رویکردی مبتنی بر الگوریتم ژنتیکی به حلّ مسئله‌ی قطب‌گماری با بازخور خروجی

ناصر ساداتی (دانشیار)
آزمایشگاه سیستم‌های هوشمند
دانشکده‌ی مهندسی برق، دانشگاه صنعتی شریف

در این نوشتار، روش جدیدی برای حلّ مسئله‌ی قطب‌گماری با بازخور (فیدبک) خروجی ثابت پیشنهاد شده است. روش پیشنهادی برمبنای الگوریتم ژنتیکی شکل گرفته و می‌تواند تمامی قطب‌های سیستم حلقه‌بسته را با دقت بسیار در مکان‌های مورد نظر جای دهد. به‌شرط آن که جوابی برای مسئله وجود داشته باشد. همچنین نشان داده شده است که چگونه این الگوریتم، منجر به حلّ مشکلات موجود در روش‌های کلاسیک قطب‌گماری با بازخور خروجی ثابت شده است. مثال‌های عددی در پایان این نوشتار، کاربرد روش پیشنهادی را نشان می‌دهد.

مقدمه

سیستم خطی تغییرناپذیر با زمان زیر را در نظر بگیرید:

$$\dot{x}(t) = A\,x(t) + B\,u(t)$$
$$y(t) = C\,x(t) \qquad (۱)$$

که در آن، $x \in R^n$، $u \in R^m$ و $y \in R^l$. همچنین A، B و C ماتریس‌های ثابت حقیقی با ابعاد مناسب‌اند. ماتریس‌های B و C، از رتبه‌ی کامل فرض شده و جفت (A, B) کنترل‌پذیر و جفت (C, A) رؤیت‌پذیرند. مسئله‌ی تحت بررسی، یافتن ماتریس K با ابعاد $m \times l$ به گونه‌یی است که قطب‌های سیستم حلقه‌بسته تحت قانون بازخور $u = K y$ در مکان‌های متقارن $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ قرار گیرند.

در دو دهه‌ی اخیر، مسئله‌ی قطب‌گماری با بازخور خروجی ثابت، برای سیستم‌های چندورودی ـ چندخروجی مورد مطالعه قرار گرفته است. اوّلین نتایج نشان می‌دهد که اگر سیستم کنترل‌پذیر و رؤیت‌پذیر باشد، ماتریس بازخور خروجی ثابت می‌تواند $max\{m, l\}$ قطب سیستم حلقه‌بسته را تقریباً به‌دلخواه تعیین کند.[۱-۳] مطالعات بیشتر نیز نشان داده است که تحت شرایط مشخصی، $min\{n, m + l - ۱\}$ قطب به‌دلخواه تعیین می‌شود.[۴-۶] بنابراین اگر $n \le m + l - ۱$ باشد، قطب‌گماری کامل با بازخور خروجی امکان‌پذیر است. اخیراً نیز ثابت شده است که شرط $m \times l \ge n$ ـ که محدودیت کمتری ایجاد می‌کند ـ می‌تواند برای بررسی امکان قطب‌گماری مورد آزمایش قرار گیرد.[۱۶-۱۹] در اکثر این روش‌ها، الگوریتم یافتن ماتریس بازخور خروجی از لحاظ عددی مناسب نیست، به‌علاوه بعضی از این الگوریتم‌ها فقط می‌توانند قطب‌های سیستم حلقه‌بسته را در مکان‌های مجزا جای دهند و در برخی دیگر از این روش‌ها، قطب‌های سیستم حلقه‌بسته نمی‌توانند منطبق بر قطب‌های سیستم حلقه‌باز باشد.[۱۰-۱۶]

تعیین همزمان مقادیر ویژه و بردارهای ویژه (ساختار ویژه)، روش دیگری برای حل مسئله‌ی قطب‌گماری با بازخور خروجی است. اگر این روش در حوزه‌ی فرکانس به‌کار گرفته شود، معادله‌ی مشخصه‌ی سیستم حلقه‌بسته با استفاده از خواص دترمینان‌ها و مشتق‌های آنها مورد بحث قرار می‌گیرد و در نتیجه، شکل بسته‌یی برای ماتریس K به دست می‌آید که برحسب مجموعه‌یی از بردارهای پارامتری بیان می‌شود.[۷ - ۹] از سوی دیگر، نتیجه‌ی تعیین همزمان مقادیر ویژه و بردارهای ویژه در حوزه‌ی زمان، معادلات سیلوستر دوگانه است که از حل آنها ماتریس بهره‌ی دلخواه به دست می‌آید.[۱۳-۱۴]

مسئله‌ی قطب‌گماری با بازخور خروجی را می‌توان از دو جنبه مورد مطالعه قرار داد:

۱) امکان قطب‌گماری با بازخور خروجی برای هر سیستم؛

۲) چگونگی به دست آوردن ماتریس بازخور خروجی برای قطب‌گماری دلخواه.

در این نوشتار، جنبه‌ی دوّم مسئله‌ی قطب‌گماری مورد مطالعه قرار گرفته است. یعنی با فرض وجود جواب برای مسئله، الگوریتم جدیدی طراحی شده است تا قطب‌گماری با بازخور خروجی را انجام

دهد. در روش پیشنهادی، برای جستجوی ماتریس بهره‌ی مناسب، از الگوریتم ژنتیکی استفاده می‌شود و سپس ماتریس‌های به دست آمده با الگوریتم تکرارگرای دیگری تنظیم می‌شود. در این روش هیچ پیش‌فرضی درباره‌ی توزیع قطب‌های سیستم حلقه‌بسته وجود ندارد. در حالت کلی، شرط $m \times l \geq n$ نیز تضمین می‌کند که آزادی کافی در حل این مسئله وجود داشته باشد. اگرچه روش پیشنهادی می‌تواند مسئله را در حالت $m \times l < n$ نیز حل کند، به‌شرط آن که پاسخی برای مسئله وجود داشته باشد.

الگوریتم تکرارگرا

در این بخش، به شرح مختصر الگوریتم تکرارگرایی که از منبع شماره‌ی ۲۰ استخراج شده است می‌پردازیم. علت استفاده از این الگوریتم در روش پیشنهادی، نزدیک کردن ماتریس‌های بهره‌ی موجود است به ماتریس بهره‌ی مورد نظر که قطب‌گماری دلخواه را انجام می‌دهد. ماتریس‌های بهره به‌عنوان عضوهای جمعیت (کروموزوم‌ها) در الگوریتم ژنتیکی کد شده است. همچنین در این بخش، علّت اصلی ناکارایی الگوریتم تکرارگرا در یافتن ماتریس بازخور خروجی شرح داده می‌شود.

الف) الگوریتم تکرارگرای مقدماتی

سیستم توصیف شده با رابطه‌ی ۱ را در نظر بگیرید. اگر بازخور خطی $u = K y$ به کار گرفته شود، معادله‌ی مشخصه‌ی سیستم حلقه‌بسته به‌صورت زیر نتیجه می‌شود:

$$P_{closed}(s)=det[sI-A-BKC] \qquad (۲)$$

حال از دو روش مشابه ــ روش ستونی و روش سطری ــ می‌توان به‌عنوان راه حل تکرارگرا بهره جست. در روش ستونی، ماتریس بازخور خروجی K را برحسب ستون‌هایش بسط می‌دهیم:

$$K = KI_{l\times l} = \sum_{i=1}^{l} k_i e_i^T = \sum_{i=1}^{l} k_i e_i^T + K_j e_j^T = K_j + k_j e_j^T \quad (i \neq j) \qquad (۳)$$

که در آن e_i^T، iامین سطر ماتریس واحد $(l \times l)$ و K_j همان ماتریس K است که در ستون jام آن صفر قرار می‌گیرد. بنابراین معادله‌ی مشخصه‌ی سیستم حلقه‌بسته به‌صورت زیر درمی‌آید:

$$\begin{aligned} P_{closed}(s) &= det[sI - A - BK_j C - BK_j e_j^T C] \\ &= det[sI - A - BK_j C].\ det[sI - A - BK_j C]^{-1} BK_j e_j^T C] \\ &= det[sI - A - BK_j C] - e_j^T C\, adj(sI - A - BK_j C)\, BK_j \\ &= p_j(s) - N_j(s)\, K_j \end{aligned} \qquad (۴)$$

معادله‌ی فوق برای یک K_j مشخص، برحسب K_j خطی است. بنابراین درصورت معلوم بودن تمام ستون‌ها، غیر از ستون jام، می‌توان ستون jام را محاسبه کرد. از رابطه‌ی فوق می‌توان به‌عنوان مبنایی برای ساختن الگوریتم تکرارگرا به‌منظور حل مسئله‌ی قطب‌گماری با بازخور خروجی استفاده کرد. برای حل رابطه‌ی ۴، برای هر j، Λ به l زیرمجموعه تقسیم می‌شود:

$$\begin{aligned} \Lambda &= \{\lambda_{1j}\} \cup \{\lambda_{2j}\} \cup \ldots \cup \{\lambda_{lj}\} \\ &= \Lambda_1 \cup \Lambda_2 \ldots \cup \Lambda_l \end{aligned} \qquad (۵)$$

به‌نحوی که Λ_j، برای به‌دست آوردن K_j استفاده می‌شود. الگوریتم تکرارگرای زیر برای قطب‌گماری با بازخور خروجی، برمبنای نکات فوق شکل می‌گیرد.

الگوریتم تکرارگرا:

گام ۱: $K = K^*$

گام ۲: به‌ازای $j = 1, 2, \ldots, l$، مجموعه معادلات خطی زیر را حل کنید:

$$\begin{bmatrix} N_j(\lambda_{1j}) \\ N_j(\lambda_{2j}) \\ \vdots \\ N_j(\lambda_{rj}) \end{bmatrix} K_j = \begin{bmatrix} P_j(\lambda_{1j}) - P_{closed}(\lambda_{1j}) \\ P_j(\lambda_{2j}) - P_{closed}(\lambda_{2j}) \\ \vdots \\ P_j(\lambda_{rj}) - P_{closed}(\lambda_{rj}) \end{bmatrix} \qquad (۶)$$

که در آن $\lambda_{ij} \in \Lambda_j$. برای هر K_j به‌دست آمده، مقادیر قبلی را با مقادیر جدید جایگزین کنید.

گام ۳: مقادیر ویژه‌ی $\lambda_{i\text{-}closed}$ ماتریس سیستم حلقه‌بسته $A+BKC$ را محاسبه کنید. اگر

$$\sum_{i=1}^{n} (\lambda_i - \lambda_{i\text{-}closed}) < \varepsilon$$

که $\varepsilon > 0$ تولرانس مشخص است، تکرار را متوقف کنید. در غیر این صورت به گام ۲ برگردید.

نکته‌ی ۱: در معادله‌ی ۶، $P_{closed}(\lambda_{ij})$ برابر صفر است، زیرا $P_{closed}(s)$ معادله‌ی مشخصه سیستم حلقه‌بسته‌یی را که باید در قطب‌های مورد نظر برابر صفر شود، تعیین می‌کند. بنابراین معادله‌ی ۶ را می‌توان به‌شکل زیر نوشت:

$$\begin{bmatrix} N_j(\lambda_{1j}) \\ N_j(\lambda_{2j}) \\ \vdots \\ N_j(\lambda_{rj}) \end{bmatrix} K_j = \begin{bmatrix} P_j(\lambda_{1j}) \\ P_j(\lambda_{2j}) \\ \vdots \\ P_j(\lambda_{rj}) \end{bmatrix} \qquad (۷)$$

نکته‌ی ۲: الگوریتم سطری نیز به‌همین ترتیب و با در نظر گرفتن

بسط ماتریس K نسبت به سطرهایش ساخته می‌شود. مزیت ایجاد الگوریتم‌های سطری و ستونی جداگانه این است که در بعضی مسائل، فقط الگوریتم ستونی به یافتن پاسخ مورد نظر کمک می‌کند و در مسائل دیگر، الگوریتم سطری تنها راه دستیابی به جواب است.

نکته‌ی ۳: مجموعه معادلات ۶ مشروط بر آن که $m\times l\geq n$، حداقل یک جواب دارد. این شرط به‌این معناست که تعداد معادلات از تعداد مجهولات بیشتر نباشد. درحالت خاص که $m\times l<n$، مجموعه معادلات ۶ ممکن است دارای جواب باشد یا نباشد.

نکته‌ی ۴: برای اجرای الگوریتم، مجموعه‌ی Λ باید به زیرمجموعه‌های مجزا (Λ_i) تقسیم شود. درصورتی‌که $m\times l>n$ باشد ولی m (یا l) n را بخش نکند، Λ_iها ممکن است تعداد عضوهای متفاوتی داشته باشند. در این حالت، از آنجا که تعداد معادلات از تعداد مجهولات کمتر است، از لحاظ محاسباتی بهتر است یک راه حل مشخص را با ثابت فرض‌کردن برخی درایه‌های K در طول تکرارها آزمایش کنیم.[۲۰]

ب) کاستی‌های الگوریتم تکرارگرا

الگوریتم تکرارگرای فوق مستقیماً نمی‌تواند برای به دست آوردن ماتریس بهره با بازخور خروجی به کار رود. عواملی که بر همگرایی الگوریتم تکرارگرا تأثیر می‌گذارند عبارتند از:

۱. مقادیر اوّلیه‌ی درایه‌های ماتریس بازخور خروجی (K)؛

۲. توزیع قطب‌های دلخواه در m (یا l) زیرمجموعه؛

۳. نوع الگوریتم مورد استفاده (سطری یا ستونی)؛

۴. تعیین درایه‌های ثابت ماتریس بهره و مقادیر ثابت آنها، در حالتی که $m\times l>n$ است ولی m (یا l) n را بخش نکند.

از آنجا که همگرایی الگوریتم وابستگی زیادی به هر یک از عوامل فوق دارد، نامشخص‌بودن هریک از عوامل فوق پیش از اجرای الگوریتم، مانع از همگرایی مطلوب آن می‌شود.

در حالتی که مجموعه‌ی قطب‌های دلخواه سیستم حلقه‌بسته شامل مقادیر تکراری باشد، اگر جای‌دادن قطب‌های تکراری در زیرمجموعه‌های متفاوت امکانپذیر باشد، الگوریتم همگرا می‌شود. در غیر اینصورت، معادلات یکسانی برای مقادیر تکراری بوجود می‌آید که خاصیت یکسان‌بودن قطب‌ها را در مجموعه معادلات منعکس نمی‌کند. علاوه بر آن که شرط توزیع اجباری قطب‌های تکراری در زیرمجموعه‌های متفاوت، همگرایی الگوریتم را تحت تأثیر قرار می‌دهد، قطب‌هایی که تکرار آنها بزرگ‌تر از m یا l باشد با این الگوریتم به دست نمی‌آیند.

الگوریتم ژنتیکی پیشنهادی

الگوریتم ژنتیکی راهی مناسب برای یافتن نقاط بهینه‌ی سراسری است، بویژه هنگامی که مسئله غیرخطی است و روش‌های متداول بهینه‌سازی که مبتنی بر مشتق‌گیری هستند، ناتوان از حل مسائل پیچیده‌اند. هر نسل در الگوریتم ژنتیکی، از کروموزوم‌هایی تشکیل می‌شود که معرف شکل کدشده‌ی پاسخ‌های ممکن برای مسئله‌اند. تابع برازندگی که توسط طراح الگوریتم انتخاب می‌شود، تعیین می‌کند که هر کروموزوم تا چه میزان به بهترین پاسخ نزدیک است. این تابع مقدار برازندگی را برای هر کروموزم مشخص می‌کند. برمبنای این مقادیر شایستگی، بهترین کروموزوم‌ها به‌عنوان والدین نسل بعد انتخاب می‌شوند. والدین به‌صورت اتفاقی با هم ترکیب شده و سپس جهش داده می‌شوند تا جمعیت فرزندان را به‌وجود آورند. با تکرار این مراحل، کروموزوم‌ها به سمت بهترین جواب تغییر می‌کنند تا هنگامی که با برآورده شدن معیار توقف، اجرای الگوریتم پایان یابد.[۲۱-۲۳]

الگوریتم پیشنهادی مرحله‌ی دیگری دارد که متشکل از الگوریتم تکرارگرای ارائه شده در بخش قبل است. در هر تکرار الگوریتم ژنتیکی، بعد از تشکیل جمعیت جدید، الگوریتم تکرارگرا برای هر کروموزوم موجود در جمعیت اجرا می‌شود. گزینش، ترکیب و جهش به جای آن که بر کروموزوم‌های عضو جمعیت مستقیماً اثر کنند بر کروموزوم‌های به دست آمده از الگوریتم تکرارگرا عمل می‌کنند.

الف) کدکردن و تولید جمعیت اوّلیه

پیش از این به‌اختصار، عوامل مؤثر بر همگرایی الگوریتم تکرارگرا را تشریح کردیم. از آنجا که الگوریتم پیشنهادی از الگوریتم تکرارگرا استفاده می‌کند، باید شرایط اوّلیه‌ی گوناگونی برای اجرای آن فراهم کند. این شرایط اوّلیه به‌عنوان کروموزوم‌های جمعیت در الگوریتم ژنتیکی کد می‌شوند. برای پوشاندن تمامی شرایط اوّلیه، هر کروموزوم به‌نحوی کد می‌شود که شامل قسمت‌های زیر باشد:

۱. قطب‌های مطلوب سیستم حلقه‌بسته که چیدن آنها با ترتیب اتفاقی باعث توزیع متفاوت این مقادیر در زیرمجموعه‌های الگوریتم تکرارگرا می‌شود.

۲. علامت درایه‌های ماتریس بازخور خروجی که مجموعاً $m\times l$ بیت با مقدار ۱ یا ۱- هستند.

۳. قسمت صحیح درایه‌های ماتریس بازخور خروجی، که از سه رقم —بین صفر تا نه— تشکیل می‌شود، به این معنا که اندازه‌ی هر درایه به هزار محدود است.

۴. قسمت اعشاری درایه‌های ماتریس بازخور خروجی که از $m\times l$ مقدار بین صفر تا یک تشکیل می‌شود.

| | | |

نداشته باشد. فرض کنید λ_{ij} و λ_{i+1j} دو قطب مزدوج مختلط در زیر مجموعه‌ی jام باشند. آنگاه:

$$e_j^T C(\lambda_{ij} I - A - BK_j C)^{-1} BK_j = 1$$
$$e_j^T C(\lambda_{i+1j} I - A - BK_j C)^{-1} BK_j = 1 \qquad (13)$$

از آنجا که $(\lambda_{ij} I - A - BK_j C)^{-1}$ و $(\lambda_{ij+1} I - A - BK_j C)^{-1}$ نیز مزدوج مختلط‌اند، به‌آسانی می‌توان نشان داد که:

$$e_j^T C\, Re\{(\lambda_{ij} I - A - BK_j C)^{-1} BK_j\} = 1$$
$$e_j^T C\, Imag\{(\lambda_{ij} I - A - BK_j C)^{-1} BK_j\} = 0 \qquad (14)$$

یکی از معادلات فوق برای اولین قطب مختلط و دیگری برای مزدوج آن به کار خواهد رفت.

در پایانِ الگوریتم تکرارگرا، اگر ماتریس بهره‌ی به دست آمده محدود بماند (همگرایی)، کروموزوم مشابهی به الگوریتم اصلی بازگردانده می‌شود، به غیر از قسمت ماتریس بهره که با ماتریس به دست آمده از تکرارها جایگزین می‌شود. در غیر این‌صورت، قطب‌ها در قسمت اول این کروموزوم به‌صورت اتفاقی جابه‌جا می‌شوند و بیت مشخصه‌ی الگوریتم نیز تغییر می‌کند. افزون بر آن، قسمت بهره نیز جایگزین آخرین ماتریس بهره‌ی به دست آمده قبل از واگرایی الگوریتم می‌شود. همگرایی الگوریتم تکرارگرا با آزمایش وضعیت ماتریس $(\lambda_{ij} I - A - BK_j C)$ مشخص می‌شود. وقتی درایه‌های K به سمت بی‌نهایت میل می‌کنند، $\lambda_{ij} I - A - BK_j C$ منفرد می‌شود.

ج) مراحل ژنتیکی الگوریتم پیشنهادی

برای یافتن ماتریس بهره‌یی که قطب‌های سیستم حلقه‌بسته را در مکان‌های مشخص جای دهد، الگوریتم ژنتیکی هر یک از کروموزوم‌ها را طی نسل‌های متمادی تکامل می‌دهد. عملگرهای ژنتیکی جمعیت فرزندان را از جمعیت والدین تولید می‌کند تا خصوصیات جدیدی در جمعیت به وجود آید، اگرچه فرزندان همواره بخشی از خصوصیات والدین را به ارث می‌برند. ترکیب خصوصیات جدید و قدیم، همراه با تنظیم ماتریس‌های بهره از طریق الگوریتم تکرارگرا، جمعیت را به سمت نقطه‌ی بهینه سوق می‌دهد.

گزینش و تابع برازندگی

الگوریتم ژنتیکی بهترین کروموزوم‌های هر نسل را به‌عنوان والدین نسل بعد انتخاب می‌کند. برای گزینش بهترین کروموزوم‌ها سازوکار چرخ رولت به کار گرفته می‌شود. همان طور که برای ارزیابی هر کروموزوم معیار برازندگی باید تعریف شود. مقدار برازندگی متناظر با هر کروموزوم نشان می‌دهد که چه مقدار قطب‌های سیستم حلقه‌بسته به مقادیر مورد نظر نزدیک‌اند. واضح است که یکی از آسان‌ترین روش‌ها برای تعریف تابع برازندگی، در نظر گرفتن مجموع مربعات فاصله‌ی بین قطب‌های مطلوب و قطب‌های سیستم حلقه‌بسته است. مقدار برازندگی هر کروموزوم می‌تواند برابر با معکوس این فاصله تعریف شود. اما آزمایش بر روی سیستم‌های مختلف نشان می‌دهد که با این معیار، الگوریتم در دام نقاط بهینه‌ی محلی گرفتار می‌شود. اگر M_i را iامین کروموزوم جمعیت و K_i را ماتریس بهره‌ی متناظر با آن فرض کنیم، آنگاه:

$$S = (s_{i1}, s_{i2}, \dots, s_{in}) = eigen(A + B\,K_i C)$$

حال قطب‌های دلخواه سیستم حلقه‌بسته را به‌نحوی مرتب می‌کنیم که هر قطب در مکان نزدیک‌ترین قطب در بردار S قرار گیرد:

$$\Lambda_{new} = (\lambda_1, \lambda_2, \dots, \lambda_n)$$

درصد فاصله‌ی نسبی ماتریس بهره (K_i) نیز به‌صورت زیر تعریف می‌شود:

$$\text{Rel-distance } (K_i) = 100 \times \text{absolute } \{(\lambda_1 - s_{i1}) / \lambda_1, (\lambda_2 - s_{i2}) / \lambda_2, \dots, (\lambda_n - s_{in}) / \lambda_n\} \qquad (15)$$

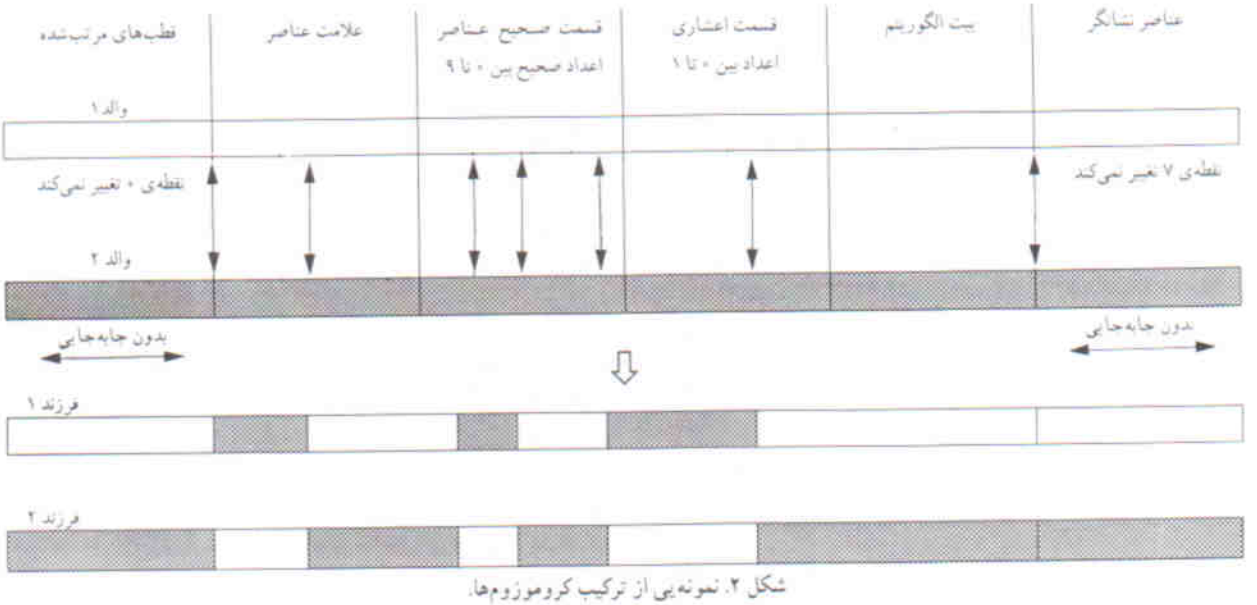
اگر λ_i صفر باشد، $(\lambda_i - s_i)$ جایگزین $(\lambda_i - s_i)/\lambda_i$ می‌شود. در نهایت، مقدار برازندگی کروموزوم M_i طبق رابطه‌ی زیر تعریف می‌شود:

$$Fitness\ (M_i) = (Max\ \{Rel\text{-}distance\ (K_i)\})^{-2} \qquad (16)$$

این تعریف از تابع برازندگی، الگوریتم را قادر می‌سازد که قطب‌های سیستم حلقه‌بسته را نزدیک به مقادیر کوچک همانند مقادیر بزرگ قرار دهد. تعداد کروموزوم‌های انتخاب شده برای مرحله‌ی ترکیب در تمامی نسل‌ها ۴۰ در نظر گرفته می‌شود (با فرض این که ۵۰ کروموزوم در جمعیت وجود دارد).

تقاطع (ترکیب)

عملگر تقاطع در الگوریتم ژنتیکی برای تولید فرزندان از والدین به کار می‌رود. از آنجا که هر کروموزوم در الگوریتم طراحی شده از اعدادی در محدوده‌های گوناگون تشکیل شده است، عملگر تقاطع بر هر قسمت جداگانه اثر می‌کند. برای دستیابی به بهترین نتیجه، جابه‌جایی ۵ نقطه‌یی در این الگوریتم به کار گرفته می‌شود؛ یک نقطه در قسمت علامت، سه نقطه در بخش صحیح درایه‌ها و یک نقطه در بخش اعشاری و بیت مشخصه‌ی الگوریتم. هر جفت از کروموزوم‌هایی که برای تقاطع انتخاب می‌شوند با احتمال مشخص



شکل ۲. نمونه‌یی از ترکیب کروموزوم‌ها.

جدول ۱. قوانین جهش در الگوریتم پیشنهادی

	وضعیت تولید عدد تصادفی	تأثیر جهش بر عناصر انتخاب شده
قسمت علامت	برای هر مقدار یک عدد تصادفی تولید می‌شود	ضرب در ۱-
قسمت صحیح	برای هر مقدار یک عدد تصادفی تولید می‌شود	بیت تصادفی جدید جایگزین می‌شود
قسمت اعشاری	برای هر مقدار یک عدد تصادفی تولید می‌شود	اضافه کردن یک عدد تصادفی
بیت الگوریتم	برای کلیه‌ی مقادیر یک عدد تصادفی تولید می‌شود	۱ به ۰ تغییر می‌کند و بالعکس
نشانگر	برای کلیه‌ی مقادیر یک عدد تصادفی تولید می‌شود	مجموعه‌ی جدیدی از عناصر نشانگر جایگزین نشانگرهای قبلی می‌شوند

در نقاط انتخاب شده شکسته شده و جابجایی بین نقاط زوج و فرد صورت می‌گیرد. درنتیجه دو کروموزوم فرزند به جمعیت والدین اضافه می‌شوند. شکل ۲ چگونگی انجام این کار را نشان می‌دهد. در این الگوریتم، احتمال جابه‌جایی ۰/۸ است که به این معنا است که ۰/۸×۴۰×۲=۶۴ کروموزوم در جمعیت وجود خواهد داشت.

جهش

در این بخش، قوانین کلی جهش شرح داده خواهد شد. جهش همانندتقاطع قسمت به قسمت انجام می‌شود. برای قسمت اوّل هر کروموزوم، عددی اتفاقی تولید می‌شود و اگر این عدد از احتمال جهش P_m کوچکتر باشد، قطب‌های دلخواه به‌صورت جدید چیده می‌شوند. قوانین جهش برای قسمت‌های بعد طبق جدول ۱ است.

چیدن قطب‌های مطلوب می‌تواند در دستیابی به بهترین توزیع آنها در زیرمجموعه‌های سطری یا ستونی در الگوریتم تکرارگرا کمک کند. جهش درایه‌های ماتریس بهره، جستجوی سریع ماتریس بهره‌ی مطلوب را به‌همراه دارد. تغییر مداوم بیت مشخصه‌ی الگوریتم، الگوریتم ژنتیکی را از افتادن در دام نقاط بهینه‌ی محلی می‌رهاند.

تمامی روش‌های فوق برای جهش به این منظور به کار گرفته می‌شوند که خواص جدیدی را در جمعیت بوجود آورند تا در کوتاه‌ترین زمان ممکن پاسخ مطلوب به دست آید.

ساختار کلی الگوریتم پیشنهادی

تاکنون تمامی مراحل الگوریتم پیشنهادی را شرح داده‌ایم. حال با دانستن این مراحل نشان خواهیم داد که چگونه این الگوریتم، ماتریس بازخور خروجی مناسب برای قطب‌گماری دلخواه را می‌یابد.

ماتریس بهره‌ی مورد نظر طی ۴ مرحله محاسبه می‌شود که هر یک از این مراحل از ساختار ژنتیکی برخوردار است. در مرحله‌ی اوّل، پس از تولید جمعیت اوّلیه به‌صورت اتفاقی، کروموزوم‌ها توسط الگوریتم تکرارگرا اندکی کامل می‌شوند. در این مرحله جهش به قسمت اعشاری درایه‌های ماتریس بهره اثر نمی‌کند. نرخ جهش ۸٪ است مگر در قسمت اشاره‌گرها که نرخ جهش در آن به ۲۵٪ افزایش می‌یابد. در پایانِ هر یک از تکرارهای الگوریتم، پنج والد برتر نیز به همراه فرزندان به نسل بعد راه می‌یابند.

در پایانِ هر تکرار الگوریتم ژنتیکی، مجموع مربعات خطای ناشی از هر کروموزوم محاسبه می‌شود. در صورتی که این مجموع از یک کمتر باشد، الگوریتم به مرحله‌ی بعدی می‌رود. کروموزوم انتخاب شده «کروموزوم مرجع» نامیده می‌شود.

در آغاز مرحله‌ی بعد، برای تولید جمعیت اوّلیه، کروموزوم مرجع ۵۰ بار تکرار می‌شود. آنگاه قسمت قطب‌های دلخواه در بخش اوّل کروموزوم‌ها، با ترتیب جدیدی چیده می‌شود و بیت مشخصه‌ی الگوریتم نیز تغییر می‌کند. تمامی کروموزوم‌ها در جمعیت جدید، مجموع مربعات خطای یکسانی خواهند داشت، اگرچه به‌علت تغییرات ایجاد شده، اجرای الگوریتم تکرارگرا در مورد هر یک از آنها نتایج مختلفی به همراه خواهد داشت.

مراحل ۲، ۳ و ۴ ساختاری مشابه ساختار مرحله‌ی ۱ دارند. در این مراحل، جهش بر قسمت اعشاری درایه‌های ماتریس بهره اثر می‌کند. هر قسمت اعشاری که برای جهش انتخاب شود با عددی اتفاقی بین ۰/۱ و ۰/۱- جمع می‌شود. الگوریتم مراحل ۲، ۳ و ۴ با یافتن کروموزومی که مجموع مربعات خطای آن کمتر از $۱۰^{-۲}$، $۱۰^{-۴}$ و $۱۰^{-۶}$ باشد پایان می‌پذیرد.

مثال‌های عددی

مثال‌هایی که در این بخش انتخاب شده‌اند سیستم‌هایی را معرفی می‌کنند که الگوریتم‌های دیگر ناتوان از یافتن ماتریس بازخور خروجی مناسب برای قطب‌گماری دلخواه بوده‌اند. این مثال‌ها و مثال‌های متعدد دیگر نشان می‌دهد که هیچ‌گونه محدودیتی برای مجموعه قطب‌های مطلوب سیستم حلقه‌بسته وجود ندارد و این مجموعه می‌تواند شامل مقادیر یکسان و یا قطب‌های سیستم حلقه‌باز باشد.

مثال ۱. سیستم توصیف شده با تحقق زیر را در نظر بگیرید:

$$A=\begin{bmatrix} ۰ & ۱ & ۰ & ۰ \\ -۱ & ۰ & ۲ & ۰ \\ ۰ & ۰ & ۰ & ۱ \\ ۰ & ۱ & ۰ & -۳ \end{bmatrix} \quad B=\begin{bmatrix} ۰ & -۱ \\ ۱ & ۰ \\ ۰ & ۳ \\ ۰ & ۱ \end{bmatrix}$$

$$C=\begin{bmatrix} ۱ & -۱ & ۰ & ۰ \\ ۰ & ۰ & ۱ & ۲ \end{bmatrix}$$

مرجع شماره ۹، این مثال را با قراردادن ۲ قطب در ۲- و ۱- حل، و سپس بهترین مکان ممکن برای ۲ قطب دیگر را جستجو کرده است. یعنی الگوریتم ۹ نتوانسته به‌طور همزمان ۴ قطب را تعیین کند.

با استفاده از الگوریتم طراحی شده، ۲ قطب دیگر نیز در ۱- قرار داده می‌شوند تا مسئله در حالت دشوار ــ قراردادن سه قطب در ۱- به‌طور همزمان ــ حل شود. اجرای الگوریتم پس از ۴۸ تکرار الگوریتم ژنتیکی، ماتریس زیر را نتیجه می‌دهد:

$$K=\begin{bmatrix} ۰/۹۶۶۲۱۳۸۹۴۴۸۸ & -۲/۴۸۵۶۸۰۶۷۶۶۱۹ \\ ۰/۳۶۵۱۶۳۲۲۵۹۹ & -۰/۱۳۳۷۸۶۰۸۴۰۳۴ \end{bmatrix}$$

چنان که مشاهده می‌شود این ماتریس، قطب‌های سیستم حلقه‌بسته را در ۰/۹۹۹۹۲۳۳۶۷۱۱۵۳۶-، ۱/۰۰۰۱۰۸۳۷۵۴۰۸۴۷-، ۱/۰۰۰۲۶۱۸۶۶۰۱۴۷۰- و ۲/۰۰۰۱۳۹۴۶۶۲۴۴۷- قرار می‌دهد.

مثال ۲. در این مثال، مسئله‌یی را بررسی می‌کنیم که در آن یکی از قطب‌های سیستم حلقه‌بسته باید منطبق بر یکی از قطب‌های سیستم حلقه‌باز باشد. برای این منظور، سیستم توصیف شده با تحقق زیر را در نظر بگیرید:

$$A=\begin{bmatrix} ۱ & ۰ & ۰ & ۰ \\ ۰ & ۲ & ۰ & ۰ \\ ۰ & ۰ & -۳ & ۰ \\ ۰ & ۰ & ۰ & -۴ \end{bmatrix} \quad B=\begin{bmatrix} ۱ & ۰ \\ ۰ & ۱ \\ ۱ & ۰ \\ ۱ & ۱ \end{bmatrix}$$

$$C=\begin{bmatrix} ۱ & ۱ & ۰ & ۰ \\ ۰ & ۰ & ۱ & ۱ \end{bmatrix}$$

مجموعه قطب‌های مطلوب سیستم حلقه‌بسته عبارتند از: ۱-، ۲-، ۳- و ۵-. بعد از ۸ تکرار ماتریس زیر به دست می‌آید:

$$K=\begin{bmatrix} ۸/۴ & ۱/۲ \\ -۱۶/۲ & -۱/۶ \end{bmatrix}$$

قطب‌های سیستم حلقه‌بسته با به کارگیری این ماتریس، در نقاط ۱-، ۲-، ۳- و ۵- قرار می‌گیرند.

نتیجه گیری

دراین نوشتار، روش جدیدی برای حل مسئله‌ی قطب‌گماری با بازخور خروجی ارائه شده است. الگوریتم طراحی‌شده محدودیتی بر مجموعه‌قطب‌های مورد نظر اعمال نمی‌کند و این مجموعه می‌تواند شامل مقادیر یکسان، یا قطب‌های سیستم حلقه‌باز باشد.

قطب‌گماری با بازخور خروجی، حل دستگاهی از معادلات غیرخطی است. در الگوریتم طراحی شده، از طریق تبدیل این مسئله به چند مجموعه از معادلات خطی، جواب جستجو می‌شود. بخش ژنتیکی الگوریتم، بهترین ماتریس‌های بهره را از الگوریتم دیگری که الگوریتم تکرارگرا نامیده می‌شود انتخاب و نیز شرایط اوّلیه‌ی لازم را برای الگوریتم تکرارگرا فراهم می‌کند. آزمایش الگوریتم بر روی سیستم‌های مختلف، کارایی آن را به اثبات رسانده است.

منابع

1. Davison, E.J., "On pole assignment in linear systems with incomplete state feedback," *IEEE Trans. Automatic Control*, pp. 348-351 (June 1970).
2. Davison, E.J., Chatterjee, "A note on pole assignment in linear systems with incomplete state feedback," *IEEE Trans. Automatic Control*, pp. 98-99 (February 1971).
3. Sridhar, B. and Lindorff, D.P., "Pole placement with constant gain output feedback," *INT. J. Control*, **18**(5), pp. 993-1003 (1973).
4. Kimura, H. "Pole assignment by gain output feedback," *IEEE Trans. Automatic Control*, pp. 516-519 (August 1975).
5. Davison, E.J. and Wang, S.H., "On pole assignment in linear multivariable systems using output feedback," *IEEE Trans. Automatic Control*, pp. 509-515 (August 1975).
6. Kimura, H., "A further result on the problem of pole assignment by output feedback," *IEEE Trans. Automatic Control*, pp. 458-463 (June 1977).
7. Fahmu, M.M. and O'Reily, J., "Multistage parametric eigenstructure assignment by output feedback control," *INT. J. Control*, **48**(1), pp. 97-116 (1988).
8. Fahmy, M.M. and O'Reily, J., "Parametric eigenstructure assignment by output feedback control; the case of multiple eigenvalues," *INT. J. Control*, **48**(4), pp. 1519-1535 (1988).
9. Roppenecker, G. and O'Reilly, J., "Parametric output feedback controller design," *Automatica*, **25**(2), pp. 259-265 (1989).
10. Seraji, H., "A new method for pole placement using output feedback," *INT. J. Control*, **28**(1), pp. 147-155 (1979).
11. Calvo-Ramon, J.R.,"Eigenstructure assignment by output feedback and residue analysis," *IEEE Trans. Automatic Control*, **AC-31**(3), pp. 247-249 (March 1986).
12. Kwon, B.H. and Youn, M.J., "Eigenvalue-Generalized eigenvector assignment by output feedback," *IEEE Trans. Automatic Control*, **AC-32**(5), pp. 417-421 (March 1987).
13. Duan, G.R., "Solutions of the equation AV+BW=VF and their application to eigenstructure assignment in linear systems," *IEEE Trans. Automatic Control*, **38**(2), pp. 276-280 (February 1993).
14. Syrmos, V.L. and Lewis, L., "Output feedback eigenstructure assignmet using two sylvester equations," *IEEE Trans. Automatic Control*, **38**(3), pp. 495-499 (March 1993).
15. Alexandridis, A.T. and Paaraskevopoulos, P.N.,"A new approach to eigenstructure by output feedback," *IEEE Trans. Automatic Control*, **41**(7), pp. 1046-1050 (July 1996).
16. Byrnes, C.I., Anderson, B.D., "Output feedback and generic stabilizability," *SIAM J. Contr. Optim*, **22**, pp. 362-380 (May 1984).
17. Tarokh, M. and "Approach to pole assignment by centralised and decentralised output feedback," *IEE Proc., D*, **136**, (March 1989).
18. Herman, R. and Martin, C.I.,"Application of algebric geometry to systems theory: Part I," *IEEE Trans. Automatic Control*, **AC-22**, pp. 19-25 (1977).
19. Giannakopoulos, C. and Karcanias, N.,"Pole assignment of strictly proper and proper systems by constant output feedback," *INT. J. Control*, pp. 543-565 (1985).
20. Lee, T.H., Wang O.G. and Koh, E.K.,"An iterative algorithm for pole placement by output feedback," *IEEE Trans. Automatic Control*, **39**(3), pp. 565-568 (March 1994).
21. Goldberg, D.E., "Genetic algorithms and rule learning in dynamic system control," *in Proc. 1st Int. Conf. on Genetic Algorithms and Their Applications*, Hillsdale, NJ: Lawrence Erlbaum, pp. 8-15 (1985).
22. Goldberg, D.E., *Genetic algorithms in search, optimization and machine learning, Reading*, MA: Addison-Wesley, (1989).
23. Davis, L., Ed., *Handbook of Genetic Algorithms*, New York: Van Nostrand Reinhold, (1991).