برنامه ریزی غیرخطی پیچیده با استفاده از یک الگوریتم فراابتکاری و حل عدی آن

دکتر علی اصغر توفیق
استادیار دانشکده مهندسی صنایع، دانشگاه صنعتی امیرکبیر

منیره السادات محمودی(نویسنده مسئول مکاتبات)
کارشناس ارشد مهندسی صنایع، دانشگاه صنعتی امیرکبیر
mahmoudi.monireh@gmail.com

چکیده

از آنجایی که الگوریتم‌های دقیق قادر به یافتن جواب بهینه به صورت دقیق هستند، در مورد **مسائل بهینه سازی** سخت کارایی ندارند و زمان حل آنها در این مسائل به صورت نمایی افزایش می‌یابد. به علاوه الگوریتم‌های تقریبی قادر به یافتن جواب‌های خوب (نزدیک به بهینه) در زمان حل کوتاه برای مسائل بهینه‌سازی سخت هستند. یکی از مسائل برنامه ریزی غیر خطی، مسئله ی حداقل سازی حجم بیضی ای به مرکز O که مجموعه ای متناهی از نقاط فضا را پوشش می دهد، می باشد. این مسئله تنها در فضای دو بعدی و آن هم با راه حل های ابتکاری به سختی قابل حل است. اما در سه بعد و بالاتر به علت افزایش بی رویه ی حجم محاسبات، حل صریح این مسئله بسیار سخت و دشوار می شود. در حل این مسئله از الگوریتم SA استفاده شده و مسئله ی موردنظر در دو حالت فضای دو بعدی و سه بعدی حل شده و در فضای n بعدی نیز قابل تعمیم است.

کلید واژه:

مسئله حداقل سازی حجم بیضی، الگوریتم آنیلینگ شبیه سازی شده

مقدمه

روش‌ها و **الگوریتم‌های بهینه‌سازی** به دو دسته **الگوریتم‌های** دقیق[1] و **الگوریتم‌های تقریبی** [2]تقسیم‌بندی می‌شوند. الگوریتم‌های دقیق قادر به یافتن جواب بهینه به صورت دقیق هستند اما در مورد **مسائل بهینه سازی** سخت کارایی ندارند و زمان حل آنها در این مسائل به صورت نمایی افزایش می‌یابد. الگوریتم‌های تقریبی قادر به یافتن جواب‌های خوب (نزدیک به بهینه) در زمان حل کوتاه برای مسائل بهینه‌سازی سخت هستند. الگوریتم‌های تقریبی نیز به دو دسته الگوریتم‌های ابتکاری[3] و فراابتکاری[4] تقسیم‌بندی می شوند. دو مشکل اصلی الگوریتم‌های ابتکاری، قرار گرفتن آنها در بهینه‌های محلی و عدم قابلیت آنها برای کاربرد در مسائل مختلف است. الگوریتم‌های فراابتکاری یا متاهیوریستیک‌ها برای حل این مشکلات الگوریتم‌های ابتکاری ارائه شده‌اند. در واقع الگوریتم‌های فراابتکاری، یکی از انواع الگوریتم‌های بهینه‌سازی تقریبی هستند که دارای مکانیزم‌های خروج از بهینه محلی می‌باشند و قابل کاربرد در طیف وسیعی از مسائل هستند . از الگوریتم‌های متداول فراابتکاری مبتنی بر جمعیت می‌توان الگوریتم‌های تکاملی، **الگوریتم ژنتیک**، برنامه‌ریزی ژنتیک، **بهینه‌سازی کلونی مورچگان**، کلونی زنبورها و ... را نام برد. از الگوریتم‌های متداول فراابتکاری مبتنی بر یک جواب می‌توان **الگوریتم جستجوی ممنوعه** و

الگوریتم تبرید شبیه‌سازی شده را نام برد[۱].

مسئله ی حداقل سازی حجم بیضی ای به مرکز مبدأ مختصات که مجموعه ای متناهی از نقاط فضا را پوشش دهد[۵]. یکی از مسائل معروف حوزه ی برنامه ریزی غیرخطی است که تاکنون الگوریتم های مختلفی برای حل این مسئله ارائه شده است. در این الگوریتم ها دو رویکرد متفاوت به چشم می خورد:

۱- رویکردهای تئوریک که مسئله را به صورت تئوریک و با استفاده از قضایای مختلف این حوزه حل می کنند.

۲- رویکردهای عددی که مسئله را با روش های عددی حل کرده و تا حد ممکن به جواب بهینه نزدیک می شوند.

در رویکرد اول با استفاده از قوانین جبر ماتریس ها و قضایای سخت و پیچیده ی حوزه برنامه ریزی غیر خطی[۶] راه حل صریحی را در یکی از ساده ترین حالت های مسئله ارائه داده اند که در ادامه به آن خواهیم پرداخت[۲].

در رویکرد دوم با استفاده از روش های نیوتن[۷]. و الگوریتم نقاط درونی[۸].افراد مختلفی از جمله J-L. Goffin، Z-Q Luo و p.E.Gill، W.Murray، M.A Saunders، J.A Tomlin، M.H. wright راه حل های عددی را برای این مسئله ارائه کرده اند [۴] ،[۳].

۱.بیان مسئله:

یکی از مسائل معروف در حوزه برنامه ریزی غیر خطی که حل صریح بسیار طولانی حتی در ابعاد کوچک فضا دارد، مسئله حداقل سازی حجم بیضی ای به مرکز مبدأ مختصات که مجموعه ای متناهی از نقاط فضا را پوشش دهد می باشد. در این مسئله، مجموعه ای متناهی از نقاط در فضای تعریف شده ی دو بعدی یا سه بعدی فرض می شود، مسئله این است که بیضی یا بیضی گونی به مرکز o را بیابیم که تمام این نقاط را بپوشاند و کمترین مساحت در فضای دو بعدی و کمترین حجم را در فضای سه بعدی داشته باشد. مسئله بیان شده در بالا به زبان ریاضی به صورت زیر تعریف می شود:

$$\min V(E_Q)$$

$$s.t:$$

$$Span(y_1,\dots,y_m) = R^n \qquad m \geq n$$

$$y_1 \in E_Q, E_Q = \{x, x^T X^2 x \leq 1\}$$

همانگونه که در مدل بالا نیز مشاهده می شود، تعداد نقاط تعریف شده در مسئله بایستی حداقل به تعداد بعد فضای مسئله مورد نظر باشد. مسئله ی کمترین حجم بیضی معادل مسئله ی زیر است:

$$Min - \log\det X$$

$$s.t:$$

$$\|Xy_i\|_2 \leq 1$$

$$X \succ 0$$

از آنجایی که مقدار دترمینان یک ماتریس، برابر حاصلضرب مقادیر ویژه آن ماتریس است و مقادیر ویژه در ماتریس X، همان طول قطرهای بیضی می باشد، ارتباط بین حجم بیضی و $-\log\det x$ قابل توجیه است [۲].

این مسئله شامل دو محدودیت است:

الف. محدودیت $\|Xy_i\|_2 \le 1$:

توضیحی کوتاه در رابطه با نرم اقلیدسی[9] $\|X\|_2$ (نرم دو):

نرم اقلیدسی یا نرم دو، توسط ضرب داخلی دو بردار تعریف می شود و عبارتست از:

$$\|x\|_E^2 = \|x\|_2^2 = \langle x, x\rangle_E = x^T x$$

محدودیت $\|Xyi\|_2 \le 1$ معادل محدودیت $\|Xy_i\|_2^2 \le 1$ است. این عبارت نیز هم ارز عبارت $y_i^T X^T X y_i \le 1$ می باشد که در آن y_i یک ماتریس $n\times n$ و X یک ماتریس $n\times n$ می باشد [۱].

ب. محدودیت $X \succ 0$:

این محدودیت به معنای آن است که ماتریس X[10] یک ماتریس **pd** است.

تعریف ماتریس **pd**:

ماتریس A را یک ماتریس معین مثبت یا همان **pd** می نامند اگر و فقط اگر:

$$\forall x \in R^n, x \neq 0 : x^T A x > 0$$

اما راه ساده تری نیز برای تشخیص اینکه یک ماتریس معین مثبت (pd) است، وجود دارد:

اگر از گوشه شمال غربی ماتریس A، دترمینان ماتریس های مربعی را Δ_1، Δ_2 و ... که به ترتیب مرتبه ی ۱، ۲ و ... هستند در نظر بگیریم، یعنی: ...و $\Delta_2 = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ و $\Delta_1 = a_{11}$، آنگاه اگر همه ی Δ_i ها $(1 \le i \le n)$ مثبت باشند A مثبت معین است[۴].

۱. *حل مسئله حداقل سازی حجم بیضی که مجموعه ای متناهی از نقاط فضا را در فضای دو بعدی پوشش می دهد به روش صریح:*

صورت مسئله را یک بار دیگر می آوریم:

$$Min - \log\det X$$
$$s.t:$$
$$\|Xy_i\|_2 \le 1$$
$$X \succ 0$$

تابع هدف F(X) عبارت است از:

$$f(x) = \log\det X^{-1} = -\log\det X$$

و محدودیت g۱(X) عبارتست از:

$$\|Xy_i\|_2 \le 1 \equiv \|Xy_i\|_2^2 \le 1 \rightarrow g_1(x) = \|Xy_i\|_2^2 - 1$$

مشتق سویی[11] محدودیت g۱(X) در جهت بردار d برابر است با:

$$g'_{1d}(X) = \lim_{t \to 0} \frac{g_1(X+tD) - g_1(X)}{t}$$

$$g'_{1d}(X) = \lim_{t \to 0} \frac{\langle (X+tD)y_i, (X+tD)y_i \rangle - 1 - \langle Xy_i, Xy_i \rangle + 1}{t}$$

$$g'_{1d}(X) = \lim_{t \to 0} \frac{(Xy_i)^T X y_i + (Xy_i)^T (tDy_i) + (tDy_i)^T X y_i + (tDy_i)^T (tDy_i) - (Xy_i)^T X y_i}{t} =$$

$$y_i^T X^T D y_i + y_i^T D^T X y_i = 2 y_i^T X^T D y_i = < 2 y_i y_i^T X, D >_F$$

پس در نتیجه خواهیم داشت:

$$\nabla g_1(\bar{x}) = 2 y_i y_i^T X$$

در اینجا لازم است یک سری تعاریف و قضایای مهم در برنامه ریزی غیر خطی آورده شود تا برای ادامه حل مسئله به روش صریح به کار گرفته شود[۲].

تعریف فضای هیلبرت[۱۲]: به فضای ضرب داخلی گفته می شود که کامل باشد[۲]

تعریف مجموعه ی کامل[۱۳]: یک مجموعه X را در نظر بگیرید. این مجموعه کامل است اگر هر دنباله ی کوشی در آن همگرا به یک نقطه ی داخل آن مجموعه همگرا باشد[۲].

تعریف دنباله کوشی[۱۴][۲] :

$$\forall \varepsilon, \quad \exists N : \forall n, m \geq N \quad \rho(x_n, x_m) < \varepsilon$$

قضیه ۱:۱ First Order Fritz-John Necessary Condition ($^{\circ}FJNC$) : فرض کنید $\bar{x}$ نقطه درونی مجموعه ی X' باشد و توابع f و g روی X' مشتق پذیر باشند و V نیز یک فضای هیلبرت باشد. اگر $\bar{x}$ یک حداقل محلی باشد آنگاه:

$$\exists \gamma_1, \ldots, \gamma_m \geq 0 : \gamma_0 \nabla f(\bar{x}) + \sum_{i=1}^{m} \gamma_i \nabla g_i(\bar{x}) = 0$$

$$\gamma_i g_i(\bar{x}) = 0$$

قضیه ۲: First Order Mangasarian-Fromoritz Constraint Qualification ($1^{\circ}MFCQ$):

شرط مرتبه اول MFCQ در $\bar{x}$ برقرار است اگر:

$$\exists d: < \nabla g_i(\bar{x}), d > < 0$$

به ازای محدودیت هایی که $\bar{x}$ در آنها فعال باشد.

قضیه ۳: First Order KKT Necessary Condition ($1^{\circ}KKTNC$): همه فرضیات $1^{\circ}FJNC$ را در نظر بگیرید، به علاوه اینکه در $\bar{x}$ یکی از شرایط کیفیت مرتبه اول (مثلاً $1^{\circ}MFCQ$) برقرار باشد. آنگاه:

$$\exists U_1, \ldots, U_m \geq 0$$

$$\nabla f(\bar{x}) + \sum_{i=1}^{m} U_i \nabla g_i(\bar{x}) = 0$$

$$U_i g_i(\bar{x}) = 0$$

ادامه حل مسئله: می خواهیم ببینیم که شرط کیفیت مرتبه اول MFCQ برقرار است یا نه. D را برابر X– قرار می دهیم:

$$< \nabla g_1(X), -X >_F = -2 X^T y_i y_i^T X = -2tr(A^T A) = -2 < A, A >_F < 0$$

پس شرط کیفیت مرتبه اول MFCQ برقرار است.

طبق قضیه ی $(1^{st}KKTNC)$:

$$\exists U_1, \dots, U_m \geq 0$$

$$\nabla f(\bar{x}) + \sum_{i=1}^{m} U_i \nabla g_i(\bar{x}) = 0$$

$$U_i g_i(\bar{x}) = 0$$

$\nabla g_1(\bar{x})$ را در صفحات قبلی محاسبه نمودیم. حال باید $\nabla f(\bar{x})$ را محاسبه نماییم:

برای این کار کافیست مشتق سویی تابع $f(\bar{x})$ را در جهت بردار d بیابیم که مسئله ی نسبتاً پیچیده ای است.

$$f'_d(X) = \lim \frac{log\,det(X + tD) - log\,det(X)}{t} = ?$$

$$t \to 0 \;, D \in S_n^{++}$$

$$\log\det(X + tD) = \log\det\left[X^{1/2}(I + X^{-\frac{1}{2}}(tD)X^{-\frac{1}{2}})X^{1/2}\right] =$$

$$\log\det(X^{\frac{1}{2}}) + \log\det\left[I + X^{-\frac{1}{2}}(tD)X^{-\frac{1}{2}}\right] + \log\det(X^{\frac{1}{2}}) =$$

$$\log\det(X) + \log\det\left[I + X^{-\frac{1}{2}}(tD)X^{-\frac{1}{2}}\right]$$

$$f'_d(X) = \lim \frac{log\,det(X + tD) - log\,det(X)}{t}$$

$$= \lim \frac{\log\det(X) + \log\det\left[I + X^{-\frac{1}{2}}(tD)X^{-\frac{1}{2}}\right] - \log\det(X)}{t}$$

$$t \to 0 \;, D \in S_n^{++}$$

ماتریس pt را به صورت روبرو تعریف می کنیم:

$$p_t = X^{-\frac{1}{2}}(tD)X^{-\frac{1}{2}} \qquad t \to 0, p_t \to 0$$

$$\log\det(X + tD) = \log\det(I + p_t) = \log\prod_{i=1}^{n}(1 + \lambda_i(p_t))$$

$$\lambda_i(p_t) \to 0$$

وقتی یک ماتریس را با ماتریس I جمع می کنیم، تمامی عناصر روی قطر آن به میزان یک واحد افزایش می یابند، در نتیجه مقادیر ویژه آن نیز به میزان یک واحد افزایش می یابند.

اثبات: در تعریف مقادیر ویژه داریم که:

$$AX = \lambda X \;\rightarrow\; |A - \lambda I| = 0$$

حال اگر ماتریس I با ماتریس A جمع شود، مقادیر ویژه آن به صورت زیر تغییر می کنند:

$$(A + I)X = \lambda' X$$

$$|A + I - \lambda' I| = 0$$

$$|A - (\lambda' - 1)I| = 0$$

$$\lambda = \lambda' - 1 \;\rightarrow\; \lambda' = \lambda + 1$$

در ادامه، $\log\prod_{i=1}^{n}(1+\lambda_i(p_t))$ برابر است با:

$$\log\prod_{i=1}^{n}(1+\lambda_i(p_t)) = \sum_{i=1}^{n}\log(1+\lambda_i(p_t)) = \sum_{i=1}^{n}\lambda_i(p_t)$$

$$\lambda_i(p_t) \to 0$$

$$\sum_{i=1}^{n}\lambda_i(p_t) = tr(p_t) = tr(X^{-\frac{1}{2}}(tD)X^{-\frac{1}{2}})$$

از طرفی با توجه به قوانین موجود در جبر ماتریس ها، می دانیم که:

$$tr\left(X^{-\frac{1}{2}}(tD)X^{-\frac{1}{2}}\right) = tr(X^{-1}D) = \ll X^{-1}, D \gg_F$$

پس در نتیجه خواهیم داشت:

$$\nabla f(\tilde{x}) = X^{-1}$$

$$-X^{-1} + 2\sum_{i=1}^{n}U_i\,y_i y_i^T X = 0 \to X^{-1} = 2\sum_{i=1}^{n}U_i\,y_i y_i^T X \to 1 = 2\sum_{i=1}^{n}U_i\,y_i y_i^T X^2$$

پس داریم:

$$X^2 = \frac{1}{2}\left(\sum_{i=1}^{n}U_i\,y_i y_i^T\right)^{-1}$$

مسئله را در ساده ترین حالت ممکنه یعنی برای دو نقطه ی $y_1 = \binom{a}{0}, y_2 = \binom{0}{b}$ حل می کنیم[۵]:

پس:

$$X^2 = \frac{1}{2}\left(\sum_{i=1}^{n}U_i\,y_i y_i^T\right)^{-1} = \frac{1}{2}\left(U_1\begin{pmatrix}a^2 & 0\\ 0 & 0\end{pmatrix} + U_2\begin{pmatrix}0 & 0\\ 0 & b^2\end{pmatrix}\right)^{-1} = \frac{1}{2}\begin{pmatrix}\frac{1}{U_1 a^2} & 0\\ 0 & \frac{1}{U_2 b^2}\end{pmatrix}$$

اگر روی محدودیت $\|Xy_i\|_2 \le 1$ قرار داشته باشیم:

$$\|Xy_i\|_2 = 1 \to y_1^T X^2 y_1 = 1\ ,\ y_2^T X^2 y_2 = 1$$

پس:

$$\frac{1}{2}\binom{a}{0}^T\begin{pmatrix}\frac{1}{U_1 a^2} & 0\\ 0 & \frac{1}{U_2 b^2}\end{pmatrix}\binom{a}{0} = 1 \to \frac{1}{2}\times\frac{1}{u_1} = 1 \to u_1 = \frac{1}{2}$$

و به طور مشابه، $u_2 = \frac{1}{2}$ بدست می آید.

پس : $X^2 = \frac{1}{2}\begin{pmatrix}\frac{1}{1/2a^2} & 0\\ 0 & \frac{1}{1/2b^2}\end{pmatrix}$ و در نتیجه $X^2 = \begin{pmatrix}\frac{1}{a^2} & 0\\ 0 & \frac{1}{b^2}\end{pmatrix}$ بدست می آید.

محدودیت ۱≤ ۲‖Xyi‖ یا به عبارتی $y_i^T X^T X y_i \le 1$ بیضی مورد نظر را می دهد و از آنجا که $X^T X$ همان X^2 می باشد با بدست آمدن مقدار X^2 به جواب نهایی رسیده ایم. به واقع ماتریس X^2 بیانگر بیضی یا بیضی گونی به مرکز o است که تمام مجموعه نقاط داده شده را بپوشاند و کمترین مساحت در فضای دو بعدی و کمترین حجم را در فضای سه بعدی داشته باشد[۵].

۴ .حل مسئله با استفاده از الگوریتم Simulated Annealing (SA):

فرایند طراحی و پیاده‌سازی الگوریتم‌های فراابتکاری دارای سه مرحله‌ی متوالی است که هر کدام از آن‌ها دارای گام‌های مختلفی هستند. در هر گام فعالیت‌هایی باید انجام شود تا آن گام کامل شود. مرحله‌ی ۱ آماده‌سازی است که در آن باید شناخت دقیقی از مسئله‌ای که می‌خواهیم حل کنیم بدست آوریم، و اهداف طراحی الگوریتم فراابتکاری برای آن باید با توجه به روش‌های حل موجود برای این مسئله به طور واضح و شفاف مشخص شود. مرحله‌ی بعدی، ساخت نام دارد. مهمترین اهداف این مرحله انتخاب استراتژی حل، تعریف معیارهای اندازه گیری عملکرد، و طراحی الگوریتم برای استراتژی حل انتخابی می‌باشد. آخرین مرحله پیاده‌سازی است که در آن پیاده‌سازی الگوریتم طراحی شده در مرحله‌ی قبل، شامل تنظیم پارامترها، تحلیل عملکرد، و در نهایت تدوین و تهیه گزارش نتایج باید انجام شود[۱].

تعریف مسئله و طرح کدگذاری مسئله: ابتدا باید بعد فضا را مشخص کنیم، یعنی به زبان ساده تر تعیین کنیم که فضای ما دو بعدی یا سه بعدی می باشد. همانطور که می دانیم در فضای بیش از سه بعد، مفهوم بیضی به عنوان شکلی در فضا تغییر کرده و تنها می توان این موضوع را با عبارات ریاضی و ماتریس ها بیان کرد و مفهوم توپولوژیکی بیضی را نمی توان برای آن متصور شد. به همین دلیل از حل مسئله در فضایی بیش از سه بعد پرهیز نموده ایم.

جواب های مسئله به صورت ماتریس $X_{n\times n}$ در نظر گرفته می شود. مجموعه ای از نقاط پراکنده در فضا را در نظر بگیرید. اما پیش از آن باید تعداد نقاطی را که به عنوان ورودی می خواهیم به الگوریتم بدهیم را مشخص کنیم. لازم به ذکر است که تعداد نقاط تعریف شده در مسئله بایستی حداقل به تعداد بعد فضای مسئله مورد نظر باشد. چراکه این نقاط باید کل فضا را Span کنند. سپس نقاط را وارد می کنیم. بزرگترین بیضی به مرکز o که تمام نقاط را پوشش دهد، دایره ای به مرکز o و به شعاع "فاصله ی نقطه ای که بیشترین فاصله را با مرکز مختصات دارد تا مرکز مختصات" می باشد. اگر این فاصله " فاصله ی نقطه ای که کمترین فاصله را با مرکز مختصات دارد تا مرکز مختصات" را با miny نمایش دهیم، تمامی عناصر ماتریس X بین ۱/miny− تا ۱/miny+ قرار دارند.

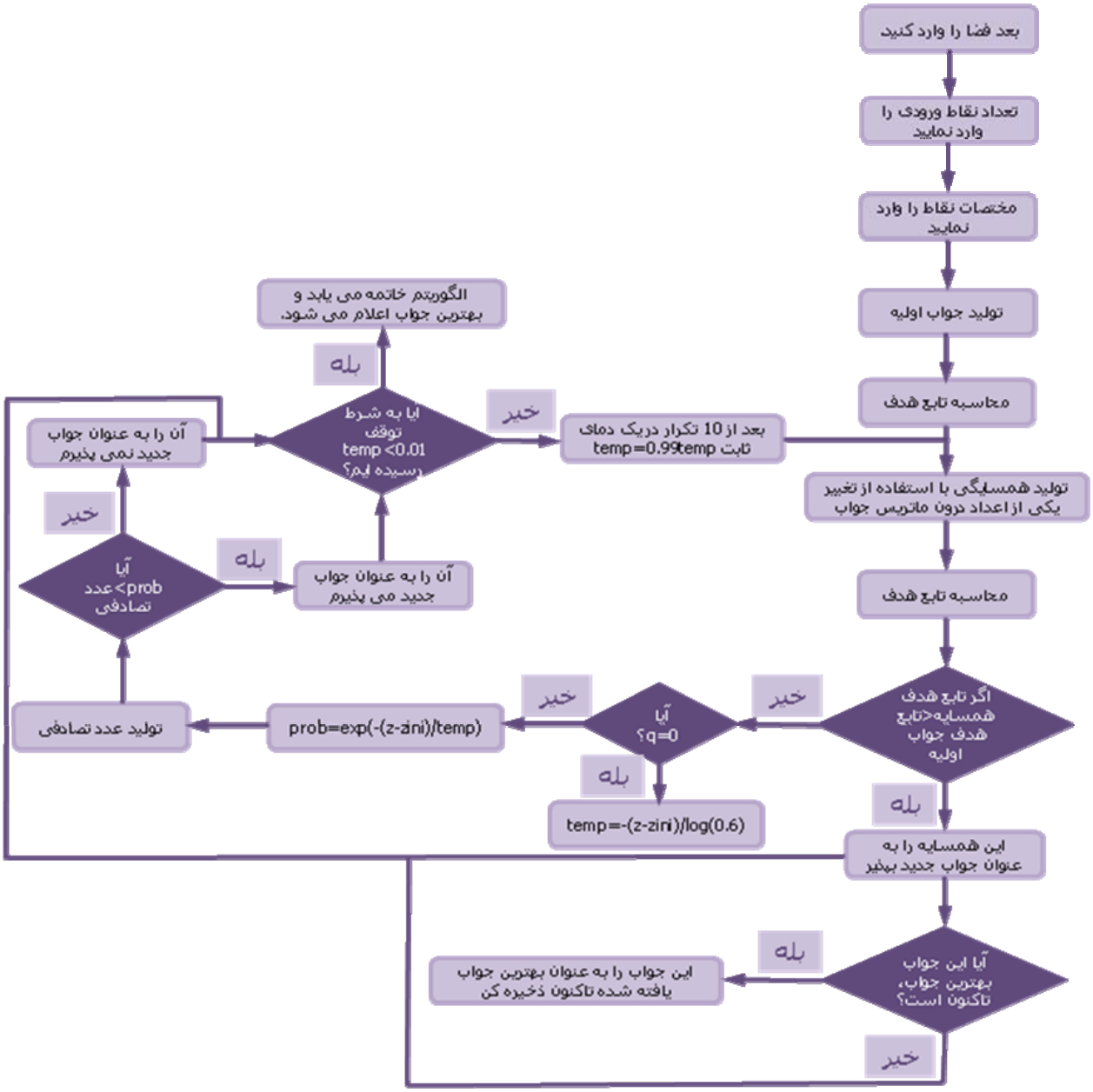
برای تعریف همسایگی این ماتریس نیز در هر تکرار، یکی از عناصر ماتریس را به صورت کاملاً تصادفی انتخاب کرده و عددی تصادفی را در بازه [۱/miny,۱/miny−] به آن می دهیم. پس ابتدا لازم است که فاصله ی تمامی نقاط را تا مرکز مختصات بدست آوریم و نقطه ای که کمترین فاصله را تا مرکز مختصات دارد، پیدا کنیم. برای تولید جواب اولیه، کل عناصر ماتریس X به صورت تصادفی انتخاب می شود. این ماتریس بایستی در محدودیت های یک و دو صدق کند.

دمای فرضی اولیه برابر ۳ در نظر گرفته می شود تا حل الگوریتم آغاز شود. در صورتیکه جواب های بعدی مقدار تابع هدفی کمتر از مقدار تابع هدف جواب اولیه داشته باشند، آن جواب پذیرفته شده و در صورتیکه مقدار تابع هدفی بیشتر از مقدار تابع هدف جواب فعلی داشته باشند با احتمالی پذیرفته می شوند. این احتمال از تابع احتمال بولتزمن تبعیت می کند.

برای محاسبه دمای شروع، مقدار احتمال تابع بولتزمن برابر ۰/۶ در نظر گرفته می شود و مقدار دمای اولیه محاسبه می شود. نرخ کاهش دما به صورت هندسی و با سعی و خطای بسیار، به صورت temp=۰,۹۹temp در نظر گرفته شده است. در هر دما ۱۰ تکرار وجود دارد. این بدان معناست که در هر دما، ۱۰ همسایه ی مختلف ماتریس X بررسی می شود. لازم به ذکر است که بررسی

این تعداد همسایگی نیز پس از سعی و خطای بسیاری بدست آمد و الگوریتم با در نظر گرفتن ۳ همسایگی، ۵ همسایگی، ۱۵ همسایگی، ۲۰ همسایگی و ... اجرا شد و بهترین جواب هایی که در زمان نسبتاً مناسبی بدست می آمدند، در تعداد ۱۰ همسایگی بود. برای تراز کردن[۱۵] پارامترهای الگوریتم، مقدار دمای اولیه[۱۶] را به مرور از ۱۰۰۰ تا ۱۰۰ تغییر دادیم و بهبود چندانی در تابع هدف بدست نیامد. اما به مرور که مقدار دمای اولیه را کاهش دادیم به نتایج بهتری رسیدیم.

نکته ای که در اینجا بسیار حائز اهمیت است، اینست که مقدار دمای اولیه وابستگی شدیدی به مختصات نقاط ورودی دارد. در صورتی که مختصات این نقاط مقادیری کوچکتر از یک باشند، برای مقدار دهی به دمای اولیه بایستی از دماهای بالا مانند ۱۰۰۰ شروع کرد. اما در صورتی که مختصات این نقاط مقادیری بزرگتر از یک باشند، شروع از دماهای نسبتاً پایین بهتر است. علت این امر را می توان به کوچک بودن مقدار (z-zini)– نسبت داد. این عبارت برای نقاطی با مختصات بزرگتر از یک، مقداری کوچک است، حال آنکه این عبارت برای نقاطی با مختصات کوچکتر از یک، مقداری بزرگ می باشد. بدنه اصلی الگوریتم پیشنهادی، به صورت Flowchart در نمودار (۱) توضیح داده شده است:



نمودار(۱) الگوریتم حل مسئله به روش SA

نتایج:

نتایج حاصل از ۱۰ بار اجرا کردن الگوریتم به ازای نقاط مختلف در فضاهای دو بعدی و سه بعدی در زیر آمده است:جداول (۱-۵)

جدول (۱) نتایج اجرای در فضای دو بعدی با فرض $n=۲; m=۲; y_۱=(۱،۳);y_۲=(۰،۵)$

	Best	Worst	Average
memorizemin	۱,۷۱۲۵	۱,۷۳۱۳	۱,۷۲۱۰
bestx	$\begin{bmatrix} 0.0994 & 0.0979 \\ -0.0994 & 0.0972 \end{bmatrix}$	$\begin{bmatrix} 0.0986 & 0.0898 \\ -0.0994 & 0.0977 \end{bmatrix}$	$\begin{bmatrix} 0.0972 & 0.0983 \\ -0.0987 & 0.0957 \end{bmatrix}$
time(seconds)	۹۲,۱۵۱۵۵۲	۱۲,۰۷۷۶۵۶	۳۹,۱۶۸۱۹۷

جدول(۲) نتایج اجرا در فضای دو بعدی با فرض $n=۲; m=۲; y_۱=(۱،۴);y_۲=(۲،۵)$

	Best	Worst	Average
memorizemin	۲,۱۶۷۶	۲,۱۷۳۴	۲,۱۷۳۴
bestx	$\begin{bmatrix} 0.0588 & 0.0587 \\ -0.0575 & 0.0582 \end{bmatrix}$	$\begin{bmatrix} 0.0585 & 0.0576 \\ -0.0585 & 0.0572 \end{bmatrix}$	$\begin{bmatrix} 0.0585 & 0.0576 \\ -0.0585 & 0.0572 \end{bmatrix}$
time(seconds)	۲۶,۷۰۲۶۶۴	۴۹,۴۶۶۱۸۹	۵۴,۶۵۹۸۶۵

جدول (۳) نتایج اجرا در فضای سه بعدی با فرض $n=۳; m=۳; y_۱=(۱،۹،۱);y_۲=(۲،۵،۲);y_۳=(۲،۱،۳)$

	Best	Worst	Average
memorizemin	۲,۸۹۷۱	۲,۹۰۳۳	۲,۸۹۷۳
bestx	$\begin{bmatrix} 0.0671 & -0.0680 & 0.0629 \\ 0.0674 & 0.0706 & 0.0669 \\ -0.0711 & 0.0343 & 0.0683 \end{bmatrix}$	$\begin{bmatrix} 0.0709 & 0.0705 & -0.0651 \\ -0.0682 & 0.0351 & -0.0657 \\ -0.0658 & 0.0695 & 0.0634 \end{bmatrix}$	$\begin{bmatrix} 0.0713 & -0.0416 & 0.0700 \\ 0.0703 & 0.0682 & -0.0707 \\ -0.0628 & 0.0610 & 0.0700 \end{bmatrix}$
time(seconds)	۴۷,۱۱۳۶۶۶	۴۳,۷۸۷۲۷۸	۵۸,۸۰۵۱۳۳

جدول (۴) نتایج اجرا در فضای سه بعدی با فرض $n=۳; m=۳; y_۱=(۱،۵،۲);y_۲=(۸،۹،۲);y_۳=(۲،۳،۶)$

	Best	Worst	Average
memorizemin	۳,۸۵۹۴	۳,۸۷۵۵	۳,۸۶۳۱
bestx	$\begin{bmatrix} 0.0331 & 0.0327 & -0.032 \\ -0.0316 & 0.0171 & -0.032 \\ -0.0292 & 0.0333 & 0.0330 \end{bmatrix}$	$\begin{bmatrix} 0.0333 & -0.0308 & -0.033 \\ 0.0038 & 0.0304 & -0.033 \\ 0.0319 & 0.0321 & 0.0325 \end{bmatrix}$	$\begin{bmatrix} 0.0281 & -0.0326 & 0.031 \\ 0.0322 & 0.0327 & 0.032 \\ -0.0331 & 0.0306 & 0.032 \end{bmatrix}$
time(seconds)	۵۱,۶۵۶۷۴۰	۳۳,۲۶۶۰۳۸	۵۶,۴۳۳۲۶۳

جدول(۵) نتایج اجرا در فضای سه بعدی نهایی $n=۳;\ m=۴;\ y_۱=(۱،۱،۱);y_۲=(۲،۷،۲);y_۳=(۳،۱،۹);y_۴=(۴،۱،۸)$

	Best	Worst	Average
memorizemin	۲,۰۹۹۱	۲,۱۳۴۷	۲,۱۰۸۳
bestx	0.3237 0.0334 −0.182 −0.2173 0.0312 −0.011 −0.3308 0.1764 0.119	0.1371 −0.0201 0.053 −0.3173 0.1699 0.111 −0.3182 −0.0341 0.153	0.3320 0.0021 −0.130 −0.1993 0.0484 0.1603 0.3089 −0.1446 −0.041
time(seconds)	۱۰۱,۱۸۷۹۳۶	۷۳,۶۴۰۴۴۶	۱۰۱,۵۷۹۲۶۷

همانگونه که از جداول (۱ تا۵) بالا پیداست، در فضای دو بعدی جواب ها بسیار نزدیک به جواب ها در حل به روش صریح می باشد و در حالت سه بعدی نیز اختلاف بین بهترین و بدترین جواب بسیار اندک است. اگرچه حل به روش صریح به جواب دقیق منتج می شود، اما نیازمند صرف وقت زیاد و انجام محاسبات ماتریسی پیچیده است. در حالیکه الگوریتم آنیلینگ شبیه سازی شده، که یکی از روشهای فراحسی[۱۷] برای حل مسائل بسیار پیچیده به کار می رود، مراحل رسیدن به جواب مناسب را بسیار کوتاه کرده است. در حقیقت، پیداکردن بیضی یا بیضی گونی که کل نقاط مفروض را پوشش دهد و کمترین مساحت (حجم در فضای سه بعدی) را اشغال کند یکی از مسائل پرکاربرد در دنیای واقعی محسوب می شود که بنا به پیچیدگی محاسبات حل صریح معمولاً روش های عددی مختلف برای حل آن مرجح است.

منابع

الگوریتم_های_فراابتکاری/http://fa.wikipedia.org/wiki

Convex Optimization, Stephen Boyd & Lieven Vandenberghe, Cambridge university press, ۲۰۰٤.

P.E.Gill, G.H.Golub, W.Murray, and M.A.Saunders. Methods for modifying matrix factorizations, Mathematics of Computation ۲۸(۱۲٦), ۵۰۵-۵۳۵ (۱۹۷٤).

Non Linear Programming, Theory and Algorithms, Mokhtar S.Bazaraa, Hanif D.Sherali, C.M.Shetty, John Wiley & sons, ۱۹۹۳.

C. Davis. Notions generalizing convexity for functions defined on spaces of matrices. In V. L. Klee, editor, convexity, volume VH of proceedings of the symposia in Pure Mathematics, page ۱۸۷-۲۰۱, American Mathematical society, ۱۹٦۳.

پیوست ۱

کد الگوریتم SA برای مسئله *minimum volume ellipsoid covering a finite set of points*

```
tic
disp('please enter dimension of the space(2 or 3).')
n=input('n=');
disp('how many points you want to enter?')
disp('(You should enter at least n points to span the space.)')
m=input('m=');
disp('please enter the points which span the space.')
for i=1:m
    for j=1:n
        y{i}(j)=input('y=');
    end
end
for i=1:m
    a(i)=0;
    for j=1:n
        a(i)=a(i)+(y{i}(j))^2;
    end
end
miny=a(1);
for i=2:m
    miny=min(miny,a(i));
end
temp=3;q=0;o=1;memorizemin=1000000;
while temp>0.01
  for i=1:10
        t=1;
        while t~=0
            p=1;
            if o==1
                for i=1:n
                    for j=1:n
                        x(i,j)=random('unif',-1/miny,1/miny);
                    end
                end
                s=x;
```

```
    else
        row=random('unid',n);
        column=random('unid',n);
        s=x;
        s(row,column)=random('unif',-1/miny,1/miny);
    end
    for i=1:m
        c(i)=y{i}*s'*s*(y{i})';
        if c(i)>1
            p=0;
        end
    end
    if n==2 & p==1
        if det(s)>0 & s(1,1)>0
            t=0;
            o=o+1;
        end
    else
        if p==1
            for i=1:2
                for j=1:2
                    z(i,j)=s(i,j);
                end
            end
            if det(s)>0 & det(z)>0 & s(1,1)>0
                t=0;
                o=o+1;
            end
        end
    end
end
if o==2
    zini=-log10(det(s));
else
    z=-log10(det(s));
    if z<zini
        zini=z;
        x=s;
    else
        if q==0
            temp=-(z-zini)/log(0.6)
            q=1;
            rn=random('unif',0,1);
            if rn<0.6
                zini=z;
                x=s;
            end
        else
            rn=random('unif',0,1);
            if exp(-(z-zini)/temp)>rn
                zini=z;
                x=s;
            end
        end
```

```
            end
        end
        memorizemin=min(memorizemin,zini)
        if memorizemin==zini
            bestx=x;
        end
    end
    temp=0.99*temp;
end
memorizemin
bestx
toc
```

پی نوشت

۱ *exact*
۲ *approximate algorithms*
۳ *heuristic*
٤ *meta-heuristic*
٥ *Minimum volume ellipsoid covering a finite set of points*
٦ *Stephen Boyd& Lieven Vandenberghe*
۷ *Pure Newton & Damp Newton*
۸ *Euclidean Norm*
۹ *Interior Point Method*
۱۰ *Positive Definite Matrix (pd)*
۱۱ *Directional Derivative*
۱۲ *Hilbert Space*
۱۳ *Complete set*
۱٤ *Cauchy Sequence*
۱٥ *Tune*
۱٦ *Initial temperature*
۱۷ *Meta heuristic methods*