{i,k}(\lambda) + \frac{(s_k - H_{i,k}(\lambda)Y_{i,k}(\lambda))(s_k - H_{i,k}(\lambda)Y_{i,k}(\lambda))^T}{(s_k - H_{i,k}(\lambda)Y_{i,k}(\lambda))^T Y_{i,k}(\lambda)}, \quad (3)$$

که در آن:

$$H_{i,k+1}(\lambda_{k+1})Y_{i,k}(\lambda_{k+1}) = s_k,$$

$$Y_{i,k}(\lambda) := \frac{\lambda}{\mu_i} s_j + y_j, \quad i = 1, \dots, s.$$

و اگر $\lambda_{k+1} > 0$ ، $s_k^T y_k > 0$ داریم:

$$s_k^T Y_{i,k}(\lambda_{k+1}) > 0.$$

حال اگر قرار دهیم:

$$a = Y_{i,k}(\lambda)^T [B_{i,k}(\lambda)]^{-1} Y_{i,k}(\lambda), \quad b = Y_{i,k}(\lambda)^T s_k.$$

آنگاه می‌توان رابطه‌ی بهروزرسانی معکوس تقریب ماتریس هسیان را به صورت زیر بازنویسی کرد:

$$H_{i,k+1}(\lambda) = W_k^T (Y_{i,k}(\lambda)) W_k + \rho_k s_k s_k^T,$$

که در آن:

$$\begin{cases} W_k = I - Y_{i,k}(\lambda) u_k^T, \\ u_k = \frac{1}{\sqrt{b_k(b_k - a_k)}} s_k + \frac{\sqrt{b_k - a_k} - \sqrt{b_k}}{a_k \sqrt{b_k - a_k}} H_{i,k}(\lambda) Y_{i,k}(\lambda), \\ \rho_k = (b_k)^{-1}. \end{cases}$$

برای m مشخص شده توسط کاربر و یک ماتریس اولیه مثبت معین روش به روزرسانی L-SR1 را می‌توان توسط روش زیر به دست آورد. در این روش برای $k+1 \leq m$ داریم:

$$H_{i,k+1}(\lambda) = (W_k^T W_{k-1}^T \cdots W_1^T W_0^T) H_i(\lambda) (W_0 W_1 \cdots W_{k-1} W_k)$$

$$+ (W_k^T W_{k-1}^T \cdots W_1^T) \rho_s s_s^T (W_1 \cdots W_{k-1} W_k)$$

⋮

$$+ W_k^T \rho_{k-1} s_{k-1} s_{k-1}^T W_k$$

$$+ \rho_k s_k s_k^T.$$

و برای $m > k+1$ روش L-SR1 را به صورت زیر خواهیم داشت:

$$H_{i,k+1}(\lambda) = (W_k^T W_{k-1}^T \cdots W_{k-m+1}^T) H_i(\lambda) (W_{k-m+1} \cdots W_{k-1} W_k)$$

$$+ (W_k^T W_{k-1}^T \cdots W_{k-m+1}^T) \rho_{k-m+1} s_{k-m+1} s_{k-m+1}^T (W_{k-m+1} \cdots W_{k-1} W_k)$$

⋮

$$+ W_k^T \rho_{k-1} s_{k-1} s_{k-1}^T W_k$$

$$+ \rho_k s_k s_k^T.$$

برای راحتی می‌توان از نماد $\hat{m} = \min\{k, m-1\}$ در روابط بالا استفاده کرد و آن را به صورت زیر نوشت:

$$H_{i,k+1}(\lambda) = (W_k^T W_{k-1}^T \cdots W_{k-\hat{m}}^T) H_i(\lambda) (W_{k-\hat{m}} \cdots W_{k-1} W_k)$$

$$+ (W_k^T W_{k-1}^T \cdots W_{k-\hat{m}+1}^T) \rho_{k-\hat{m}} s_{k-\hat{m}} s_{k-\hat{m}}^T (W_{k-\hat{m}+1} \cdots W_{k-1} W_k)$$

⋮

$$+ W_k^T \rho_{k-1} s_{k-1} s_{k-1}^T W_k$$

$$+ \rho_k s_k s_k^T.$$

اگر ماتریس اولیه‌ای که برای شروع تقریب‌ها؛ یعنی H استفاده می‌شود یک ماتریس مثبت معین باشد، آنگاه

می‌توان گفت که ماتریس‌هایی که روابط بالا را معرفی می‌کنند ماتریس‌های مثبت معین هستند.

با توجه به رابطه‌ی (۳) در می‌یابیم که اگر $(s_k - H_{i,k}(\lambda)Y_{i,k}(\lambda))^T Y_{i,k}(\lambda)$ برابر با صفر باشد، نمی‌توانیم از این

روش استفاده کنیم و اگر مقدار آن خیلی کوچک باشد ممکن است، واگرا شود. برای این منظور ما از روش

بدون حافظه برای به دست آوردن بهروزرسانی ماتریس تقریب استفاده می‌کنیم.

قضیه ۱.۳. پارامتر:

$$\sigma(A) = \frac{\zeta_{\max}}{\det(A)^{\frac{1}{n}}},$$

را که در آن ζ_{\max} بزرگ‌ترین مقدار ویژه ماتریس $n \times n$ است در نظر بگیرید. فرض کنید

$$\gamma_k = \frac{\mathbf{s}_k^T \mathbf{s}_k}{\mathbf{s}_k^T \mathbf{y}_k} - \left[\left(\frac{\mathbf{s}_k^T \mathbf{s}_k}{\mathbf{s}_k^T \mathbf{y}_k} \right)^r - \frac{\mathbf{s}_k^T \mathbf{s}_k}{\mathbf{y}_k^T \mathbf{y}_k} \right]^{\frac{1}{r}}. \quad (4)$$

آنگاه معکوس ماتریس SR1 که از به روز رسانی I / γ_k به دست آمده و به صورت زیر است:

$$\mathbf{H}_{k+1} = \gamma_k \mathbf{I} + \frac{(\mathbf{s}_k - \gamma_k \mathbf{y}_k)(\mathbf{s}_k - \gamma_k \mathbf{y}_k)^T}{\mathbf{y}_k^T (\mathbf{s}_k - \gamma_k \mathbf{y}_k)}$$

یک جواب منحصر به فرد:

$$\min \sigma(\mathbf{H}_{k+1}^{-1})$$

$$\text{s.t. } \mathbf{H}_{k+1}^{-1} \mathbf{s}_k = \mathbf{y}_k,$$

و \mathbf{H}_{k+1}^{-1} یک ماتریس مثبت معین است. برهان. [۲۱]

حال می‌توان با استفاده از مطلب فوق بعد از محاسبه ماتریس به روزرسانی شده جهت جستجو را به صورت زیر به دست آورد:

$$\mathbf{d}_k = -\gamma_{k-1} \mathbf{g}_k$$

$$+\gamma_{k-1} \left(\frac{\mathbf{s}_{k-1}^T \mathbf{g}_k - \gamma_{k-1} \mathbf{y}_{k-1}^T \mathbf{g}_k}{\mathbf{y}_{k-1}^T \mathbf{s}_{k-1} - \gamma_{k-1} \mathbf{y}_{k-1}^T \mathbf{y}_{k-1}} \right) \mathbf{y}_{k-1}$$

$$-\left(\frac{\mathbf{s}_{k-1}^T \mathbf{g}_k - \gamma_{k-1} \mathbf{y}_{k-1}^T \mathbf{g}_k}{\mathbf{y}_{k-1}^T \mathbf{s}_{k-1} - \gamma_{k-1} \mathbf{y}_{k-1}^T \mathbf{y}_{k-1}} \right) \mathbf{s}_{k-1}.$$

که در آن γ_{k-1} در رابطه (۴) داده شده است. حال یک رویکرد بازگشتی را جهت محاسبه مؤثرتر $H.u$ ارایه می-دهیم.

۳-۲ الگوریتم پیشنهادی

الگوریتم ۳-۱ (الگوریتم بازگشتی دو حلقه‌ای L-SR1)

ورودی‌ها: جفت بردارهای مورد استفاده در حافظه محدود $\{\mathbf{s}_j, \mathbf{y}_j\}$ ، مقدار ویژه μ_i و λ_{k+1} ، k ، m ، q و ماتریس هسیان اولیه $\mathbf{H}_{i,k}(\lambda_k)$.

```

 $\bar{m} = \min\{k, m-1\};$ 
 $q \leftarrow v;$ 
for  $j = k-1, k-2, \dots, k-m$ 
    Compute  $Y_{i,j}(\lambda_{k+1})$  using :
    
$$Y_{i,j}(\lambda_{k+1}) = \frac{\lambda_{k+1}}{\mu_i} s_j + y_j.$$

     $\alpha_j \leftarrow u_j^T q;$ 
     $q \leftarrow q - \alpha_j Y_{i,j}(\lambda_{k+1});$ 
     $p_i \leftarrow q;$ 
end(for)
if  $k \leq m$ 
     $r \leftarrow H_i(\lambda)$ 
else
     $r \leftarrow H_{i,k}(\lambda_{k+1})q;$ 
for  $j = k-\hat{m}, \dots, k-1$ 
     $\beta_j \leftarrow Y_{i,j}(\lambda_{k+1})r;$ 
     $r \leftarrow r - \beta_j u_j;$ 
if  $j < k-1$ 
     $r \leftarrow r + \rho_j s_j^T p_{j+1} s_j;$ 
else
     $r \leftarrow r + \rho_j (s_j^T g_{j+1} s_j);$ 
end(for)
stop with result  $H_{i,k}(\lambda_k)q =: r.$ 

```

با توجه به مطالب فوق در صورتی که H را به عنوان تقریبی برای معکوس ماتریس هسیان، g را به عنوان گرادیان تابع هدف و y را به عنوان تفاضل گرادیان تابع هدف در دو تکرار فعلی و قبلی لحاظ کنیم، حاصل عبارت‌های g و $H.y$ می‌توانند به ترتیب با جایگزین نمودن v با g و u با y (و u از الگوریتم فوق محاسبه می‌شوند) به دست آیند؛ بنابراین u باید در هر تکرار طوری ذخیره شود که در تکرار بعدی بتواند مورد استفاده قرار گیرد. علاوه بر این، نیازمند ذخیره نمودن s ، y و ρ در هر تکرار هستیم.

$$S_k = [s_{k-m}, \dots, s_{k-2}, s_{k-1}], \quad Y_k = [y_{k-m}, \dots, y_{k-2}, y_{k-1}], \\ \rho_k = [\rho_{k-m}, \dots, \rho_{k-2}, \rho_{k-1}], \quad U_k = [u_{k-m}, \dots, u_{k-2}, u_{k-1}].$$

حال در این قسمت الگوریتمی را به صورت زیر ارایه می‌کنیم. با توجه به اینکه در روش پیشنهادی از ادغام روش‌های جستجوی خطی و ناحیه مطمئن استفاده شده است آن را الگوریتم پیوندی جستجوی مطمئن می‌نامیم.

الگوریتم ۲-۳. الگوریتم پیوندی جستجوی مطمئن (HTSA^۱)

گام اول. برای نقطه اولیه داده شده x_0 و $M \geq 2$ ، $\eta \in (0, 0.1)$ ، $\delta \in (0, 1)$ ، $\varepsilon > 0$ ، $k := 0$ قرار دهد: $H_k = \nabla f(x_k) = g_k \in \mathbb{R}^{n \times 1}$

گام دوم. شرط توقف. اگر $\varepsilon \leq \|\nabla f(x_k)\|$ آنگاه روند الگوریتم را متوقف کنید در غیر این صورت به گام سوم بروید.

گام سوم. محاسبه گام آزمایشی. زیر مساله ناحیه مطمئن را جهت مشخص نمودن یک گام آزمایشی d_k که در شرط‌های زیر صدق می‌کند، حل کنید.

گام چهارم. پذیرش گام آزمایشی. برای پارامتر ρ_k اگر $\rho_k > \eta$ قرار

$$\rho_k = \frac{\text{ared}_k}{\text{pred}_k} = \frac{f(x_k) - f(x_k + d_k)}{-(g_k^T d_k + \frac{1}{2} d_k^T H_k^{-1} d_k)}$$

دهید $x_{k+1} = x_k + d_k$ و برو به گام پنجم (استفاده از ایده روش زیر فضا)، در غیر این صورت برو به گام هشتم.

گام پنجم. پارامتر $m_k = \min\{k, M\}$ را تعریف و قرار می‌دهیم: $V_k = \{g_k, g_{k-1}, \dots, g_{k-m_k}\} \in \mathbb{R}^{n \times m_k+1}$ حال اگر ماتریس $h_k V_k^T H_k^{-1} V_k + I$ معین مثبت باشد، به گام بعدی می‌رویم در غیر این صورت l_k را کوچک-ترین عدد صحیح مثبت قرار دهد که معین مثبت باشد ولذا $h_k = 2^{-l_k}$.

گام ششم. دستگاه معادلات خطی $(h_k V_k^T H_k^{-1} V_k + I) y = -h_k V_k^T \nabla f(x_k)$ را جهت محاسبه y_k حل کنید.

گام هفتم. بردار $y_k = V_k y_k$ را محاسبه کرده اگر $f(x_k + d_k) \leq f(x_k) + \delta g_k^T d_k$ آنگاه $x_{k+1} = x_k + d_k$ و برو به گام نهم، در غیر این صورت برو به گام هشتم.

گام هشتم. قرار دهد $h_k = \tau h_{k+1}$ و $x_{k+1} = x_k + \alpha_k d_k$ و $h_{k+1} = \tau h_k$ را توسط مراحل زیر محاسبه کنید و h_k را توسط رابطه $h_{k+1} = \frac{1}{h_{k+1}} \frac{c}{\|\nabla f(x_k)\|}$ بروز رسانی کنید و سپس قرار دهد λ_{k+1} و بروید به گام آخر.

محاسبه جهت جستجو:

گام ۴-۱. با مقادیر اولیه $x_{k+1}^{(0)} = x_k + \alpha_k d_k$ و بردارهای λ_k و y_i و s_i و d_i را عدد صفر در نظر بگیرید.

گام ۴-۲. اگر $d_n \leq 0$ ، آنگاه متوقف شوید.

گام ۴-۳. $x_{k+1}^{(j+1)}$ را با استفاده از رابطه‌ی زیر و استفاده از الگوریتم بازگشتی دو حلقه‌ای L-SR1 محاسبه کنید.

$$x_{k+1}^{(1)} = x_{k+1}^{(0)} - (\lambda_k I + \nabla^T f(x_{k+1}^{(0)}))^{-1} (\lambda_k (x_{k+1}^{(0)} - x_k) + \nabla f(x_{k+1}^{(0)}))$$

گام ۴-۴. قرار دهد $d_n = d_k^T g_k$ و $d_k = x_{k+1}^{(j+1)} - x_k$.

محاسبه طول گام: با استفاده از شرایط زیر (شرایط لوف) طول گام را باید:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \zeta \alpha_k g_k^T d_k,$$

¹ Hybrid Trust-Search Algorithm

$$\nabla f(x_k + \alpha_k d_k)^T d_k \geq \zeta^T g_k^T d_k,$$

گام نهم. H_k را توسط فرمول SR1 به H_{k+1} به رویزرسانی کنید.

گام دهم. قرار دهید $k := k + 1$ و به گام دوم بروید.

استفاده از الگوریتم فوق بعد دستگاه خطی $(h_k V_k^T H_k^{-1} V_k + I) y = -h_k V_k^T \nabla f(x_k)$ را از n به $m_k + 1$ تقلیل داده و بنابراین حل دستگاه فوق در حالتی که $m < n$ به مراتب آسان‌تر خواهد بود. در این روش عملیات محاسباتی نیز برای مسایل با ابعاد بزرگ کاهش خواهد یافت. در الگوریتم فوق زمانی که تعداد متغیرها کم باشد، محاسبه و ذخیره ماتریس H_k کم هزینه می‌باشد؛ بنابراین بهتر است H_k توسط دستور SR1 محاسبه شود. با این حال ذخیره‌سازی و فرض‌های محاسباتی نسبت به n^2 افزایش می‌یابد؛ بنابراین n بیش از حد بزرگ می‌شود و در این حالت بهتر است که از نمایش فشرده آن؛ یعنی استفاده از الگوریتم بازگشتی دوحلقه‌ای L-SR1 استفاده نمود.

با توجه به توضیحات اشاره شده در قسمت‌های قبل و دقت در الگوریتم ارایه شده ملاحظه می‌شود که ماتریس V_k که با توجه به اطلاعات مراحل قبل به دست آمده است، علاوه بر اینکه نقش اساسی در ساده‌سازی دستگاه معادلات خطی دارد، می‌تواند باعث همگرایی پایدار در محاسبات عملی شود. همچنین انتخاب‌های دیگری نیز از ماتریس V_k می‌توان برای به دست آوردن جواب‌های بهتر داشت.

۴ نتایج عددی

در این قسمت، نتایج عددی به دست آمده از رفتار الگوریتم پیشنهادی را ارایه می‌دهیم. همچنین، مقایسه‌ای با الگوریتم‌های FRSCG و L-BFGS جهت نشان دادن توانایی الگوریتم پیشنهادی خواهیم داشت. الگوریتم‌ها بر روی ۵۲ مساله آزمون استاندارد از مجموعه مسایل آزمون استاندارد ارایه شده در [۲۲]، پیاده‌سازی شده است. همچنین نقاط اولیه برای مجموعه مسایل استاندارد همانند نقاط آغازی در مراجع فوق در نظر گرفته شده که در ضمیمه نیز ارایه شده است. در کل ۱۵۶ اجرا برای هر الگوریتم انجام شده است. برای هر تابع، ۳ بعد مختلف، ۱۰۰۰، ۵۰۰۰ و ۱۰۰۰۰ در نظر گرفته شده است.

زبان برنامه نویسی مورد استفاده فورترن^۱ می‌باشد، از کامپیوتر با مشخصات Pentium® Intel Core با رم ۴ Gb جهت اجرا و پیاده‌سازی استفاده شده است. تلرانس توقف مورد استفاده در تمامی الگوریتم‌ها^۲ ۱۰^{-۵} می‌باشد. در تمامی حالت‌ها دقت مضاعف محاسباتی^۳ لحاظ شده است. همچنین در اجرای تمامی الگوریتم‌ها شکست را به این معنی در نظر خواهیم گرفت که شرط توقف پس از حداقل ۱۰۰۰ تکرار برقرار نباشد؛ بنابراین پس از تکرار ۱۰۰۰ ام اجرای الگوریتم را متوقف و به عنوان "شکست"^۳ لحاظ خواهد شد. پارامترهای مورد استفاده در الگوریتم $h_i = 10^{-4}$ ، $\delta_i = 10^{-4}$ ، $\tau_i = 0.5$ ، $\gamma_i = 0.9$ ، $\delta_i = 10^{-4}$ ، $\tau_i = 0.2$ می‌باشند.

¹Fortran g77

²Double Precision Arithmetic

³Failure

از الگوریتم‌های IMPBOT، LBFGS و FRSCG جهت مقایسه با الگوریتم پیشنهادی استفاده شده است. این الگوریتم‌ها به ترتیب در منابع [۳] و [۹] ارایه شده‌اند.

نتایج دقیق الگوریتم‌ها در جدول ۱ ارایه شده است، در جدول ۱، n_i و $n_{f/g}$ به ترتیب تعداد تکرار و تعداد فراخوانی‌های تابع/گرادیان می‌باشد. نتایج عددی به دست آمده نشان می‌دهند که الگوریتم پیشنهادی در مقایسه با الگوریتم‌های IMPBOT و FRSCG نتایج قابل قبولی را علی الخصوص برای مسایل مقیاس بزرگ به دست می‌دهد و در بسیاری از موارد با الگوریتم L-BFGS در رقابت می‌باشد.

جدول ۱. نتایج آزمون برای روش‌های FRSCG، L-BFGS، IMPBOT، HTSA

| نام مساله | بعد | HTSA | | IMPBOT | | L-BFGS | | FRSCG | |
|---------------------|-------|-------|-----------|--------|-----------|--------|-----------|-------|-----------|
| | | n_i | $n_{f/g}$ | n_i | $n_{f/g}$ | n_i | $n_{f/g}$ | n_i | $n_{f/g}$ |
| Freudenstein & Roth | ۱۰۰۰ | ۱۸ | ۲۴ | ۲۶ | ۲۷ | ۲۰ | ۲۵ | ۴۶ | ۱۷۴ |
| | ۵۰۰۰ | ۱۹ | ۳۴ | ۳۱ | ۲۹ | ۲۲ | ۳۴ | ۸۴ | ۱۹۸ |
| | ۱۰۰۰۰ | ۲۲ | ۲۶ | ۳۱ | ۳۲ | ۲۴ | ۳۰ | ۷۰ | ۳۰۳ |
| Trigonometric | ۱۰۰۰ | ۳۷ | ۳۹ | ۴۸ | ۵۱ | ۴۳ | ۴۵ | ۱۷۱ | ۲۳۰ |
| | ۵۰۰۰ | ۳۵ | ۳۶ | ۴۵ | ۵۶ | ۴۱ | ۴۷ | ۱۸۰ | ۱۰۷ |
| | ۱۰۰۰۰ | ۳۴ | ۲۸ | ۴۶ | ۶۳ | ۳۸ | ۳۹ | ۲۷۵ | ۳۲۴ |
| Rosenbrock | ۱۰۰۰ | ۲۷ | ۵۳ | ۴۴ | ۶۶ | ۳۱ | ۵۸ | ۷۴ | ۱۱۸ |
| | ۵۰۰۰ | ۲۸ | ۴۸ | ۵۳ | ۸۷ | ۳۵ | ۵۵ | ۹۷ | ۱۶۵ |
| | ۱۰۰۰۰ | ۳۲ | ۴۵ | ۵۹ | ۸۹ | ۳۸ | ۵۷ | ۱۰۷ | ۱۸۹ |
| White & Holst | ۱۰۰۰ | ۳۶ | ۵۸ | ۵۳ | ۷۴ | ۴۴ | ۶۴ | ۷۸ | ۱۴۶ |
| | ۵۰۰۰ | ۴۰ | ۶۷ | ۴۶ | ۸۸ | ۴۷ | ۷۷ | ۵۳ | ۹۷ |
| | ۱۰۰۰۰ | ۳۱ | ۶۱ | ۴۴ | ۷۲ | ۳۸ | ۶۹ | ۴۸ | ۸۷ |
| Beale | ۱۰۰۰ | ۲۰ | ۲۹ | ۳۱ | ۲۷ | ۲۳ | ۳۶ | ۱۹ | ۳۵ |
| | ۵۰۰۰ | ۱۳ | ۲۰ | ۳۳ | ۳۳ | ۱۸ | ۲۵ | ۱۹ | ۳۶ |
| | ۱۰۰۰۰ | ۲۲ | ۳۲ | ۳۸ | ۴۲ | ۲۸ | ۴۰ | ۲۹ | ۵۴ |
| Penalty | ۱۰۰۰ | ۳۷ | ۴۵ | ۴۳ | ۵۷ | ۴۵ | ۵۵ | ۲۴ | ۴۰ |
| | ۵۰۰۰ | ۳۸ | ۴۳ | ۴۸ | ۵۷ | ۴۴ | ۵۷ | ۲۵ | ۴۱ |
| | ۱۰۰۰۰ | ۴۳ | ۵۱ | ۵۵ | ۶۴ | ۵۲ | ۶۶ | ۲۴ | ۴۵ |
| Raydan ۱ | ۱۰۰۰ | ۱۸۷ | ۲۵۱ | ۲۵۲ | ۳۰۱ | ۲۲۸ | ۲۴۵ | ۳۱۱ | ۴۰۰ |
| | ۵۰۰۰ | - | - | - | - | - | - | - | - |
| | ۱۰۰۰۰ | - | - | - | - | - | - | - | - |
| Raydan ۲ | ۱۰۰۰ | ۵ | ۱۰ | ۸ | ۹ | ۷ | ۹ | ۴ | ۹ |
| | ۵۰۰۰ | ۵ | ۱۰ | ۸ | ۹ | ۷ | ۹ | ۴ | ۹ |
| | ۱۰۰۰۰ | ۸ | ۱۳ | ۹ | ۱۱ | ۷ | ۱۰ | ۴ | ۹ |
| Diagonal ۱ | ۱۰۰۰ | ۱۹۵ | ۲۴۶ | ۳۰۴ | ۴۲۷ | ۲۵۴ | ۲۶۸ | ۴۹۳ | ۸۲۹ |
| | ۵۰۰۰ | - | - | - | - | - | - | - | - |
| | ۱۰۰۰۰ | - | - | - | - | - | - | - | - |
| Diagonal ۲ | ۱۰۰۰ | ۱۷۱ | ۱۸۲ | ۱۹۸ | ۲۲۳ | ۱۶۳ | ۱۹۳ | ۲۵۳ | ۳۲۵ |
| | ۵۰۰۰ | ۲۶۷ | ۳۸۹ | ۴۳۷ | ۴۳۱ | ۳۹۸ | ۴۴۱ | ۵۳۷ | ۶۶۸ |
| | ۱۰۰۰۰ | - | - | - | - | - | - | - | - |

۱۵۰ جدول ۱. نتایج آزمون برای روش‌های FRSCG, L-BFGS, IMPBOT, HTSA

| نام مساله | بعد | HTSA | | IMPBOT | | L-BFGS | | FRSCG | |
|---------------------|-------|-------|-----------|--------|-----------|--------|-----------|-------|-----------|
| | | n_i | $n_{f/g}$ | n_i | $n_{f/g}$ | n_i | $n_{f/g}$ | n_i | $n_{f/g}$ |
| Perturbed Quadratic | ۱۰۰۰ | ۲۸۱ | ۳۰۴ | ۳۰۳ | ۳۲۹ | ۲۹۵ | ۳۲۰ | ۳۸۴ | ۴۳۹ |
| | ۵۰۰۰ | - | - | - | - | - | - | - | - |
| | ۱۰۰۰۰ | - | - | - | - | - | - | - | - |
| G. Tridiagonal ۱ | ۱۰۰۰ | ۲۷ | ۴۲ | ۳۹ | ۴۷ | ۳۴ | ۴۴ | ۶۱ | ۹۲ |
| | ۵۰۰۰ | ۲۶ | ۵۱ | ۳۷ | ۵۶ | ۳۲ | ۵۴ | ۲۳ | ۵۲ |
| | ۱۰۰۰۰ | ۲۹ | ۵۶ | ۴۱ | ۵۹ | ۳۶ | ۵۶ | ۹۸ | ۶۱۹ |
| Tridiagonal ۱ | ۱۰۰۰ | ۲۱ | ۳۱ | ۳۰ | ۳۵ | ۲۷ | ۳۵ | ۲۸ | ۵۵ |
| | ۵۰۰۰ | ۲۸ | ۴۲ | ۳۴ | ۴۶ | ۳۳ | ۴۴ | ۱۰ | ۲۲ |
| | ۱۰۰۰۰ | ۲۶ | ۳۵ | ۲۷ | ۳۹ | ۲۷ | ۳۷ | ۳۸ | ۷۴ |
| Three Expo Terms | ۱۰۰۰ | ۱۴ | ۱۶ | ۱۵ | ۲۰ | ۱۴ | ۱۸ | ۱۷ | ۳۰ |
| | ۵۰۰۰ | ۱۷ | ۲۰ | ۲۱ | ۲۲ | ۱۹ | ۲۲ | ۲۳ | ۴۰ |
| | ۱۰۰۰۰ | ۲۰ | ۲۲ | ۱۴ | ۲۴ | ۱۲ | ۲۴ | ۲۸ | ۱۰۸ |
| G. Tridiagonal ۲ | ۱۰۰۰ | ۷۱ | ۹۱ | ۸۶ | ۱۰۵ | ۸۲ | ۱۰۰ | ۱۹۳ | ۲۲۸ |
| | ۵۰۰۰ | ۷۸ | ۹۴ | ۹۱ | ۱۰۸ | ۸۶ | ۱۰۳ | ۱۱۷ | ۱۵۲ |
| | ۱۰۰۰۰ | ۹۱ | ۱۱۴ | ۱۰۶ | ۱۲۷ | ۱۰۲ | ۱۲۲ | ۳۶۴ | ۴۸۰ |
| NONDIA | ۱۰۰۰ | ۲۰ | ۳۱ | ۲۷ | ۳۵ | ۲۴ | ۳۳ | ۱۲ | ۲۵ |
| | ۵۰۰۰ | ۱۴ | ۲۳ | ۱۹ | ۲۴ | ۱۷ | ۲۴ | ۷ | ۱۴ |
| | ۱۰۰۰۰ | ۱۷ | ۲۳ | ۱۹ | ۳۵ | ۱۷ | ۲۲ | ۷ | ۱۴ |
| DQDRIC | ۱۰۰۰ | ۳۴ | ۴۵ | ۴۲ | ۵۰ | ۴۱ | ۴۹ | ۶ | ۱۳ |
| | ۵۰۰۰ | ۲۲ | ۳۱ | ۳۶ | ۴۷ | ۲۷ | ۳۵ | ۷ | ۱۵ |
| | ۱۰۰۰۰ | ۳۳ | ۴۵ | ۴۳ | ۵۳ | ۳۸ | ۴۹ | ۷ | ۱۵ |
| DENSCHNB | ۱۰۰۰ | ۷ | ۱۴ | ۱۷ | ۱۹ | ۱۳ | ۱۵ | ۸ | ۱۷ |
| | ۵۰۰۰ | ۹ | ۱۳ | ۱۴ | ۱۸ | ۱۲ | ۱۴ | ۸ | ۱۷ |
| | ۱۰۰۰۰ | ۹ | ۱۸ | ۱۸ | ۲۰ | ۱۴ | ۱۸ | ۹ | ۱۹ |
| DENSCHNF | ۱۰۰۰ | ۱۵ | ۲۱ | ۲۲ | ۳۷ | ۲۱ | ۳۶ | ۱۴۵ | ۴۳۸ |
| | ۵۰۰۰ | ۱۶ | ۲۳ | ۲۳ | ۳۹ | ۲۰ | ۳۴ | ۱۳۲ | ۴۹۱ |
| | ۱۰۰۰۰ | ۲۱ | ۲۸ | ۲۹ | ۴۴ | ۲۶ | ۴۲ | ۹۹ | ۵۰۶ |
| SINQUAD | ۱۰۰۰ | ۱۳۷ | ۲۵۹ | ۱۶۸ | ۲۸۴ | ۱۵۷ | ۲۷۵ | ۲۵۹ | ۴۵۷ |
| | ۵۰۰۰ | - | - | - | - | - | - | - | - |
| | ۱۰۰۰۰ | - | - | - | - | - | - | - | - |

۱۵۰ جدول ۱. نتایج آزمون برای روش‌های FRSCG, L-BFGS, IMPBOT, HTSA

| نام مساله | بعد | HTSA | | IMPBOT | | L-BFGS | | FRSCG | |
|----------------------------|-------|-------|-----------|--------|-----------|--------|-----------|-------|-----------|
| | | n_i | $n_{f/g}$ | n_i | $n_{f/g}$ | n_i | $n_{f/g}$ | n_i | $n_{f/g}$ |
| PPQ ^۱ | ۱۰۰۰ | ۲۳۰ | ۲۸۹ | ۲۵۵ | ۳۰۳ | ۲۴۶ | ۲۹۸ | ۶۰۷ | ۶۸۸ |
| | ۵۰۰۰ | ۹۴ | ۱۳۴ | ۱۱۹ | ۱۵۲ | ۱۱۰ | ۱۴۵ | ۲۵۷ | ۴۴۰ |
| | ۱۰۰۰۰ | ۵۷ | ۹۵ | ۸۵ | ۱۰۹ | ۷۵ | ۱۰۳ | ۳۹ | ۶۹ |
| Broyden Tridiagonal | ۱۰۰۰ | ۳۳ | ۵۹ | ۴۲ | ۶۴ | ۳۸ | ۶۲ | - | - |
| | ۵۰۰۰ | ۵۴ | ۸۲ | ۶۷ | ۸۹ | ۶۱ | ۸۵ | ۲۷۵ | ۳۱۴ |
| | ۱۰۰۰۰ | ۵۲ | ۶۹ | ۵۸ | ۷۴ | ۵۶ | ۷۲ | ۱۸۳ | ۲۲۰ |
| Almost Perturbed Quadratic | ۱۰۰۰ | ۲۶۵ | ۳۰۳ | ۲۸۲ | ۳۰۶ | ۲۷۸ | ۳۰۵ | ۷۷۲ | ۸۲۶ |
| | ۵۰۰۰ | - | - | - | - | - | - | - | - |
| | ۱۰۰۰۰ | - | - | - | - | - | - | - | - |
| EDENSCH | ۱۰۰۰ | ۲۳ | ۴۷ | ۲۶ | ۵۴ | ۲۶ | ۵۱ | ۳۰ | ۵۶ |
| | ۵۰۰۰ | ۲۷ | ۶۱ | ۳۱ | ۶۳ | ۳۰ | ۶۲ | ۴۳ | ۱۸۳ |
| | ۱۰۰۰۰ | ۳۲ | ۷۷ | ۳۵ | ۸۱ | ۳۵ | ۷۹ | ۴۳ | ۷۴ |
| LIARWHD | ۱۰۰۰ | ۲۹ | ۴۸ | ۳۷ | ۵۲ | ۳۴ | ۵۰ | ۱۱۶ | ۲۳۰ |
| | ۵۰۰۰ | ۳۳ | ۵۶ | ۴۲ | ۶۱ | ۳۸ | ۵۹ | ۷۳ | ۱۴۱ |
| | ۱۰۰۰۰ | ۳۸ | ۵۹ | ۴۵ | ۶۳ | ۴۲ | ۶۱ | ۹۱ | ۱۷۳ |
| Diagonal ۹ | ۱۰۰۰ | ۶ | ۹ | ۸ | ۱۰ | ۷ | ۹ | ۴ | ۹ |
| | ۵۰۰۰ | ۶ | ۹ | ۸ | ۱۰ | ۷ | ۹ | ۴ | ۹ |
| | ۱۰۰۰۰ | ۶ | ۹ | ۸ | ۱۰ | ۷ | ۱۰ | ۴ | ۹ |
| DIXMAANF | ۱۰۰۰ | ۱۱۴ | ۱۲۷ | ۱۳۹ | ۱۴۴ | ۱۳۲ | ۱۴۰ | ۲۲۹ | ۲۶۵ |
| | ۵۰۰۰ | ۲۳۵ | ۲۶۲ | ۲۶۱ | ۲۷۲ | ۲۵۳ | ۲۶۷ | - | - |
| | ۱۰۰۰۰ | - | - | - | - | - | - | - | - |
| Diagonal ۴ | ۱۰۰۰ | ۶ | ۱۳ | ۷ | ۱۳ | ۶ | ۱۲ | ۶ | ۱۲ |
| | ۵۰۰۰ | ۶ | ۱۳ | ۷ | ۱۳ | ۶ | ۱۲ | ۷ | ۱۴ |
| | ۱۰۰۰۰ | ۶ | ۱۳ | ۷ | ۱۳ | ۶ | ۱۲ | ۲۲ | ۴۱ |
| Diagonal ۵ | ۱۰۰۰ | ۵ | ۷ | ۶ | ۹ | ۵ | ۸ | ۴ | ۹ |
| | ۵۰۰۰ | ۵ | ۷ | ۶ | ۹ | ۵ | ۸ | ۴ | ۹ |
| | ۱۰۰۰۰ | ۵ | ۷ | ۵ | ۸ | ۴ | ۸ | ۴ | ۹ |
| Himmelblau | ۱۰۰۰ | ۱۳ | ۲۶ | ۱۵ | ۳۰ | ۱۴ | ۲۸ | ۱۶ | ۳۰ |
| | ۵۰۰۰ | ۱۴ | ۲۸ | ۱۶ | ۲۷ | ۱۵ | ۲۷ | ۱۹ | ۳۶ |
| | ۱۰۰۰۰ | ۱۶ | ۳۲ | ۲۱ | ۳۱ | ۱۹ | ۲۲ | ۲۱ | ۳۷ |
| G.PSC ^۱ | ۱۰۰۰ | ۷۹ | ۱۰۲ | ۹۹ | ۱۱۹ | ۹۳ | ۱۱۳ | ۲۳۱ | ۳۷۳ |
| | ۵۰۰۰ | ۱۱۲ | ۱۳۷ | ۱۳۰ | ۱۵۲ | ۱۲۵ | ۱۴۹ | - | - |
| | ۱۰۰۰۰ | - | - | - | - | - | - | - | - |

۱۵۰ جدول ۱. نتایج آزمون برای روش‌های FRSCG, L-BFGS, IMPBOT, HTSA

| نام مساله | بعد | HTSA | | IMPBOT | | L-BFGS | | FRSCG | |
|-----------|-------|-------|-----------|--------|-----------|--------|-----------|-------|-----------|
| | | n_i | $n_{f/g}$ | n_i | $n_{f/g}$ | n_i | $n_{f/g}$ | n_i | $n_{f/g}$ |
| PSC1 | ۱۰۰۰ | ۱۰ | ۱۸ | ۲۳ | ۳۰ | ۱۸ | ۲۰ | ۱۱ | ۲۳ |
| | ۵۰۰۰ | ۱۱ | ۱۸ | ۲۵ | ۳۶ | ۱۹ | ۲۱ | ۱۱ | ۲۳ |
| | ۱۰۰۰۰ | ۱۹ | ۲۶ | ۲۹ | ۴۰ | ۱۹ | ۲۴ | ۳۵ | ۸۰۷ |
| Maratos | ۱۰۰۰ | ۶۸ | ۱۱۰ | ۷۷ | ۱۱۹ | ۷۴ | ۱۱۶ | ۶۲۸ | ۸۴۹ |
| | ۵۰۰۰ | ۵۹ | ۹۸ | ۶۴ | ۱۰۲ | ۶۳ | ۱۰۰ | - | - |
| | ۱۰۰۰۰ | ۴۷ | ۷۷ | ۵۴ | ۸۳ | ۵۲ | ۸۲ | ۶۲۹ | ۹۴۸ |
| Cliff | ۱۰۰۰ | ۵۱ | ۵۸ | ۵۹ | ۶۵ | ۵۶ | ۶۲ | ۵۷ | ۹۳ |
| | ۵۰۰۰ | ۵۶ | ۶۵ | ۶۱ | ۶۹ | ۵۹ | ۶۷ | ۲۵ | ۵۲ |
| | ۱۰۰۰۰ | ۶۱ | ۷۵ | ۶۰ | ۶۸ | ۶۰ | ۷۰ | ۴۴ | ۸۸ |
| Wood | ۱۰۰۰ | ۲۲۶ | ۳۰۷ | ۲۱۲ | ۳۰۳ | ۲۳۲ | ۳۱۲ | ۷۰ | ۱۳۲ |
| | ۵۰۰۰ | ۲۳۷ | ۲۷۱ | ۲۲۳ | ۲۷۸ | ۲۲۱ | ۲۷۸ | ۹۴ | ۱۷۸ |
| | ۱۰۰۰۰ | - | - | - | - | - | - | ۸۱ | ۱۵۵ |
| Hiebert | ۱۰۰۰ | ۷۴ | ۱۳۳ | ۸۵ | ۱۴۲ | ۸۲ | ۱۴۰ | ۱۰۹ | ۲۰۵ |
| | ۵۰۰۰ | ۵۸ | ۱۰۲ | ۶۸ | ۱۱۱ | ۶۵ | ۱۰۹ | ۸۴ | ۱۵۱ |
| | ۱۰۰۰۰ | ۹۱ | ۱۶۰ | ۱۰۲ | ۱۶۹ | ۹۸ | ۱۶۷ | ۶۹ | ۱۳۱ |
| QF1 | ۱۰۰۰ | ۲۶۴ | ۲۷۸ | ۲۶۶ | ۲۹۴ | ۲۶۷ | ۲۸۸ | ۷۲۸ | ۷۸۳ |
| | ۵۰۰۰ | - | - | - | - | - | - | - | - |
| | ۱۰۰۰۰ | - | - | - | - | - | - | - | - |
| QP1 | ۱۰۰۰ | ۱۸ | ۲۲ | ۱۹ | ۲۹ | ۱۸ | ۲۰ | ۱۲ | ۲۸ |
| | ۵۰۰۰ | ۲۶ | ۲۸ | ۲۶ | ۳۳ | ۲۴ | ۲۶ | - | - |
| | ۱۰۰۰۰ | ۲۶ | ۲۹ | ۲۶ | ۳۳ | ۲۴ | ۲۶ | - | - |
| QP2 | ۱۰۰۰ | ۳۸ | ۷۱ | ۴۵ | ۸۲ | ۴۲ | ۷۷ | - | - |
| | ۵۰۰۰ | ۵۴ | ۸۵ | ۶۱ | ۹۴ | ۵۸ | ۸۹ | ۲۴۸ | ۳۷۲ |
| | ۱۰۰۰۰ | ۳۲ | ۶۱ | ۳۹ | ۶۹ | ۳۶ | ۶۶ | ۲۸۵ | ۴۲۵ |
| QF2 | ۱۰۰۰ | ۳۲۱ | ۳۸۷ | ۳۵۱ | ۳۷۷ | ۳۴۱ | ۳۷۴ | - | - |
| | ۵۰۰۰ | - | - | - | - | - | - | - | - |
| | ۱۰۰۰۰ | - | - | - | - | - | - | - | - |
| EP1 | ۱۰۰۰ | ۵ | ۸ | ۴ | ۷ | ۴ | ۷ | ۲ | ۵ |
| | ۵۰۰۰ | ۵ | ۸ | ۴ | ۷ | ۴ | ۷ | ۳ | ۶ |
| | ۱۰۰۰۰ | ۵ | ۸ | ۴ | ۷ | ۴ | ۷ | ۵ | ۹ |
| ARWHEAD | ۱۰۰۰ | ۱۳ | ۲۴ | ۲۲ | ۲۸ | ۱۸ | ۲۷ | ۲۴ | ۴۷ |
| | ۵۰۰۰ | ۱۴ | ۲۸ | ۱۰ | ۳۳ | ۱۱ | ۳۱ | ۳۵ | ۴۳۰ |
| | ۱۰۰۰۰ | ۱۶ | ۳۸ | ۱۲ | ۴۰ | ۱۳ | ۳۹ | - | - |

۱۵۰ جدول ۱. نتایج آزمون برای روش‌های FRSCG, L-BFGS, IMPBOT, HTSA

| نام مساله | بعد | HTSA | | IMPBOT | | L-BFGS | | FRSCG | |
|-----------|-------|-------|-----------|--------|-----------|--------|-----------|-------|-----------|
| | | n_i | $n_{f/g}$ | n_i | $n_{f/g}$ | n_i | $n_{f/g}$ | n_i | $n_{f/g}$ |
| DIXMAANA | 1000 | 8 | 10 | 11 | 13 | 10 | 12 | 11 | 22 |
| | 5000 | 8 | 10 | 11 | 13 | 10 | 12 | 11 | 22 |
| | 10000 | 8 | 10 | 11 | 13 | 10 | 12 | 12 | 24 |
| DIXMAANB | 1000 | 12 | 16 | 17 | 19 | 15 | 18 | 12 | 21 |
| | 5000 | 13 | 18 | 18 | 20 | 16 | 19 | 12 | 21 |
| | 10000 | 13 | 18 | 18 | 20 | 16 | 19 | 12 | 21 |
| DIXMAANC | 1000 | 15 | 20 | 14 | 18 | 14 | 18 | 15 | 27 |
| | 5000 | 16 | 23 | 14 | 16 | 14 | 18 | 16 | 28 |
| | 10000 | 16 | 23 | 16 | 18 | 16 | 18 | 15 | 26 |
| DIXMAANE | 1000 | 151 | 173 | 165 | 173 | 160 | 174 | 352 | 381 |
| | 5000 | 389 | 411 | 402 | 400 | 406 | 434 | 712 | 743 |
| | 10000 | - | - | - | - | - | - | 839 | 871 |
| DIXMAANG | 1000 | 133 | 140 | 146 | 153 | 141 | 148 | 273 | 303 |
| | 5000 | 223 | 245 | 255 | 263 | 224 | 260 | 571 | 702 |
| | 10000 | - | - | - | - | - | - | 792 | 825 |
| DIXMAANH | 1000 | - | - | - | - | - | - | 599 | 663 |
| | 5000 | - | - | - | - | - | - | 763 | 849 |
| | 10000 | - | - | - | - | - | - | - | - |
| DIXMAANI | 1000 | 123 | 138 | 147 | 161 | 139 | 153 | 312 | 343 |
| | 5000 | 265 | 278 | 297 | 318 | 286 | 304 | 574 | 705 |
| | 10000 | - | - | - | - | - | - | 889 | 921 |
| ENGVAL1 | 1000 | - | - | - | - | - | - | - | - |
| | 5000 | 23 | 44 | 26 | 51 | 25 | 48 | 28 | 49 |
| | 10000 | 21 | 46 | 26 | 48 | 24 | 47 | 22 | 46 |
| SQ1 | 1000 | 239 | 255 | 238 | 259 | 241 | 257 | 886 | 951 |
| | 5000 | - | - | - | - | - | - | - | - |
| | 10000 | - | - | - | - | - | - | - | - |
| SQ2 | 1000 | 120 | 209 | 133 | 213 | 128 | 211 | - | - |
| | 5000 | 315 | 388 | 335 | 417 | 327 | 407 | - | - |
| | 10000 | - | - | - | - | - | - | - | - |

جدول ۲. نسبت نتایج الگوریتم IMPBOT به HTSA

| الگوریتم | HTSA | |
|---------------|---------------|-------------------------|
| | تعداد تکرارها | تعداد فرآخوانی‌های تابع |
| میانگین حسابی | ۰/۹۲ | ۰/۹۰ |
| میانگین هندسی | ۰/۸۴ | ۰/۸۴ |

جدول ۳. نسبت نتایج الگوریتم HTSA به L-BFGS

| الگوریتم | HTSA | |
|---------------|---------------|-------------------------|
| | تعداد تکرارها | تعداد فراخوانی‌های تابع |
| میانگین حسابی | ۰/۹۵ | ۰/۹۲ |
| میانگین هندسی | ۰/۹۱ | ۰/۸۵ |

جدول ۴. نسبت نتایج الگوریتم HTSA به FRSCG

| الگوریتم | HTSA | |
|---------------|---------------|-------------------------|
| | تعداد تکرارها | تعداد فراخوانی‌های تابع |
| میانگین حسابی | ۰/۷۷ | ۰/۷۴ |
| میانگین هندسی | ۰/۷۱ | ۰/۶۹ |

خلاصه نتایج در جدول‌های ۲، ۳ و ۴ ارایه شده است. نتایج ارایه شده در این جداول بیان می‌کند که الگوریتم IMPBOT به طور موثری عملکرد الگوریتم HTSA را بهبود بخشیده است. بهبود HTSA بر روی IMPBOT به طور متوسط ۸ تا ۱۱٪ در تعداد تکرار می‌باشد و ۱۰ الی ۱۶٪ به طور میانگین در تعداد فراخوانی تابع/گرادیان می‌باشد. با توجه به جدول ۳-۴ بهبود الگوریتم HTSA نسبت به الگوریتم L-BFGS به طور متوسط ۵ الی ۹٪ در نظر گرفتن تعداد تکرارهای مورد نیاز جهت رسیدن به جواب و ۸ الی ۱۲٪ در تعداد محاسبات تابع/گرادیان می‌باشد. این مقادیر نسبت به الگوریتم FRSCG به طور متوسط ۲۳ الی ۲۹٪ در تعداد تکرار و ۲۶ الی ۳۱٪ در تعداد محاسبات تابع/گرادیان در جدول ۴ تغییر یافته است.

علاوه بر این از معیار دیگری نیز جهت بررسی عملکرد الگوریتم پیشنهادی در مقایسه با سایر الگوریتم‌ها استفاده شده است، این معیار که به نمودار عملکرد^۱ نیز معروف است توسط دولان و مور [۲۳] ارایه شده است و عملکرد هر الگوریتم را در تمامی اجراهای بر روی مجموعه مسایل آزمون به صورت گرافیکی نشان می‌دهد و برتری بیشتری به نسبت سایر ابزارهای تعیین معیار موجود برای آزمون مجموعه‌های بزرگ دارد.

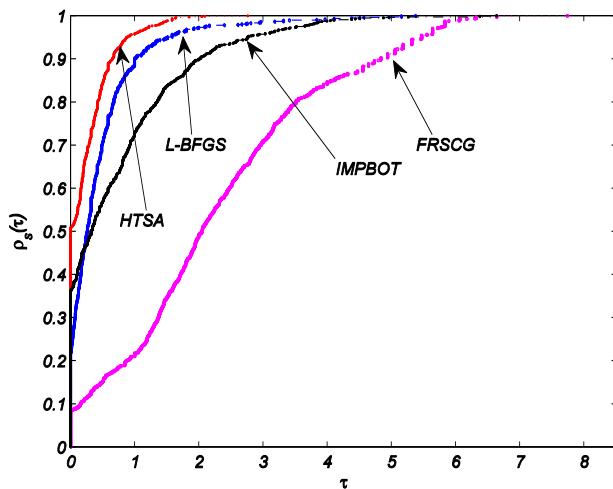
نسبت عملکرد به صورت $\frac{l_{p,s}}{l_p^*}$ تعریف می‌شود، که $l_{p,s}$ تعداد محاسبات مورد نیاز تابع هدف برای حل مساله p توسط الگوریتم s می‌باشد، و l_p^* نشان دهنده کمترین دفعات مورد نیاز جهت محاسبه تابع هدف در هر الگوریتم می‌باشد؛ لذا برای هر p و s داریم: $r_{p,s} \geq 1$. هرگاه الگوریتم قادر به حل مساله‌ای باشد برای نسبت $r_{p,s}$ عدد بزرگی مانند M تخصیص می‌یابد به طوری که برای هر p و s ، $r_{p,s} < M$ ؛ بنابراین نمودار

$$\text{عملکرد را به صورت } \frac{r_{p,s} \leq \tau}{\text{Number of all problems}}$$

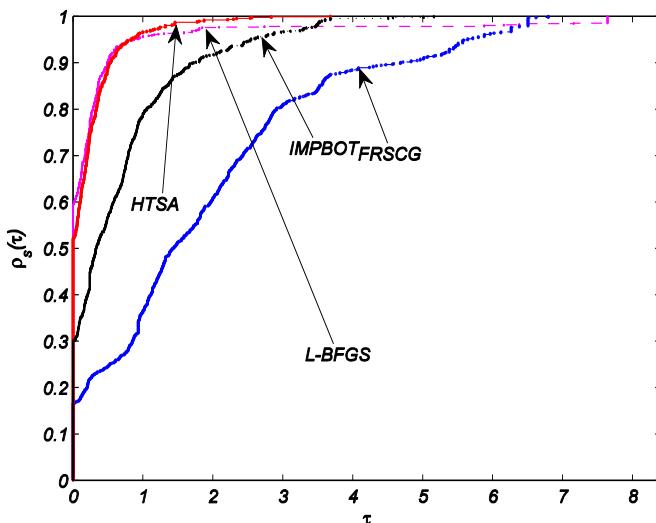
عملکرد را به صورت $\rho_s(\tau)$ می‌توان تعریف کرد. برای جزئیات بیشتر در خصوص نمودار عملکرد به [۲۳] مراجعه کنید.

¹Performance Profile

در شکل‌های ۱ و ۲ نمودار عملکرد الگوریتم‌های FRSCG، L-BFGS، IMPBOT، HTSA، به ترتیب بر مبنای تعداد تکرار و تعداد فراخوانی‌های تابع/گرادیان نمایش داده شده است.



شکل ۱. نمودار عملکرد الگوریتم‌های FRSCG، L-BFGS، IMPBOT، HTSA، بر مبنای تعداد تکرار



شکل ۲. نمودار عملکرد الگوریتم‌های FRSCG، L-BFGS، IMPBOT، HTSA، بر مبنای تعداد فراخوانی‌های تابع/گرادیان

همان‌طور که در شکل‌های ۱ و ۲ نشان داده شده است، الگوریتم HTSA در بین سایر الگوریتم‌ها و برای مجموعه مسائل آزمون مورد استفاده بهترین می‌باشد؛ بنابراین، از نتایج عددی موثر بودن روش پیشنهادی نسبت به روش‌های FRSCG، L-BFGS و IMPBOT به طور واضح مشهود است.

۵ نتیجه‌گیری

در مقاله حاضر روشی جدید با استفاده از ادغام استراتژی‌های ناحیه مطمئن و جستجوی خطی مبتنی بر ODE ارایه شد، روش پیشنهادی نه تنها از مزایای هر دو استراتژی ناحیه مطمئن و جستجوی خطی بهره می‌برد؛ بلکه با

توجه به نتایج موثر گزارش شده در خصوص فرمول بهروزرسانی SR1 از نمایش فشرده این فرمول در روش فوق استفاده شد تا برای مسایل مقیاس بزرگ به کار رود. نتایج به دست آمده موثر بودن روش پیشنهادی را نسبت به سایر الگوریتم‌ها در مجموعه مسایل آزمون مورد استفاده نشان داد. تحلیل همگرایی این روش در پژوهش‌های آتی مورد بحث قرار خواهد گرفت.

منابع

- [1] Shi, Z.J., Xu, Z.W., (2009). The convergence of subspace trust region methods. *J. Comput. Appl. Math.* 231, 365–377.
- [2] More, J., Garbow, B., Hillstrom, K., (1981). Testing unconstrained optimization software. *ACM Trans. Math. Softw.* 7, 17–41.
- [3] Luo, X.L., Kelley, C.T., Liao, L.Z., Tam, H.W., (2009). Combining trust-region techniques and Rosenbrock methods to compute stationary points. *J. Optim. Theory Appl.* 140, 265–286.
- [4] Liu, D.C., Nocedal, J., (1989). On the limited memory BFGS method for large scale optimization. *Math. Program.* 45, 503–528.
- [5] Dolan, E.D., More, J., (2002). Benchmarking optimization software with performance profiles. *Math. Program., Ser. A* 91, 201–213.
- [6] Ortega, J. M., Rheinboldt, W.C., (1970). *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York.
- [7] Andrei, N., Gradient flow algorithm for unconstrained optimization. ICI Technical Report (March 4, 2004). <http://csmo.ici.ro/neculai/anman.htm>.
- [8] Higham, D.J., (1999). Trust region algorithms and time step selection. *SIAM J. Numer. Anal.* 37, 194–210.
- [9] Raydan, M., (1997). The Barzilai and Borwein method for the large-scale unconstrained minimization problem. *SIAM J. Optim.* 7, 26–33.
- [10] Brown, A.A., Biggs, M.C., (1989). ODE versus SQP methods. *J. Optim. Theory Appl.* 62, 371–386.
- [11] Gertz, E.M., (2004). A quasi-Newton trust region method. *Math. Program., Ser. A* 100, 447–470
- [12] Sun, J., Zhang, J.P., (2001). Global convergence of convergence of conjugate gradient methods without line search. *Ann. Oper. Res.* 103, 161–173.
- [13] Barzilai, J., Borwein, J.M. (1988). Two point step size gradient methods. *IMA J. Numer. Anal.* 8, 141–188.
- [14] Gill, P.E., Leonard, M.W., (2001). Reduced-Hessian quasi-Newton methods for unconstrained optimization. *SIAM J. Optim.* 12, 209–237.
- [15] J. Nocedal and S. J. Wright, (2006). *Numerical Optimization*, Springer.
- [16] Ou, Y.G., Zhou, Q., Lin, H.C., (2009). An ODE-based trust region method for unconstrained optimization problems. *J. Comput. Appl. Math.* 232, 318–326.
- [17] Erway, J.B., Marcia, R.F., (2017). On solving large-scale limited-memory quasi-Newton equations. *Linear Algeb. Appl.* 515, 196–225.
- [18] Snijman, J. A., (1982). A New and Dynamic Method for Unconstrained Optimization, *Appl. Math Model.*, 6, 449–462.
- [19] Nocedal, J., Yuan, Y.X., (1998). Combing trust region and line search techniques. *Adv. Non. Prog.* 53–175.
- [20] Modarres, F., Leong, W.J., (2014). Quasi-Newton methods based on ordinary differential equation approach for unconstrained nonlinear optimization. *Appl. Math. Comput.* 233, 272–291.
- [21] Ortega, J. M., Rheinboldt, W.C., (1970). *Iterative Solution of Nonlinear Equations in Several Variables*.
- [22] Andrei, N., (2008). An unconstrained optimization test functions collection. *Adv. Model. Optim.* 10(1), 147–161.
- [23] Han, L.X., (1993). On the convergence properties of an ODE algorithm for unconstrained optimization. *Math. Numer. Sin.* 15, 449–455.