مجله مهندسی مکانیک، شماره پیاپی ۹۶، جلد ۵۱، شماره ۳، پاییز ۱۴۰۰، صفحه ۲۳۹-۲۴۹

بهبود عملکرد الگوریتم رقابت استعماری و کاربرد آن در کنترل پرواز بالگرد

امیرعلی نیکخواه * دانشیار، دانشکده مهندسی هوافضا، دانشگاه صنعتی خواجه نصیرالدین طوسی، تهران، ایران

فرهاد پاکرو دانشجوی دکتری، دانشکده مهندسی هوافضا، دانشگاه صنعتی خواجه نصیرالدین طوسی، تهران، ایران

چکیده

در این مقاله هدف بهبود عملکرد الگوریتم رقابت استعماری است. روش مورد استفاده افزودن جستجوی محلی به این الگوریتم است که این کار در دو مرحله صورت گرفته‌است. در مرحله‌ی اول انتخاب پارامترهای تحت بهینه‌سازی شروع مساله به نوعی هوشمندانه انجام می‌شود و جمعیت به سمت نقطه‌ی بهینه هدایت می‌شوند. در مرحله‌ی دوم پس از هر ۱۰ تکرار مساله، یک جستجوی محلی در میان مستعمره‌ها و استعمارگر چند فرمانروایی قدرتمند انجام می‌شود تا ضمن عدم ایجاد تغییر محسوس در فرایند رسیدن به جواب، پاسخ دقیق‌تری حاصل شود. این روش بر روی تمامی فرمانروایی‌ها اعمال نمی‌شود تا حجم محاسبات بیهوده افزایش نیابد. الگوریتم ارائه شده با توابع محک سنجیده شده‌است و طبق مقایسه‌ی انجام‌شده میان عملکرد آن‌ها، عملکرد الگوریتم اصلی در هنگام افزایش تعداد ابعاد مساله (پارامترهای مجهول بهینه‌سازی) بهبود یافته‌است. با تعداد پارامترهای مجهول کمتر، هم زمان همگرایی و هم پاسخ کمینه‌ی نهایی در دو روش نزدیک به هم هستند، اما با افزایش تعداد ابعاد مساله، حتی در پاره‌ای از اوقات زمان حل به بیش از نصف کاهش یافته و میزان کمینه تا بیش از ۱۰ مرتبه‌ی اعشار بهبود یافته‌است. در انتها از کاربرد این روش برای بهینه‌سازی در یافتن بهره‌های کنترلی مناسب برای کنترل یک بالگرد استفاده شده‌است که خود کاربردی تازه به‌شمار می‌آید و عملکرد این روش با جایابی قطب و روش کنترل پسخور حالت سنجیده و مقایسه شده‌است.

واژه‌های کلیدی: الگوریتم رقابت استعماری، جستجوی محلی، الگوریتم ژنتیک، توابع محک، کنترل بالگرد، کنترل پسخور حالت.

Improving the Performance of Imperialist Competitive Algorithm and Its Application in Helicopter Flight Control

A. A. Nikkhah Aerospace Engineering Department, K. N. Toosi University of Technology, Tehran, Iran

F. Pakro Aerospace Engineering Department, K. N. Toosi University of Technology, Tehran, Iran

Abstract

In this paper, the purpose is to improve the performance of the Imperialist Competitive Algorithm. Local search is added to the algorithm and this is done in two stages. In first stage, selection of the initial optimization parameters for starting the problem is done wisely and the population is leaded to optimal point. In second stage, after each 10 iterations a local search is performed between colonies and the imperialist of some of the strongest empires, in order to gain more accurate answers while not making a notable change in the process of reaching the answer. This method is not applied to all empires, so that calculations would not overflow. The proposed algorithm is tested using benchmark functions and based on the comparisons, the performance of the initial algorithm is much more improved when dimensions of the optimization problem goes higher. With few unknown parameters, conversion time and final minimum values are close in both methods, but by increasing the dimensions of the problem, even in some cases total calculation time is reduced to a half and the accuracy of minimum value is improved by 10 orders of magnitude. At the end, one application of the method is investigated in finding the suitable control gains for controlling a helicopter, which is a relatively a new application, and the performance is evaluated comparing to pole-placement and by state-feedback control.

Keywords: Imperialist competitive algorithm, Local search, Genetic algorithm, Benchmark functions, Helicopter control, State feedback control.

۱- مقدمه

در الگوریتم رقابت استعماری[1] به تکامل اجتماعی و تاریخی جوامع به عنوان پیچیده‌ترین و موفق‌ترین حالت تکامل، توجه شده است. این الگوریتم با الهام گرفتن از تکامل اجتماعی انسان، برای بهینه‌سازی، توسعه داده شده است و با مدل‌کردن یک فرایند اجتماعی سیاسی، نسبت به روش‌های موجود فعلی دارای توانایی بالایی بوده و تا حد بسیار زیادی نیز، سریع می‌باشد [۱].

در این الگوریتم با شروع از یک جمعیت اولیه و با درنظر گرفتن قدرت هر عضو از جمعیت، که همان هزینه‌ی هر یک از اعضا است، کم‌کم اعضای ضعیف‌تر از فرمانروایی خود حذف می‌شوند و در نهایت همه‌ی اعضا به قوی‌ترین فرمانروایی تعلق می‌گیرند. در این فرآیند نقطه‌ی بهینه و پاسخ مساله، قوی‌ترین عضو فرمانروایی نهایی است. این همان سیاستی است که کشورهای استعمارگر در طول تاریخ انجام داده‌اند.

آتش‌پز [۲] سه ویژگی جدید به ICA اصلی اضافه کرده‌است: ایجاد انقلاب که در واقع چند مستعمره‌ی موجود را با تعدادی مستعمره‌ی تصادفی به وجود آمده، جایگزین می‌کند؛ اتحاد که یعنی یکی کردن دو حکومت که امپریالیست‌های آن‌ها بسیار نزدیک به هم هست؛ پایین آوردن فرکانس رقابت نیز از نابودی خیلی سریع یک امپراتوری ضعیف،

[1] Imperialist Competitive Algorithm

* نویسنده مکاتبه کننده، آدرس پست الکترونیکی: nikkhah@kntu.ac.ir

تاریخ دریافت: ۹۸/۰۹/۰۲

تاریخ پذیرش: ۹۹/۰۵/۰۶

جلوگیری می‌کند.

چند نمونه از کارهای انجام شده نیز روی فرایند نزدیک شدن مستعمره به امپریالیست تمرکز داشته‌اند. ژنگ و همکاران [۳] این فرایند را دو قسمت کرده‌اند. در ابتدا هر مستعمره دقیقا به سمت امپریالیست خودش حرکت می‌کند و بعد از آن برخی از مستعمره‌ها به‌صورت تصادفی به هر سمتی حرکت می‌کنند تا در حل تنوع ایجاد شود. بهرامی و همکاران [۴] از یک نگاشت آشوب‌ناک برای تعیین جهت حرکت در این فرایند استفاده کرده‌اند.

نیکنام و همکاران [۵] در تحقیق خود برای خوشه بندی مجموعه‌ای از داده‌ها، روش k-means را که به یک بهینه‌ی محلی می‌انجامد، با الگوریتم رقابت استعماری ترکیب کرده‌اند تا نتایج بهتری حاصل شود. در مقایسه با تعدادی از الگوریتم‌های معروف استفاده شده، این روش پاسخ مناسب‌تری را داشته‌است. آن‌ها در ابتدا تعدادی پاسخ با الگوریتم رقابت استعماری تولید می‌کنند، با استفاده از SA پاسخ‌ها را بهبود می‌بخشند و در نهایت آن را با k-means ترکیب می‌کنند. برای ترکیب می‌توان، ابتدا با کمک k-means جمعیتی را تولید کرد، جمعیت تحت رقابت استعماری قرار می‌گیرد و سپس SA روی آن اعمال می‌شود.

صبور و همکاران [۶] در مقاله‌ی خود الگوریتم رقابت استعماری را با الگوریتم لانه‌ی مورچه ترکیب کرده‌اند. این ترکیب نشان داده‌است که سرعت همگرایی بالاتر رفته‌است و این روش بر روی سازه‌های خرپایی اعمال شده‌است. در این الگوریتم، رقابت استعماری بر روی جمعیت اعمال می‌شود و بعد از آن بر روی هر فرمانروایی الگوریتم لانه مورچه‌ای اعمال می‌شود تا به نوعی پاسخ محلی بهتری پیدا شود. پس از آن ادامه‌ی روند الگوریتم رقابت استعماری انجام می‌شود تا مرحله‌ی بعدی حل آغاز شود.

جون لین و همکاران [۷] در مقاله‌ی خود برخلاف بسیاری از کارهای انجام شده، که بر روی فرایند نزدیک شدن به امپریالیست تمرکز دارند، تمرکز اصلی را بر روی فرایند رقابت گذاشته‌اند. در این الگوریتم یک مرحله‌ی تعامل به روش قدیمی اضافه شده‌است که در آن امپریالیست هر کشور می‌تواند به کشور دیگر انتقال یابد و تعویض امپریالیست فقط درون یک کشور خاص صورت نمی‌گیرد. با تغییرات اعمال شده در نحوه‌ی رقابت، عملکرد این الگوریتم از تمامی انواع توسعه داده شده‌ی قبلی آن بهتر شده‌است.

نوذریان و وفایی جهان [۸] در مقاله‌ی خود، از یک روش ممتیک، که در واقع ترکیب الگوریتم ژنتیک و روش‌های ابتکاری برای جستجوی محلی است، استفاده کرده‌اند. در این پژوهش آن‌ها از الگوریتم رقابت استعماری به عنوان جستجوی محلی، جهت حل مساله‌ی کوله پشتی[1] استفاده کرده‌اند. پاسخ این استفاده‌ی ترکیبی از الگوریتم‌ها، بهتر از الگوریتم ژنتیک معمولی بوده‌است؛ درحالیکه، نسبت به استفاده از الگوریتم رقابت استعماری به تنهایی، بهبود چشمگیری را نشان نمی‌دهد.

لین و همکاران [۹] نیز بر روی فرایند نزدیک شدن به امپریالیست تغییراتی ایجاد کرده‌اند. در این مقاله آن‌ها رابطه‌ی نزدیک شدن را تغییر داده‌اند و جملاتی تصادفی در آن ایجاد کرده‌اند تا خاصیت حرکت تصادفی حتی در خلاف جهت امپریالیست را به‌وجود آورند.

جون لین و همکاران [۱۰] در مقاله‌ی خود، با هدف بهبود عملکرد الگوریتم رقابت استعماری، بر روی بهترین پاسخ هر امپراتوری یک الگوریتم جستجوی محلی اعمال کرده‌اند. این شکل بکارگیری الگوریتم، نه به اندازه‌ی اعمال آن به تمامی اعضا، زمان‌بر است و نه به اندازه‌ی اعمال آن به بهترین پاسخ کل جمعیت سریع و غیر کاربردی است. روش پیشنهاد شده، جستجوی خط تصادفی [۱۱] نام دارد که با استفاده از آن به شکل گفته شده، نتایج بهتری نسبت به اعمال آن بر روی بهترین پاسخ کل جمعیت بدست آمده است.

در سال‌های اخیر تحقیقات دیگری نیز در توسعه‌ی این روش بهینه‌سازی انجام شده‌اند که در برخی از آن‌ها تمرکز در گرفتن کاربردی نوین از روش ارائه‌شده بوده‌است و در برخی دیگر تغییر روش و ترکیب آن با دیگر روش‌ها، در الویت قرار داشته‌است. کابردهایی از این روش را می‌توان در پیش‌بینی ارتعاشات زمین [۱۲]، مدیریت انرژی در میکروگریدها [۱۳]، کنترل سرعت توربین باد [۱۴]، سیستم انتقال قدرت [۱۵، ۱۶]، اره ماشینی دوار [۱۷]، مساله‌ی فروشنده دوره‌گرد [۱۸]، پیش‌بینی توان تولیدی در توربین باد [۱۹]، انبار گردانی و زنجیره تامین [۲۰، ۲۱]، سیستم‌های فتوولتایی [۲۲]، مسائل باینری [۲۳]، سیستم‌های نیروگاهی [۲۴]، پیش‌بینی بازار سهام [۲۵]، عملکرد مخزن [۲۶]، توربین هیدرولیک [۲۷] و سیستم‌های الکتریکی [۲۸] ملاحظه نمود. همچنین در موارد بسیاری، این روش با روش‌های دیگر ترکیب شده‌است و از نتیجه‌ی آن الگوریتم جدیدی ارائه و آزمایش شده‌است. ترکیب الگوریتم رقابت استعماری با روش گاوس [۱۶]، عملگرهای دارای جهش[2] [۲۹]، روش‌های خوشه‌بندی فازی [۱۷]، روش SA [۳۰]، شبکه‌های عصبی [۱۹]، منطق و کنترلگر فازی [۲۰، ۲۵، ۳۱]، الگوریتم ژنتیک [۲۱]، حرکت گروهی ذرات[3] [۲۴]، کنترل مود لغزشی [۲۷] و کنترلر PID [۲۸] همگی از مثال‌های این شاخه از تحقیقات هستند.

در راستای تلاش‌های روزافزون برای یافتن روش‌های بهتر و بهینه‌تر برای پاسخ به مسائل بهینه‌سازی عددی، تصمیم بر تبیین روشی جهت بهینه‌سازی عددی گرفته شد. با توجه به فعالیت‌های پیشین انجام شده برروی الگوریتم رقابت استعماری و اثبات کارآیی آن در مطالعات قبلی، در این پژوهش با افزودن جستجوی محلی به بدنه‌ی اصلی الگوریتم در دو مرحله، بهبود به‌سزایی در فرآیند یافتن نقطه‌ی بهینه در توابع ریاضی محک ایجاد شده‌است. با نوآوری انجام شده، هم سرعت همگرایی مناسبی وجود دارد و هم قدرت جستجوی الگوریتم افزایش یافته‌است که این باعث کم کردن احتمال به دام افتادن در کمینه محلی می‌شود. روش استفاده‌شده برای کمینه محلی در این مقاله در مقایسه با پژوهش‌های پیشین، ضمن ایجاد پاسخ مناسب، هم ساده‌تر و هم کاربردی‌تر است و نحوه‌ی استفاده از آن نیز متفاوت است. در نهایت هدف تعلق گرفتن ویژگی‌های ذکر شده، به الگوریتم است. همچنین برای ایجاد نمود فیزیکی بیشتر از کاربردی نوین برای این الگوریتم، در کنترل بالگرد در روش پسخور حالت از آن استفاده

[1] Knapsack 1-0 problem

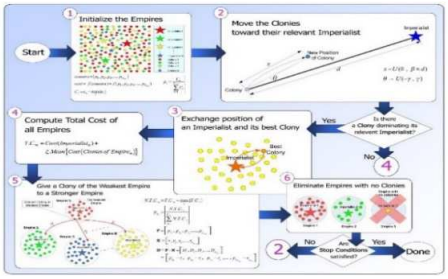
[2] mutation

[3] particle swarm optimization

می‌شود، که قبل از این به چنین مساله‌ای پرداخته نشده‌است، بدین‌گونه که بهره‌های کنترلی از طریق بهینه‌سازی با بهره‌های کنترل پسخور همگرا شوند. دراصل فارغ از وارد شدن به بحث کنترل به‌صورت جزئی، با داشتن دانش کلی در مورد روش کنترلی، با مساله کنترل بالگرد در فاز هاور، با دیدی نوآورانه، به‌صورت یک مساله‌ی صرفا بهینه‌سازی برخورد می‌شود که در آن هدف رسیدن به بهره‌های کنترلی مطلوب است و ایجاد تغییر در اصل روش کنترل در اینجا مورد بحث نیست که البته در پژوهش‌های آینده، قطعا می‌توان به آن پرداخت.

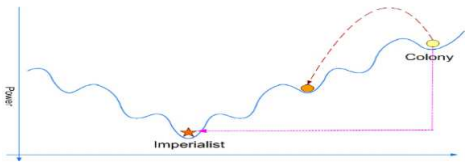
۲- شرح مساله

شکل ۱ نمای کلی الگوریتم توسعه داده شده موسوم به الگوریتم رقابت استعماری (ICA) را نشان می‌دهد [۱].



شکل ۱- نمای کلی الگوریتم رقابت استعماری [۱]

الگوریتم توسعه داده شده، همانند سایر روش‌های بهینه‌سازی تکاملی، با تعدادی جمعیت اولیه شروع می‌شود. در این الگوریتم، هر عنصر جمعیت، یک کشور نامیده می‌شود. کشورها به دو دسته مستعمره و استعمارگر تقسیم می‌شوند. هر استعمارگر، بسته به قدرت خود، تعدادی از کشورهای مستعمره را به سلطه خود درآورده و آن‌ها را کنترل می‌کند. سیاست جذب و رقابت استعماری، هسته‌های اصلی این الگوریتم را تشکیل می‌دهند. مطابق سیاست جذب که به صورت تاریخی، توسط کشورهای استعمارگری همچون فرانسه و انگلیس، در مستعمراتشان اعمال می‌شد، کشورهای استعمارگر با استفاده از روش‌هایی همچون احداث مدارس به زبان خود، سعی در از خود بی خود کردن کشور مستعمره، با از میان بردن زبان کشور مستعمره و فرهنگ و رسوم آن داشتند. در ارائه این الگوریتم، این سیاست با حرکت دادن مستعمرات یک امپراطوری، مطابق یک رابطه خاص صورت می‌پذیرد. در شکل ۲ نیز می‌توان مشاهده کرد که نزدیک شدن یک مستعمره به استعمارگر، بیان‌گر رسیدن به نقطه‌ی بهینه است [۱].



شکل ۲- اعمال سیاست جذب از طرف استعمارگران بر مستعمرات [۱]

اگر در حین حرکت، یک مستعمره، نسبت به استمارگر، به موقعیت بهتری برسد، جای آن دو با هم عوض می‌شود. در ضمن، قدرت کل یک امپراطوری به صورت مجموع قدرت کشور استعمارگر به اضافه درصدی از قدرت میانگین مستعمرات آن تعریف می‌شود [۱]. یعنی:

$$T.C._n = Cost(imperialist_n) + \xi mean\{Cost(colonies\ of\ empire_n)\} \quad (۱)$$

همانگونه که قبلاً نیز بدان اشاره شد، رقابت استعماری، بخش مهم دیگری از این الگوریتم را تشکیل می‌دهد. در طی رقابت استعماری، امپراطوری‌های ضعیف، به تدریج قدرت خود را از دست داده و به مرور زمان با تضعیف شدن از بین می‌روند. رقابت استعماری باعث می‌شود که به مرور زمان، حالتی پیش آید که در آن تنها یک امپراطوری در دنیا وجود دارد که آن را اداره می‌کند. این حالت زمانی است که الگوریتم رقابت استعماری با رسیدن به نقطه بهینه تابع هدف، متوقف می‌شود [۱]. شرط توقف نیز می‌تواند از بین رفتن تمامی فرمانروایی‌ها و باقیماندن تنها یک فرمانروایی یا گذشتن تعداد مشخصی اجرا (طبق تعریف برنامه، دهه) باشد.

به‌عنوان هدف اول و اصلی در این مقاله، بخش‌هایی به الگوریتم اضافه می‌شود تا در عملکرد آن بهبود حاصل شود و سرعت و دقت پاسخ‌ها به‌ویژه در مسائل با ابعاد بالاتر، افزایش یابد و پس از آن با استفاده از الگوریتم توسعه داده شده، یک مساله‌ی کاربردی در کنترل بالگرد مورد بررسی قرار می‌گیرد.

۳- روش حل

در این مقاله، به برنامه‌ی اولیه که کد منبع آزاد[1] آن در دسترس عموم می‌باشد، دو بخش اصلی اضافه شده‌است. این دو بخش عبارتند از:

- عملیات جستجوی محلی در جمع تمامی کشورهای اولیه، به منظور جهت‌دهی به جمعیت اولیه
- فرایند جستجوی محلی برای چند مورد از قدرتمندترین فرمانروایی‌ها و درواقع قائل شدن برتری بیشتر برای فرمانروایی‌های قدرتمندتر.

در مورد اول قبل از اجرای الگوریتم و بعد از انتخاب جمعیت اولیه به‌صورت تصادفی، یک الگوریتم شبه ژنتیک بر روی جمعیت اولیه اجرا می‌شود، تا تمامی نقاط را به‌سمت بهینه‌ی فراگیر پیش ببرد و مکان‌های اولیه‌ی کشورها برای شروع الگوریتم را بهبود ببخشد. با این کار درواقع انتخاب نقاط اولیه، در عین تصادفی بودن، هدفمند می‌شود. هدف از این الگوریتم تنها ایجاد اندکی بهبود در نقاط اولیه است. با این کار هدف استفاده از قدرت الگوریتم ژنتیک برای حل مساله نیست که باعث کم شدن توجه به خود الگوریتم رقابت استعماری شود. این اتفاقی است که در کارهای قبلی بعضا رخ داده‌است. همانطور که قبلا ذکر شد، مثلا در [۶] الگوریتم رقابت استعماری با الگوریتم لانه‌مورچه‌ای ترکیب شده‌است و در هر تکرار کد، یک بار الگوریتم لانه‌مورچه‌ای به‌طور کامل اجرا می‌شود. در این حالت تبعا با توجه به تعداد جمعیت اولیه و محل قرارگیری آن‌ها، الگوریتم به یک نقطه‌ی کمینه همگرا می‌شود که درواقع حاصل تلاش الگوریتم لانه‌مورچه‌ای

[1] Open source

است. اما در روش اجرا شده در این پژوهش تنها بر روی جمعیت اولیه و آن هم با تعداد اندکی نسل (تکرار) الگوریتم ژنتیک توسعه داده‌شده، اجرا شده‌است تا کمی بهبود در نقاط ایجاد شود. از این رو به آن الگوریتم شبه ژنتیک گفته‌شده‌است.

در مورد دوم از یک الگوریتم جستجوی محلی برای بهبود جواب‌ها استفاده شده‌است. قبل از فرایند متحد شدن و رقابت فرمانروایی‌ها و بعد از تشکیل فرمانروایی و انتخاب استعمارگر، سه فرمانروایی که بهترین وضعیت را از نظر هزینه دارند انتخاب شده‌اند و یک امتیاز مثبت برای آن‌ها درنظر گرفته شده‌است. هر کدام از کشورها در این فرمانروایی‌ها با یک عدد تصادفی کمی جابجا می‌شوند تا نقاط جدیدی نیز کشف شوند. این فرایند هم به قدرتمندتر شدن امپراتوری‌های با شرایط مناسب کمک می‌کند، که البته احتمال وجود بهینه‌ی فراگیر در آن‌ها بیشتر است، و هم به دنبال نقاط مناسب احتمالی دیگر می‌گردد. با توجه به این دو مورد، انتظار افزایش سرعت اجرا و یافتن بهینه‌ی فراگیرتر وجود دارد، اما به دلیل افزایش محاسبات، در برخی موارد کمی سرعت اجرا پایین می‌آید، به همین دلیل بعد از هر ۱۰ دهه در اجرای برنامه یک بار این الگوریتم اجرا شده‌است که البته پاسخ نیز قابل قبول بوده‌است.

۳-۱- تشریح الگوریتم شبه ژنتیک توسعه داده‌شده

در این الگوریتم همان‌طور که گفته شد تمامی کشورهای ایجاد شده در ابتدای کار وارد می‌شوند و جمعیت اولیه‌ی الگوریتم ژنتیک را شکل می‌دهند. تعداد والدین با اعمال یک ضریب τ از میان جمعیت تعیین می‌شود. فرایند انتخاب به‌صورت انتخاب دوگانه رقابتی بدون جایگزینی[1] است که پارامتر n=3 در آن برای تعداد انتخاب در هر تورنومنت در نظر گرفته شده‌است. از روش جایگزینی ترکیبی[2] با ضریب β=۰/۵ نیز استفاده شده‌است. در فرایند جهش، یک متغیر جدید در فضا ایجاد می‌شود و ضریب α برای احتمال جهش در آن برابر است با ۰/۸ که در هر گام کاهش می‌یابد، یعنی:

$$\Pr_{i+1} = \alpha \times \Pr_i \qquad (۲)$$

همان ضریب τ انتخاب شده، این بار برای انتخاب از میان جمعیت فرزندان و ورود به جمعیت اصلی استفاده شده‌است و با همان شیوه‌ی انتخاب قبلی از میان فرزندان، چند مورد برگزیده شده‌اند. فرزندان برگزیده به جای ضعیف‌ترین اعضای جمعیت می‌نشینند. الگوریتم به‌همین صورت تا همگرایی پیش می‌رود.

۳-۲- تشریح الگوریتم جستجوی محلی در فرمانروایی‌ها

همانطور که بیان شد، استفاده از این الگوریتم بدین صورت است که در هر ۱۰ تکرار حلقه‌ی اصلی، یک بار الگوریتم جستجوی محلی برای سه مورد از بهترین امپراتوری‌ها اجرا می‌شود. دو فرایند اتحاد دو امپراتوری و رقابت میان فرمانروایی‌ها، بعد از این جستجوی محلی صورت می‌گیرند. پس از انتخاب بهترین فرمانروایی‌ها، دو نوع الگوریتم جستجوی محلی بر روی کل جمعیت هر فرمانروایی اعمال می‌شود. یک نوع جستجو برای امپریالیست است و دیگری برای دیگر کشورها (مستعمرات). اساس روش، الگوریتم جستجوی خط تصادفی [۱۰] است. در شکل ۳ شبه کد این الگوریتم قابل مشاهده است.

برای هر مستعمره، تمامی پارامترهای مساله با یک آرایه‌ی رندوم با مقدار کمینه و بیشینه مشخص جمع می‌شوند و هزینه‌ی نقطه‌ی جدید سنجیده می‌شود. اگر این نقطه بهتر بود، جایگزین نقطه‌ی قبلی می‌شود. این کار برای تعداد مشخصی تکرار که با پارامتر l نشان داده می‌شود، انجام می‌شود. به محض مشاهده‌ی نقطه‌ای بهتر، دیگر نوبت کشور بعدی است و الگوریتم دوباره اجرا نمی‌شود. برای عدد تصادفی از پارامتر λ استفاده می‌شود که طبق تعریف، مقدار آن بین ۱- و ۱ است. این عدد در ۱۰ درصد از اندازه‌ی کل فضای جستجو ضرب می‌شود تا از یک حددی فراتر نرفته باشد. در این الگوریتم مقدار l برابر ۵ در نظر گرفته شده‌است.

```
1.  For each dimension k do
2.    counter ← 1
3.    While counter ≤ l do
4.          Do
5.             y ← x
6.             λ ← U(-1,1)
7.             y_k ← y_k + λ×d
8.          while (y_k > u_k or y_k < l_k)
9.          If f(y) < f(x) then
10.               x ← y
11.               counter ← l
12.         End if
13.         counter ← counter + 1
14.      End while
15.  End for
```

شکل ۳- شبه کد روش جستجوی خط تصادفی

برای هر امپریالیست نیز به همین صورت می‌توان پیش رفت و در واقع در هر گام حل، یک بعد متغیر مساله تغییر می‌کند و سپس به سراغ بُعدِ بعدی می‌رود. در اینجا برای تعداد تکرار از l استفاده شده‌است که مقدار آن را برابر ۴ درنظر گرفته شده‌است. پارامترهای λ و d همانند قبل هستند.

در نهایت کل الگوریتم موجود برای جستجوی محلی در فرمانروایی‌ها را می‌توان با شبه کد زیر نشان داد.

1. **For** each of 3 best countries ii
2. **Run** imperialist local search
3. **Run** colonies local search
4. **Change** imperialist if better one is found
5. **Compute** the total cost of empire
6. **End For**

۴- آزمایش، نتایج و بحث

در این بخش در ابتدا شش نمونه از توابع محک معروف معرفی و سپس عملکرد الگوریتم اولیه و جدید مقایسه شده‌اند. در ادامه این توابع در حالت ۳ بعدی (وجود ۲ پارامتر بهینه‌سازی n=2) مشاهده می‌شوند.

$$f_1(x) = \sum_{i=1}^{n} x_i^2 \qquad ; min\ at\ (0,0) = 0 \qquad (۳)$$

$$f_2(x) = \sum_{i=1}^{n} \left(i \times x_i^4\right) \qquad ; min\ at\ (0,0) = 0 \qquad (۴)$$

$$f_3(x) = \frac{1}{2} + \frac{\left(\sin\sqrt{x_1^2 + x_2^2} - 0.5\right)}{\left(1 + 0.001\left(x_1^2 + x_2^2\right)\right)^2} \qquad (۵)$$

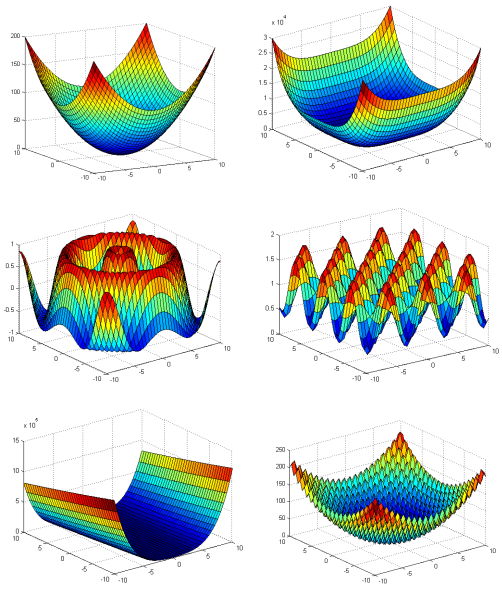
$$; min\ at\ (-4.6849, 0.0127) = -0.9359$$

[1] double tournament selection without replacement
[2] Blended Crossover

$$f_4(x)=1+\frac{1}{4000}\sum_{i=1}^{n}x_i^{\,2}-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) \quad (۶)$$
$$;\, min\ at\ (0,0)=0$$

$$f_5(x)=\sum_{i=1}^{n-1}\left(100\times\left(x_{i+1}-x_i^{\,2}\right)^2+\left(x_i-1\right)^2\right) \quad (۷)$$
$$;\, min\ at\ (1,1)=0$$

$$f_6(x)=\sum_{i=1}^{n}\left(10+x_i^{\,2}-10\cos\left(2\pi x_i\right)\right) \quad (۸)$$
$$;\, min\ at\ (0,0)=0$$



شکل ۴- توابع محک شش‌گانه به ترتیب از بالا و راست

۴-۱- مقایسه‌ی دو روش در توابع محک با دو پارامتر

در این قسمت دو الگوریتم موجود برای هر یک از شش تابع محک به تعداد ۳۰ بار اجرا و نتایج بهترین جواب‌ها در جدول ۱ مقایسه می‌شوند.

پارامترهای انتخابی برای الگوریتم به شرح زیر هستند:

Search Space = [-10 10]
Countries = 200
Initial Imperialists = 8
All Colonies = Countries – Initial Imperialists
Decades = 2000
Revolution Rate = 0.03
Assimilation Coefficient = 2
Assimilation Angle Coefficient = 0.5
Zeta = 0.02
Damp Ratio = 0.99
Uniting Threshold = 0.02
Zarib = 1.05
Alpha = 0.1

با توجه به داده‌های جدول مشاهده می‌شود که در زمان حل الگوریتم چندان تغییری ایجاد نشده است و در پاره‌ای از موارد سرعت افزایش نیز یافته‌است. با توجه به کم بودن تعداد پارامترها، الگوریتم اصلی نیز به‌خوبی پاسخ بهینه را پیدا کرده‌است که البته با این وجود، پاسخ‌های الگوریتم توسعه داده‌شده در پاره‌ای از موارد اندکی دقیق‌تر و بهینه‌تر نیز هستند. درضمن در مورد توابع پیچیده‌تر، توانایی الگوریتم توسعه داده‌شده اندکی بهتر بوده‌است. این نتایج نزدیک بودن پاسخ‌های دو الگوریتم به یکدیگر را نشان می‌دهند. افزایش یافتن زمان حل در الگوریتم توسعه داده شده، تا حدی طبیعی است زیرا برای یک مساله‌ی ساده، بار محاسباتی افزایش یافته‌است و به‌تبع آن زمان زیاد شده‌است. زیاد شدن زمان که خود ناشی از افزایش جستجو است، گاهی به یافتن پاسخ بهینه‌تر کمک کرده‌است و در مواردی نیز بی‌تاثیر بوده‌است.

۴-۲- مقایسه‌ی دو روش در توابع محک با ۳۰ پارامتر

در این قسمت دو الگوریتم موجود برای هر یک از شش تابع محک به تعداد ۳۰ بار اجرا و نتایج بهترین جواب‌ها در جدول ۲ مقایسه شده‌اند. پارامترهای الگوریتم همان موارد قبلی هستند.

در این مساله به دلیل تعداد پارامتر زیاد، پیچیدگی افزایش یافته‌است و مقایسه‌ی دو روش، اندکی آسان‌تر می‌شود. با توجه به داده‌های جدول ۲ مشاهده می‌شود که زمان پاسخ در برخی از موارد بیشتر و در برخی دیگر کمتر شده‌است ولی در مجموع تغییر محسوسی در این بخش وجود ندارد و به‌دلیل افزایش بار محاسباتی در روش ارائه شده نمی‌توان انتظار خاصی برای کاهش زمان محاسبات داشت. در هر حال در روند توسعه‌ی الگوریتم هدف این بوده‌است که زمان پاسخ مساله، تغییر محسوسی پیدا نکند که این امر محقق گشته‌است. میانگین پاسخ‌های بهینه در این حالت برای الگوریتم توسعه داده‌شده به مراتب بهتر شده است و در مواردی تا ۱۲ رقم اعشار بهبود یافته‌است. تنها در مورد تابع سوم الگوریتم اصلی هم برای پاسخ مناسب بوده‌است و پاسخ بهینه با همان الگوریتم تولید شده‌است. الگوریتم توسعه داده‌شده به دلیل افزایش جستجوی محلی پاسخ بهینه‌تری تولید کرده‌است و مقدار کمینه‌ی نهایی را بهبود بخشیده‌است. در مجموع می‌تـوان گفت که این الگوریتم در مقابل توابع محک بسیار موفق بوده‌است.

جدول ۱- نتایج ۳۰ بار اجرای الگوریتم (n=2)

تابع	الگوریتم اصلی			الگوریتم توسعه داده شده		
	میانگین زمان هر اجرا (ثانیه)	میانگین بهترین پاسخ	کمینه بهترین پاسخ	میانگین زمان هر اجرا (ثانیه)	میانگین بهترین پاسخ	کمینه بهترین پاسخ
اول	۲/۵۹	۰/۲۴۰۲e-۲۷۴	۱/۹۶۰۵e-۲۸۲	۲/۸۸	۱/۰۶۸۷e-۲۷۳	۶/۴۸۲۶e-۲۸۵
دوم	۳/۲۴	۰	۰	۳/۱۴	۰	۰
سوم	۸/۹	-۰/۹۳۵۹	-۰/۹۳۵۹	۱۰/۶۳	-۰/۹۳۵۹	-۰/۹۳۵۹
چهارم	۴/۷۸	۱/۰۰۱۷	۰	۲/۸۹	۰	۰

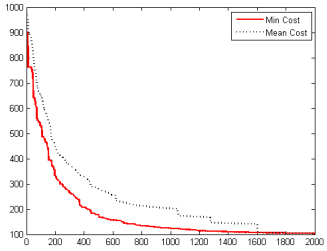
پنجم	۲/۸۲	۱/۰۵۵۹e-۳۱	۰	۲/۹۸	۴/۳۳۰۵e-۳۱	۰
ششم	۲/۵۲	۰	۰	۲/۹۴	۰	۰

جدول ۲- نتایج ۳۰ بار اجرای الگوریتم (n=30)

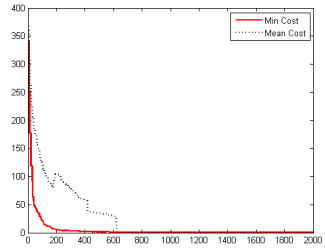
تابع	الگوریتم اصلی			الگوریتم توسعه داده شده		
	میانگین زمان هر اجرا (ثانیه)	میانگین بهترین پاسخ	کمینه بهترین پاسخ	میانگین زمان هر اجرا (ثانیه)	میانگین بهترین پاسخ	کمینه بهترین پاسخ
اول	۸/۶۵	۰/۰۰۱۱	۴/۹۰۸۷e-۶	۳/۷۷	۸/۷۲۸۹e-۷	۲/۰۸۱۷e-۸
دوم	۱۴/۷۸	۳۳۴/۰۷۱۶	۱/۴۷۸۴e-۷	۹/۳۱	۴/۷۹۸۴e-۱۰	۶/۹۰۸۹e-۱۳
سوم	۱۳/۴۵	-۰/۹۳۵۹	-۰/۹۳۵۹	۱۹/۵۵	-۰/۹۳۵۹	-۰/۹۳۵۹
چهارم	۶/۹۷	۰/۰۸	۲/۳۷۰۲e-۶	۱۰/۴۰	۰/۰۰۲۲	۲/۸۲۹۲e-۱۰
پنجم	۷/۶۱	۵۸/۶۹۲۴	۳/۳۵۰۵	۱۰/۹۵	۳۰/۴۹۶۷	۳/۸۲۲۵
ششم	۸/۱۰	۶۹/۶۹۵۵	۰/۱۷۷۸	۶/۳۶	۰/۰۲۵۹	۵/۳۶۳۹e-۷

در شکل ۵ برای نمونه، نمودارهای همگرایی برای حالت وجود ۳۰ پارامتر و برای تابع ششم، توسط دو الگوریتم نشان داده‌شده‌اند.

(الف)



(ب)



شکل ۵- نمودار همگرایی بر حسب اجرا برای تابع ششم با (الف) الگوریتم اصلی (ب) الگوریتم توسعه داده شده

طبق دو نمودار مشاهده می‌شود که نرخ همگرایی الگوریتم توسعه داده شده بسیار بالاتر است و الگوریتم با سرعت بالاتری همگرا می‌شود. حذف تمامی امپراتوری‌ها و باقی ماندن یک امپراتوری نیز بسیار سریع‌تر رخ می‌دهد. الگوریتم توسعه داده‌شده پس از دو هزار تکرار کاملا به صفر همگرا شده‌است، اما الگوریتم اصلی هنوز در حال رسیدن به صد است. افت پاسخ‌های تابع هزینه نیز در روش توسعه داده‌شده از همان ابتدا با سرعت بالا انجام می‌شود.

۵- آزمایش کاربردی در کنترل پرواز بالگرد

حال برای استفاده از الگوریتم تکاملی مورد بحث، یک مدل خطی استاندارد بالگرد ۶ درجه آزادی که مربوط به بالگرد Lynx ZD559 است و در مرجع [۳۲] اطلاعات دقیق آن در دسترس است، انتخاب می‌شود. سپس از دو روش جایابی قطب و الگوریتم تکاملی توسعه داده‌شده، سعی در پیدا کردن بهره‌های کنترلی مناسب برای کنترل پسخور حالت این بالگرد می‌شود. مقاومت کنترلر نیز با ایجاد ورودی‌های اغتشاشی در سیستم کنترل، مورد آزمایش قرار می‌گیرد.

برای حرکت این بالگرد سرعت ۲۰ گره انتخاب می‌شود و کنترل روی آن پیاده‌سازی می‌گردد. این حرکت تداعی کننده فاز نزدیک شدن[1]، پیش از فرود است.

۵-۱- سناریوی اول

در این مرحله به زاویه‌ی فراز بالگرد یک مقدار اولیه داده می‌شود ($\theta_0 = 20^\circ$) و سعی در کنترل این زاویه می‌گردد.

۱) جایابی قطب

ابتدا با تعیین ماتریس بهره‌های کنترلی در روش پسخور حالت و از طریق جایابی قطب‌ها، این کنترل صورت می‌گیرد. در این شرایط قطب‌های مطلوب در محلی مناسب قرار می‌گیرند تا پایداری سیستم تضمین گردد.

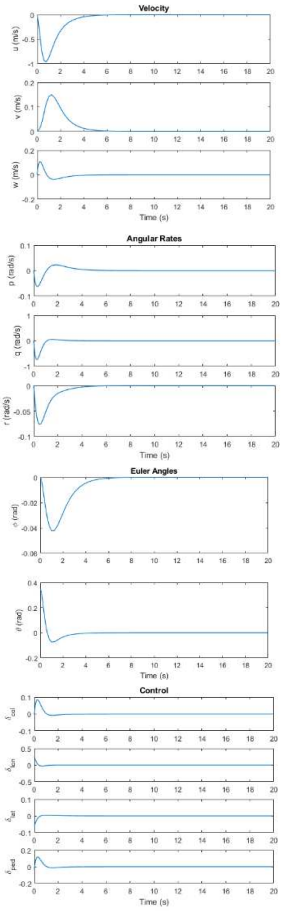
$$\lambda = [\ -1 \quad -2 \quad -3\pm 2i \quad -4\pm 2i \quad -5 \quad -6\ ] \qquad (۹)$$

طبق نتایج ماتریس بهره‌ی مناسب برای کنترل زاویه‌ی فراز به‌قرار زیر است.

$$k = \begin{bmatrix} 0.0041 & -0.0459 & -0.1244 & -0.0072 & -0.0047 & -0.0013 & 0.0059 & -0.0131 \\ -0.0626 & 0.0068 & 0.2015 & 0.7833 & -0.0289 & 0.0289 & -0.0465 & 0.0689 \\ 0.0161 & -0.0070 & -0.0378 & -0.1784 & 0.0052 & 0.0183 & -0.0643 & -0.0075 \\ 0.0440 & -0.0831 & -0.1842 & -0.0922 & 0.1921 & 0.2277 & 1.0290 & -0.8594 \end{bmatrix} \qquad (۱۰)$$

طبق نتایج شبیه‌سازی نیز می‌توان دید:

[1] approach



شکل ۶- تاریخچه‌ی متغیرهای حالت و ورودی‌های کنترلی در سرعت ۲۰ نات با جایابی قطب و $\theta_0 = 20^\circ$

بنابراین، سیستم به‌خوبی کنترل شده‌است و ضمن اینکه پایدار مانده‌است، مقادیر عددی متغیرها نیز در محدوده‌ی مناسبی قرار دارد.

۲) بهینه‌سازی از طریق الگوریتم رقابت استعماری

در این مرحله سعی می‌شود تا مقدار عددی درایه‌های ماتریس k، به‌عنوان پارامتر تحت بهینه‌سازی، با روش بهینه‌سازی الگوریتم رقابت استعماری و از طریق کمینه ساختن یک تابع هزینه، بدست آیند. این تابع هزینه به‌صورت جمع درایه‌های اختلاف دو ماتریس k تحت بهینه‌سازی و k مطلوب از طریق جایابی قطب، تعیین می‌شود. در واقع با الگوریتم بهینه‌سازی سعی در رساندن ماتریس k به مقدار مطلوبش می‌شود. بنابراین:

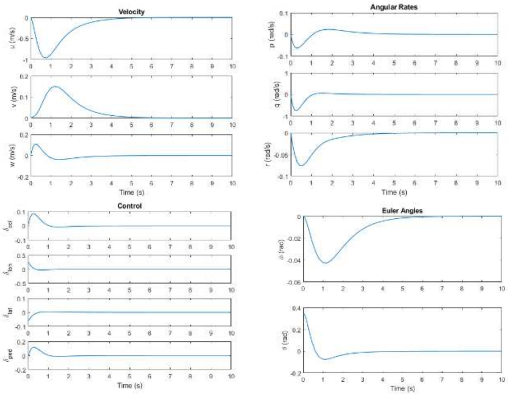
ماتریس بهره‌ی مناسب برای کنترل زاویه‌ی فراز به‌قرار زیر است.

$$k = \begin{bmatrix} 0.0041 & -0.0460 & -0.1244 & -0.0072 & -0.0047 & -0.0013 & 0.0059 & -0.0131 \\ -0.0626 & 0.0068 & 0.2015 & 0.7833 & -0.0289 & 0.0289 & -0.0465 & 0.0689 \\ 0.0161 & -0.0070 & -0.0378 & -0.1784 & 0.0052 & 0.0183 & -0.0643 & -0.0075 \\ 0.0440 & -0.0831 & -0.1842 & -0.0922 & 0.1921 & 0.2277 & 1.0290 & -0.8593 \end{bmatrix} \quad (۱۱)$$

با انتخاب مناسب پارامترهای الگوریتم، می‌توان یافت که:

Cost: 10303×10-4
Time: 148 (s)

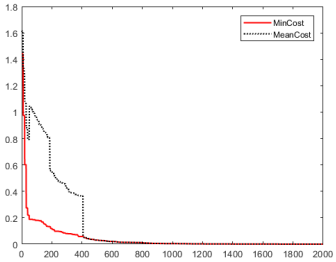
طبق نتایج شبیه‌سازی نیز می‌توان دید:



شکل ۷- تاریخچه‌ی متغیرهای حالت و ورودی‌های کنترلی در سرعت ۲۰ نات با بهینه‌سازی و $\theta_0 = 20^\circ$

بنابراین، سیستم به مانند قبل به‌خوبی کنترل شده‌است و پایدار مانده‌است.

همچنین روند همگرایی و کاهش تابع هزینه را می‌توان در زیر، مشاهده نمود:



شکل ۸- تاریخچه تغییرات تابع هزینه با سرعت ۲۰ نات و با ورودی $\theta_0 = 20^\circ$

الگوریتم بهینه‌سازی ارائه شده، توانسته است در زمانی مناسب و با دقتی بالا، بهره‌های کنترلی مطلوب را پیدا کند و سیستم را در جهت رسیدن به پایداری و تنظیم مقدار نهایی متغیرهای حالت، کنترل کند. در زمان کوتاهی پس از شروع شبیه‌سازی و اعمال مقدار اولیه نامطلوب در زاویه‌ی فراز، با مقادیر مناسبی از زمان فرونشست و مقدار پرش[1]، سیستم به حالت پایدار بازگشته‌است و مقدار نهایی به‌سرعت و با دقت دنبال شده‌است. با توجه به ذات سیستم، مقادیر سیگنال‌های کنترلی در محدوده‌ای معقول و با مقدار بیشینه‌ای منطقی از نظر فیزیک قرار دارند و در زمان کوتاهی اعمال شده‌اند تا تلاشی بیش از حد برای کنترل انجام نشود. متناسب با این سیگنال‌ها اکثر متغیرهای حالت در کوتاه‌ترین زمان ممکن و تقریبا در همان مدت زمان اعمال سیگنال کنترلی، رگوله و کنترل شده‌اند، البته برخی متغیرها نظیر زوایای اویلر، مدت زمان بیشتری را صرف رسیدن به مقدار نهایی کرده‌اند.

۵-۲- سناریوی دوم

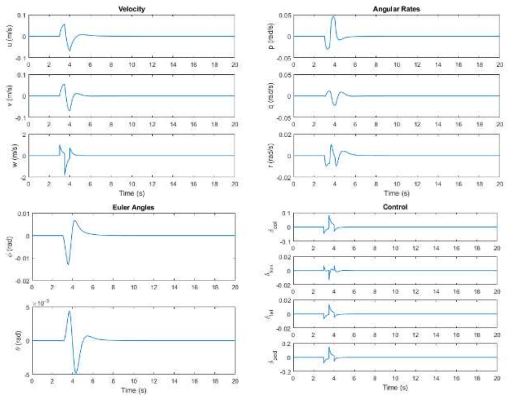
در این مرحله به سرعت عمودی یعنی w یک ورودی اغتشاشی

[1] overshoot

به‌صورت موج دوتایی[1] وارد می‌شود تا از نظر فیزیک یک تندباد لحظه‌ای را شبیه‌سازی کند. در این شرایط با دامنه ۱ و مدت زمان ۱ ثانیه برای هر یک از دو موج بالایی و پایینی، این سیگنال وارد می‌شود و سعی در کنترل سرعت می‌گردد.

۱) جایابی قطب

در اینجا با همان مکان قطب‌ها و همان ماتریس بهره‌ها طبق نتایج شبیه‌سازی نیز می‌توان دید:



شکل ۹- تاریخچه‌ی متغیرهای حالت و ورودی‌های کنترلی در سرعت ۲۰ نات با جایابی قطب و اغتشاش w

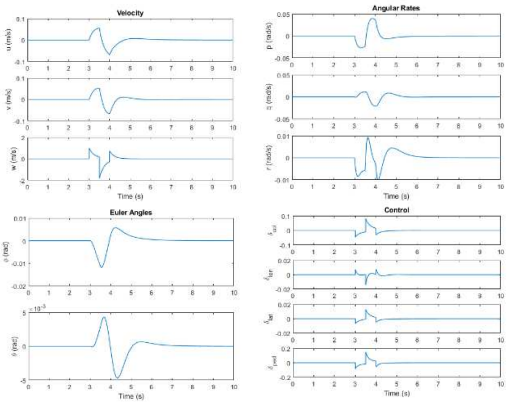
بنابراین، در این حالت نیز سیستم به‌خوبی کنترل شده‌است و از این اغتشاش تاثیرپذیری نداشته است.

۲) بهینه‌سازی از طریق الگوریتم رقابت استعماری

به مانند قبل در این مرحله نیز با انتخاب مناسب پارامترهای الگوریتم، می‌توان یافت که:

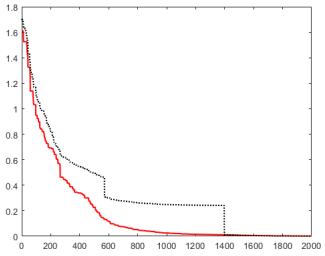
Cost: 4.3117×10-4
Time: 89 (s)

طبق نتایج شبیه‌سازی نیز می‌توان دید:



شکل ۱۰- تاریخچه‌ی متغیرهای حالت و ورودی‌های کنترلی در سرعت ۲۰ نات با بهینه‌سازی و اغتشاش w

بنابراین، همانند حالت قبل، سیستم به‌خوبی کنترل شده‌است و پایدار مانده‌است. در زمان اعمال ورودی، اندکی اغتشاش در حرکت مشاهده می‌شود، که با اعمال ورودی کنترلی در جهت عکس و به‌منظور خنثی‌سازی اثر ورودی مزاحم، سیستم دوباره به پایداری باز می‌گردد. آخرین فرمان کنترلی در ثانیه ۴ اعمال می‌شود و با اثرگذاری آن در سیستم، از حدود ثانیه ۵ به‌بعد، سیستم کاملا پایدار می‌شود و به شرایط ایده‌آل قبلی خودش باز می‌گردد. روند همگرایی و کاهش تابع هزینه را می‌توان در شکل ۱۱ مشاهده نمود:



شکل ۱۱- تاریخچه تغییرات تابع هزینه با سرعت ۲۰ نات و اغتشاش w

در این حالت نیز در زمانی مناسب و با دقتی بالا، بهره‌های کنترلی مطلوب پیدا شده‌اند و سیستم در جهت رسیدن به پایداری و تنظیم مقدار نهایی متغیرهای حالت، کنترل شده‌است. با شروع شبیه‌سازی، سیستم در حالت کاملا تحت کنترل و تعادلی[2] حرکت می‌کند تا اینکه با ایجاد ورودی اغتشاشی نامطلوب در سرعت عمودی، در زمان کوتاهی، با مقادیر مناسبی از زمان فرونشست و مقدار پرش، سیستم به حالت پایدار بازگشته‌است و مقدار نهایی مطلوب سیستم دنبال شده‌است. سیگنال‌های کنترلی در محدوده‌ای معقول و با مقدار بیشینه منطقی از نظر فیزیکی که البته از سناریوی قبلی کمتر شده‌است ولی زمان اعمال آن اندکی افزایش یافته، قرار دارند. متناسب با این سیگنال‌ها اکثر متغیرهای حالت در کوتاه‌ترین زمان ممکن و تقریبا در همان مدت زمان اعمال سیگنال کنترلی، رگوله و کنترل شده‌اند و به مقدار نهایی رسیده‌اند.

در سناریوی ۲ نسبت به سناریوی ۱، با توجه به نوع اغتشاش وارد شده از نظر فیزیکی، زمان بهینه‌سازی و رسیدن به مقدار بهینه در فرایند بهینه‌سازی، که درواقع همان تاریخچه کاهش مقدار تابع هزینه است، بیشتر شده‌است و الگوریتم مدت زمان بیشتری را صرف کاهش هزینه و رسیدن به بهینه فراگیر کرده‌است. مقدار نهایی تابع در هر دو حالت مناسب است، اما در سناریوی اول سرعت در کاهش تابع هزینه تقریبا دو برابر سناریوی دوم است.

۵-۳- مقایسه با روش‌های مشابه

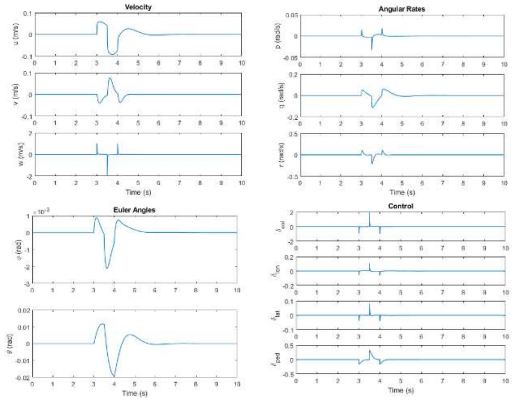
برای کنترل چنین سیستمی می‌توان از روش‌های متنوع و ابداعی زیادی استفاده نمود. بسیاری از این روش‌ها قبلا در کنترل پرواز بالگرد در فازهای مختلف، استفاده شده‌اند. برای مثال برخی از این روش‌ها براساس منطق بهینه هستند، مانند LQR، که طرفداران زیادی نیز دارند. در این روش سعی بر تعیین بهره‌های کنترلی برای تنظیم و پایداری مقادیر متغیرهای حالت می‌شود.

1 doublet

2 trim

در مقایسه‌ی این روش با عملکرد الگوریتم رقابت استعماری می‌توان به فاکتورهایی نظیر حفظ پایداری کلی و مقاوم بودن، زمان فرونشست و میزان پرش، در برابر ورودی‌های اغتشاشی توجه نمود.

در حالت کلی باوجود اغتشاش در سرعت عمودی می‌توان دریافت که با استفاده از هر دو روش، پایدار شدن سیستم و رسیدن به پاسخ مطلوب اتفاق میفتد. در ادامه می‌توان نتایج استفاده از روش LQR را برای مقادیر مشخصی از Q و R مشاهده نمود.



شکل ۱۲- تاریخچه‌ی متغیرهای حالت و ورودی‌های کنترلی در سرعت ۲۰ نات با LQR و اغتشاش w

در روشی بهینه نظیر LQR با انتخاب ماتریس‌های Q و R مناسب، می‌توان پاسخ‌هایی با کیفیت‌های متفاوت یافت، اما کلیت پاسخ تقریبا به یک شکل است.

همانطور که در نمودارها مشخص است، پایداری سیستم رخ داده‌است، اما نسبت به الگوریتم بهینه‌سازی در روش LQR سیگنال کنترلی در یک لحظه و در مدت زمان کوتاه‌تری اعمال شده‌است ولی پرش آن بیشتر بوده‌است. در مورد متغیرهای حالت، در اکثر موارد میزان پرش در LQR بیشتر شده‌است و زمان فرونشست اندکی کمتر. هرچند در مواردی برای برخی از متغیرهای حالت مانند سرعت‌های w و v عدم تغییری چندان در پرش و کاهش زمان فرونشست مشاهده می‌شود.

با توجه به نمونه‌های بررسی شده می‌توان مشاهده نمود که الگوریتم رقابت استعماری توسعه داده شده، در کاربرد تازه خود، یعنی کنترل پرواز بالگرد، تحت شرایط تعیین شده، به‌خوبی عمل کرده‌است و پاسخ قابل قبولی ارائه داده‌است. از این الگوریتم می‌توان در بهینه‌سازی پارامترها در مسائل، استفاده نمود و از دقت و سرعت مناسب آن نسبت به روش پایه‌ی الگوریتم رقابت استعماری، بهره برد.

۶- نتیجه‌گیری

در این مقاله، الگوریتم جدیدی بر مبنای توسعه‌ی الگوریتم رقابت استعماری و افزودن جستجوی محلی به آن ارائه شد و این کار در دو مرحله صورت گرفت. در مرحله‌ی اول با یک الگوریتم ژنتیک با تعداد نسل کم، انتخاب نقطه‌ی شروع مساله هدفمند شده‌است و در مرحله‌ی دوم از روش جستجوی خط تصادفی پاسخ برخی از فرمانروایی‌های قدرتمند بهبود یافته‌است. با ابداعات انجام شده بر روی الگوریتم مشاهده شد که فرایند رسیدن به جواب دقیق‌تر می‌شود. تقریبا با حفظ همان زمان اجرا و اضافه نشدن زمان قابل توجه به فرآیند حل، عملکرد الگوریتم بهبود یافته‌است و نقاط بهینه‌تری نسبت به الگوریتم اصلی پیدا شده‌اند. به‌خصوص در زمانی که مساله پیچیده‌تر شده‌است و تعداد پارامترهای بهینه‌سازی افزایش یافته‌اند، در تمامی موارد، روش ابداعی پاسخ بهینه‌تری تولید کرده‌است و کمینه‌ی فراگیر به مقدار مطلق خود نزدیک‌تر شده‌است. سپس برای استفاده عملی از این روش، مساله‌ی کنترل پرواز یک مدل بالگرد خطی مورد بررسی قرار گرفت و پارامترهای ماتریس بهره‌های کنترلی در روش پسخور حالت، با استفاده از الگوریتم بهینه‌سازی تعیین شدند. این کاربرد، استفاده‌ای تازه از الگوریتم مورد نظر است. در این مساله ابتدا با روش جایابی قطب، بهره‌های مطلوب متناسب با قطب‌های مطلوب مشخص شده‌اند و الگوریتم بهینه‌سازی در زمانی مناسب و با دقت بالا، به پارامترهای ماتریس بهره همگرا شده‌است. این آزمایش در حالت‌های مختلف و با ورودی‌های متفاوتی انجام شده‌است. مقادیر متفاوتی برای سرعت پرواز بالگرد، وجود مقدار اولیه در متغیرهای حالت و ورود مغادیر اغتشاشی به متغیرهای حالت در حین پرواز بررسی و آزمایش شده‌اند که در تمامی آن‌ها ماتریس بهره از طریق بهینه‌سازی، به‌خوبی تعیین شده‌است و سیستم در عین پایداری، پاسخی مطلوب از نظر زمان فرونشست و میزان پرش و مقدار نهایی، می‌دهد. در انتها نیز یکی از سناریوهای تست این روش با روش کنترل بهینه LQR مقایسه شده‌است و در مجموع در عین تفاوت اندک، تطابق نسبی مناسبی میان پاسخ‌ها وجود دارد. برای ادامه پژوهش نیز می‌توان از ایجاد تغییراتی در خود روش کنترل با کمک خواص الگوریتم بهینه‌سازی مذکور و درواقع مطالعه دقیق‌تر در زمینه فیزیک مساله از جنبه کنترلی، بهره برد.

۷- نمادها

d	اندازه‌ی فضای جستجو
f(x)	تابع عمومی
i	شمارنده‌ی عمومی
k	بهره‌ی کنترلی یا ماتریس بهره‌های کنترلی
l	شمارنده در جستجوی محلی
n	شمارنده جمعیت
Pr	پارامتر جهش ژنتیک
T.C.	هزینه‌ی نهایی
w	سرعت عمودی (m/s)
x, y	پارامترهای عمومی مجهول
α	ضریب جهش ژنتیک
β	ضریب ترکیب نسل ژنتیک
θ	زاویه‌ی فراز (^{o} / rad)
λ	عدد تصادفی در جستجوی محلی
ξ	ضریب قدرت مستعمرات
τ	ضریب انتخاب والدین در الگوریتم ژنتیک

۸- مراجع

[1] Atashpaz-Gargari E, *Social optimization algorithm development and performance review*, MSc. Thesis, Tehran University School of Electrical and Computer Engineering, 2009.

[2] Atashpaz-Gargari E. and Lucas C., Imperialist competitive algorithm: an algorithm for optimization inspired by

imperialistic competition. *In IEEE Congress on Evolutionary computation*, Singapore, 2007.

[3] Zhang Y., Wang Y. and Peng C., Improved imperialist competitive algorithm for constrained optimization. *In International Forum on Computer Science-Technology and Applications*, United States, 2009.

[4] Bahrami H., Faez K. and Abdechiri M., Imperialist competitive algorithm using chaos theory for optimization (CICA). *In 12th international conference on Computer modelling and simulation (UKSim)*, Cambridge, United Kingdom, 2010.

[5] Niknam T., Fard E. T., Ehrampoosh S. and Rousta A, A new hybrid imperialist competitive algorithm on data clustering. *Sadhana*, Vol. 36, No.3, pp. 293, 2011.

[6] Sabour M. H., Eskandar H. and Salehi P., Imperialist competitive ant colony algorithm for truss structures. *world applied sciences journal*, Vol. 12, No.1, pp. 94-105, 2011.

[7] Lin J. L., Tsai Y. H., Yu C. Y. and Li M. S., Interaction enhanced imperialist competitive algorithms. *Algorithms*, Vol. 5, No.4, pp. 433-448, 2012.

[8] Nozarian S. and Jahan MV., A novel memetic algorithm with imperialist competition as local search. *In International Proceedings of Computer Science and Information Technology*, Hong Kong, 2012.

[9] Lin J. L., Cho C. W. and Chuan H. C., Imperialist competitive algorithms with perturbed moves for global optimization. *Applied Mechanics and Materials*, Vol. 284, pp. 3135-3139, 2013.

[10] Lin J. L., Chuan H. C., Tsai Y. H. and Cho C. W., Improving imperialist competitive algorithm with local search for global optimization. *In 7th Asia Modelling Symposium (AMS)*, Hong Kong, 2013.

[11] Birbil Ş. İ. and Fang S. C., An electromagnetism-like mechanism for global optimization. *Journal of global optimization*, Vol. 2, No.3, pp. 263-282, 2003.

[12] Hajihassani M., Armaghani D. J., Marto A. and Mohamad E. T., Ground vibration prediction in quarry blasting through an artificial neural network optimized by imperialist competitive algorithm. *Bulletin of Engineering Geology and the Environment*, Vol. 74, No.3, pp. 873-886, 2015.

[13] Rabiee A., Sadeghi M. and Aghaei J., Modified imperialist competitive algorithm for environmental constrained energy management of microgrids. *Journal of Cleaner Production*, Vol. 202, pp. 273-292, 2018.

[14] Ali E. S., Speed control of induction motor supplied by wind turbine via imperialist competitive algorithm. *Energy*, Vol. 89, pp. 593-600, 2015.

[15] Abd-Elazim S. M. and Ali E. S., Imperialist competitive algorithm for optimal STATCOM design in a multimachine power system. *International Journal of Electrical Power & Energy Systems*, Vol. 76, pp. 136-146, 2016.

[16] Ghasemi M., Ghavidel S., Ghanbarian M. M. and Gitizadeh M., Multi-objective optimal electric power planning in the power system using Gaussian bare-bones imperialist competitive algorithm. *Information Sciences*, Vol. 294, pp. 286-304, 2015.

[17] Mikaeil R., Haghshenas S. S., Haghshenas S. S. and Ataei M., Performance prediction of circular saw machine using imperialist competitive algorithm and fuzzy clustering technique. *Neural Computing and Applications*, Vol. 29, No.6, pp. 283-292, 2018.

[18] Ardalan Z., Karimi S., Poursabzi O. and Naderi B., A novel imperialist competitive algorithm for generalized traveling salesman problems. *Applied Soft Computing*, Vol. 26, pp. 546-555, 2015.

[19] Aghajani A., Kazemzadeh R. and Ebrahimi A., A novel hybrid approach for predicting wind farm power production based on wavelet transform, hybrid neural networks and imperialist competitive algorithm. *Energy Conversion and Management*, Vol. 121, pp. 232-240, 2016.

[20] Sadeghi J., Mousavi S. M. and Niaki S. T., Optimizing an inventory model with fuzzy demand, backordering, and discount using a hybrid imperialist competitive algorithm. *Applied Mathematical Modelling*, Vol. 40, No.15-16, pp. 7318-7335, 2016.

[21] Nia A. R., Far M. H. and Niaki S. T., A hybrid genetic and imperialist competitive algorithm for green vendor managed inventory of multi-item multi-constraint EOQ model under shortage. *Applied Soft Computing*, Vol. 30, pp. 353-364, 2015.

[22] Fathy A. and Rezk H., Parameter estimation of photovoltaic system using imperialist competitive algorithm. *Renewable Energy*, Vol. 111, pp. 307-320, 2017.

[23] Mirhosseini M. and Nezamabadi-pour H., BICA: a binary imperialist competitive algorithm and its application in CBIR systems. *International Journal of Machine Learning and Cybernetics*, Vol. 9, No.12, pp. 2043-2057, 2018.

[24] Mehdinejad M., Mohammadi-Ivatloo B., Dadashzadeh-Bonab R. and Zare K., Solution of optimal reactive power dispatch of power systems using hybrid particle swarm optimization and imperialist competitive algorithms. *International Journal of Electrical Power & Energy Systems*, Vol. 83 pp. 104-116, 2016.

[25] Sadaei H. J., Enayatifar R., Lee M. H. and Mahmud M., A hybrid model based on differential fuzzy logic relationships and imperialist competitive algorithm for stock market forecasting. *Applied Soft Computing*, Vol. 40, pp. 132-149, 2016.

[26] Hosseini-Moghari S. M., Morovati R., Moghadas M. and Araghinejad S., Optimum operation of reservoir using two evolutionary algorithms: imperialist competitive algorithm (ICA) and cuckoo optimization algorithm (COA). *Water resources management*, Vol. 29, No.10, pp. 3749-3769, 2015.

[27] Chen Z., Yuan X., Yuan Y., Lei X. and Zhang B., Parameter estimation of fuzzy sliding mode controller for hydraulic turbine regulating system based on HICA algorithm. *Renewable Energy*, Vol. 133, pp. 551-565, 2019.

[28] Taher S. A., Fini M. H. and Aliabadi S. F., Fractional order PID controller design for LFC in electric power systems using imperialist competitive algorithm. *Ain Shams Engineering Journal*, Vol. 5, No.1, pp. 121-135, 2014.

[29] Xu S., Wang Y. and Lu P., Improved imperialist competitive algorithm with mutation operator for continuous optimization problems. *Neural Computing and Applications*, Vol. 28, No.7, pp. 1667-1682, 2017.

[30] Zandieh M., Khatami A. R. and Rahmati S. H., Flexible job shop scheduling under condition-based maintenance: improved version of imperialist competitive algorithm. *Applied Soft Computing*, Vol. 58, pp. 449-464, 2017.

[31] Al Khaled A. and Hosseini S., Fuzzy adaptive imperialist competitive algorithm for global optimization. *Neural Computing and Applications*, Vol. 26, No.4, pp. 813-825, 2015.

[32] Padfield G. D., *Helicopter flight dynamics: the theory and application of flying qualities and simulation modelling*. John Wiley & Sons, New York, 2008.