یک رهیافت جدید برای جایگاه یابی مسائل چند مدی با استفاده از الگوریتم بهبود یافته جهش قورباغه

ایمان سیدی[۱]، ملیحه مغفوری فرسنگی[۲]، محمد براتی[۱]، حسین نظام آبادی پور[۳]

۱-کارشناس ارشد مهندسی برق کنترل-دانشکده مهندسی -دانشگاه شهید باهنرکرمان

eiman.sayedi@gmail.com, mbaratiz@gmail.com

۲-دانشیار بخش مهندسی دانشگاه شهید باهنرکرمان

mmaghfoori@uk.ac.ir

۳-دانشیار بخش مهندسی دانشگاه شهید باهنرکرمان

nezam@uk.ac.ir

چکیده: مسأله جایگاه یکی از روش‌های مهم برای بهینه سازی مسایل چند مدی است. بیشتر روش‌های موجود در مسأله جایگاه نیاز به تعیین دقیقی از پارامترهای جایگاه به منظور عملکرد بهتر دارد. مشکل اصلی الگوریتم‌های ابتکاری در حل مسائل چند بعدی قدرت همگرایی آنها به یک جواب (عموماً بهینه فرا محلی) است. الگوریتم جهش قورباغه، از جمله الگوریتم‌های ابتکاری است که در سال‌های اخیر تا کنون نسخه‌ای از آن برای حل مسائل چند مدی ارائه نشده است. در این مقاله نسخه‌ای از این الگوریتم برای حل مسائل چند مدی با حفظ ساختارهای اساسی ارائه و با روش‌های مطرح مقایسه شده است. نتایج آزمایش‌ها روی توابع محک استاندارد توانایی الگوریتم پیشنهادی را تأیید می‌کند.

واژه‌های کلیدی: الگوریتم جهش قورباغه متحرک، بهینه‌های محلی، تابع قله-دره، روش‌های جایگاه یابی.

۱- مقدمه

در سال‌های اخیر الگوریتم‌های جستجوی تصادفی، از جمله الگوریتم‌های تکاملی، در حل بسیاری از مسائل دشوار بهینه سازی استفاده شده اند. شکل‌های استاندارد این الگوریتم‌ها معمولاً برای پیدا کردن یک بهینه (بهینه فرامحلیGlobal optimum) طراحی شده‌اند. با وجود این بسیاری از مسائل دنیای واقعی ذاتاً به صورت چند مدی هستند به طوری که چندین نقطه بهینه برای حل آنها وجود دارد. برای حل مسائل بهینه‌سازی با چندین بهینه محلی و فرامحلی ممکن است به مشخص کردن مکان تمامی بهینه-ها نیاز باشد. در گذشته تکنیک‌های زیادی برای پیدا کردن این مکان‌ها ارائه و توسعه یافته‌اند. مسأله جایگاه یکی از روش‌های مهم برای بهینه سازی مسائل چند مدی[۱] است. روش‌های جایگاه یابی حتی برای زمانی که هدف پیدا کردن یک نقطه بهینه فرامحلی است مؤثر واقع می‌شوند. از آنجایی که این روش‌ها به جستجوی موازی چندین بهینه در فضای مسأله می‌پردازند، لذا الگوریتم کمتر، در بهینه-های محلی گیر می‌کند. موفقیت الگوریتم‌های تکاملی در کاربردهای دنیای واقعی با استفاده از روش‌های جایگاه یابی همراه شده است [۲۱-۱]. همچنین برای حفظ کردن تنوع جمعیت در الگوریتم‌های تکاملی از روش‌های جایگاه

[۱]تاریخ ارسال مقاله: ۱۳۸۹/۱۱/۲۴

تاریخ پذیرش مقاله: ۱۳۹۰/۸/۱۸

نام نویسنده مسؤول: ملیحه مغفوری

نشانی نویسنده مسؤول: ایران -کرمان -دانشگاه شهید باهنر کرمان - بخش مهندسی برق

یابی استفاده شده است. برای معرفی این روش‌ها، می‌توان به روش ازدحامی[۲] [۵]، روش ازدحامی قطعی[۳] [۶]، روش تسهیم شایستگی[۴] [۷]، روش جایگاه یابی ترتیبی[۵] [۸]، روش انتخاب رقابت محدود شده[۶] [۹]، روش موازی سازی[۷] [۱۰]، روش خوشه‌بندی[۸] [۱۱]، روش پاکسازی[۹] [۱۲]، روش حفظ گونه[۱۰] [۱۳]، اشاره کرد. عموماً پارامترهای جایگاه برای تعیین کردن فاصله بین دو بهینه نزدیک به هم استفاده می‌شوند. مثال مشخصی از این موضوع می‌توان به σ_{share} در روش تسهیم شایستگی اشاره کرد [۷]. دیگر استفاده‌هایی که از پارامترهای جایگاه می‌شود، انتخاب CF در روش ازدحامی [۵]، مقدار w در روش انتخاب مسابقه‌ای محدود شده [۹] یا میانگین‌های دسته‌ها در روش خوشه بندی است [۱۱و۷]. دلایل ارائه روش جدید در مقاله برای حل مسائل چند مدی به شرح زیر است:

۱- دشواری در حفظ کردن جواب‌های پیدا شده در یک بار اجرای مسأله: برای مثال، روش ازدحامی دی جانگ نمی تواند تمامی قله‌ها را در تکرارهای رو به جلو حفظ کند [۶]. یک الگوریتم جایگاه خوب باید قادر به حفظ کردن و نظم دادن به جواب‌های پیدا شده باشد.

۲- در روش‌های مرسوم جایگاه می‌توان مشاهده کرد که عمل همبری بین دو فرد شایسته از جمعیت در دو جایگاه متفاوت ممکن است باعث تولید یک نوزاد با شایستگی پایین تر از والد شود[۳].

۳- بعضی از روش‌های جایگاه فقط برای مشخص کردن تمام بهینه فرامحلی طراحی شده اند و از تعیین بهینه-های محلی چشم پوشی شده است. برای مثال، در روش‌های جایگاه ترتیبی GA [۸]،پاکسازی [۱۲]، SCGA [۱۳]، NichePSO [۱۴]، SPSO [۱۶و۱۵] این امر اتفاق می‌افتد.

۴- پیچیدگی محاسبات بالا. بیشتر روش‌های مربوط به جایگاه به جز روش ازدحامی قطعی از این عیب رنج می-برند. با این حال، بسیاری از محققین بر این عقیده هستند که روش ازدحامی قطعی به تعداد ارزیابی بالایی در پیدا کردن جواب نیاز دارد [۱۷] و از توزیع جمعیتی مناسبی برای پیدا کردن جایگاه‌ها برخوردار نیست [۱۸]. هدف این مقاله توسعه الگوریتم جایگاه با ارائه موارد زیر است:

۱- نیازی به پارامترهای مربوط به جایگاه برای مشخص کردن فاصله بین دو نقطه بهینه یا تعدادی از بهینه‌ها نباشد.

۲- قادر به مشخص کردن چندین بهینه (بهینه محلی و فرامحلی) و حفظ کردن مکان بهینه جواب‌های پیدا شده تا پایان اجرای برنامه باشد.

۳- در مقایسه با روش‌هایی که همه نقاط جایگاه‌ها را ارائه می‌دهند دارای پیچیدگی محاسباتی کمتری باشد.

سازماندهی مقاله به این طریق است که در بخش دوم الگوریتم جهش قورباغه متحرک بیان می‌شود و در بخش سوم الگوریتم جدید پیشنهادی این مقاله برای جایگاه یابی مسائل چند مدی بیان می‌گردد. در بخش چهارم نتایج حاصل از پیاده سازی الگوریتم پیشنهادی بر روی توابع محک استاندارد، تحلیل نتایج و مقایسه با بعضی از روش‌های موجود در حیطه مسائل مربوط به جایگاه ارائه و در بخش پنجم نتیجه گیری آورده شده است.

۲- الگوریتم جهش قورباغه

الگوریتم جهش قورباغه[۱۱] یکی از الگوریتم‌های الهام گرفته از طبیعت است که توسط لنزی[۱۲] و یوسف[۱۳] توسعه داده شد. در این الگوریتم گروهی از قورباغه‌ها (مجموعه-ای از جواب‌ها) به چندین زیر مجموعه[۱۴] تقسیم می‌شوند، که هر قورباغه فرهنگ مختص به خود را دارد و می‌تواند از فرهنگ‌ها[۱۵] یا ایده‌های قورباغه‌های دیگر در طول روند تکامل استفاده کند [۲۷-۲۲]. مراحل اجرای این الگوریتم در زیر توضیح داده شده است:

۲-۱-تولید جمعیت اولیه

همانند تمامی الگوریتم‌های شهودی[۱۶] جمعیت اولیه (قورباغه‌ها) به صورت تصادفی[۱۷] بین بازه مسأله با توجه به رابطه (۱) تولید می‌شود:

$$x_i = x_i^l + rand \times (x_i^h - x_i^l) \qquad (۱)$$

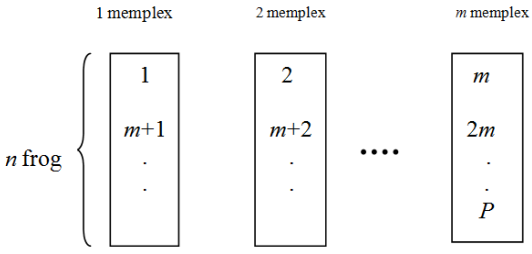
که در آن، $\mathbf{x_i}$ مقدار معادل هر عضو از جمعیت، x_i^l کران پایین متغیر x_i، x_i^h کران بالای متغیر x_i و rand یک مقدار تصادفی بین ۰ الی ۱ است.

هر قورباغه نمایانگر یک راه حل قابل قبول در مسأله بهینه سازی است که دارای یک مقدار شایستگی است. قورباغه‌ها بر اساس شایستگی شان به صورت نزولی مرتب می‌شوند و بر اساس روندی خاص به زیر مجموعه‌های مختلف تقسیم می‌شوند که در بخش ۲-۲ توضیح داده شده است.

۲-۲- دسته بندی قورباغه‌ها

فرض کنید که جمعیت اولیه با P قورباغه تولید شده است. با توجه به شکل(۱)، P قورباغه به m مجموعه تقسیم می‌شوند. روند تقسیم بندی قورباغه‌ها بدین صورت است که قورباغه اول به مجموعه اول، قورباغه دوم به مجموعه دوم و قورباغه m به مجموعه m ام و قورباغه m+1 به مجموعه اول تعلق دارند. این روند به صورت مشابه تا قورباغه آخر تکرار می‌شود. هر مجموعه m شامل n قورباغه است به طوری که رابطه (۲) ارضا می‌شود:

$$P = m \times n \qquad (۲)$$



شکل (۱): دسته بندی قورباغه‌ها به m زیر مجموعه

۲-۳- مراحل جستجوی محلی در الگوریتم SFL

در هر مجموعه موقعیت قورباغه i–ام، بر اساس اختلاف بین قورباغه بهتر (با بهترین شایستگی X_b) و قورباغه بدتر

(با بدترین شایستگی X_w) با استفاده از معادله زیر به دست می آید:

Position change: (۳)

$$D(i) = rand() \times (X_b - X_w)$$

که ()rand یک عدد تصادفی یکنواخت بین ۰ و ۱ است. موقعیت جدید قورباغه توسط رابطه (۴) به دست می‌آید که D_{max} ماکزیمم تغییراتی است که در موقعیت قورباغه می‌توان اعمال کرد.

New Position:

$$X_w = \text{current position } X_w + D_i \quad D_{max} \le D_i \le -D_{max} \qquad (۴)$$

اگر این تغییر موقعیت، قورباغه‌ای با شایستگی بهتر تولید کرد این قورباغه جایگزین قورباغه بدتر می‌شود. در غیر این صورت، قورباغه با بهترین شایستگی در کل جمعیت (بهینه فرامحلی[۱۸]) X_g جایگزین X_b در معادله (۳) شده و قورباغه جدیدی تولید می شود. اگر قورباغه‌ای با شایستگی بهتر تولید شود این قورباغه جایگزین قورباغه بدتر می‌شود. در غیر این صورت، قورباغه‌ای جدید به صورت تصادفی تولید و جایگزین بدترین قورباغه می-گردد.

۲-۴- به هم آمیختن جمعیت[۱۹]

پس از تکامل درونی چندین نسل، تمام مجموعه‌ها به هم آمیخته و بر اساس ارزش شایستگی آنها به صورت نزولی مرتب می‌شوند. سپس دوباره به چند زیر مجموعه تقسیم می‌گردند و روند تکامل در هر مجموعه تا زمانی که به معیار توقف برسد ادامه می‌یابد.

۲-۵- معایب الگوریتم SFL

در الگوریتم‌های تکاملی که بر پایه جمعیت تصادفی هستند، عمدتاً دو جنبه باید بررسی شود: ۱- کاوش؛ ۲- بهره وری. کاوش به جستجوی بهتر فضای جستجو می-پردازد و بهره وری قادر به پیدا کردن بهینه بهتر حول بهترین جواب است. نکته مهم برای داشتن عملکرد مناسب، ایجاد مصالحه بین کاوش و بهره وری است. الگوریتم SFL ممکن است در بعضی از توابع بهینه سازی قادر به پیداکردن بهینه فرامحلی نباشد و به رکود و افتادن

در بهینه محلی دچار شود. رکود الگوریتم SFL ممکن است به دو دلیل زیر باشد:

۱- در الگوریتم SFL، قورباغه‌ها متناسب با شایستگی شان به صورت نزولی مرتب و سپس همه جمعیت به چند زیر مجموعه تقسیم می‌گردند. این نوع دسته بندی نامتعادل است، زیرا عملکرد زیر مجموعه اول نسبت به زیر مجموعه‌های دیگر بهتر است. این توزیع نامتعادل قورباغه‌ها باعث ایجاد مشکلاتی در روند یادگیری می‌شود چون دیگر قورباغه‌های خوب در زیر مجموعه آخر وجود ندارند.

۲- چنانچه که ذکر شد در الگوریتم SFL موقعیت بدترین قورباغه با توجه به رابطه (۳) به‌دست می‌آید. قورباغه بدتر فقط از قورباغه بهتر تأثیر می‌پذیرد و قورباغه بهتر شانس کمتری برای یادگیری پیدا می‌کند مگر اینکه قورباغه‌های بهتر در طول تکامل تبدیل به قورباغه بدتر گردند. بنابراین این الگوریتم از مکانیزم یادگیری مناسبی برخوردار نیست.

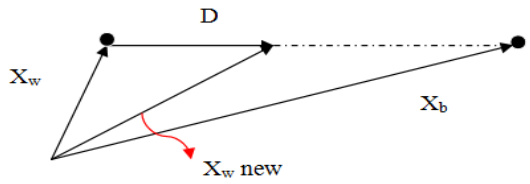
تاکنون روش‌هایی توسط محققان برای بهبود الگوریتم SFL ارائه شده است. بخش بعدی به بیان این روش‌ها اختصاص می‌یابد و سپس الگوریتم پیشنهادی این مقاله ارائه می‌شود.

۲-۶- مروری بر کارهای انجام شده در بهبود الگوریتم SFL

در سال ۲۰۰۷ ژن[۲۰] و همکارانش الگوریتم ممتیک جدیدی از ترکیب الگوریتم‌های SFL و PSO ارائه دادند [۲۴]. سپس در سال ۲۰۰۸ لی[۲۱] و همکارانش مقاله‌ای با عنوان الگوریتم جهش قورباغه آشوبناک ارائه دادند[۲۵]. در [۲۶] از ایده ممتیک در بهبود SFL استفاده شد. در سال ۲۰۰۸، هان[۲۲] پرش غیر دقیق قورباغه‌ها را با اضافه کردن عبارت عدم قطعیت به رابطه (۳) اصلاح کرد و گامی جدید در بهبود این الگوریتم برداشت و مکانیزم یادگیری را تا حدی تصحیح کرد [۲۷]. از آنجایی که روش پیشنهادی دراین مقاله بر اساس روش ارائه شده هان است، بنابراین، روش پیشنهادی هان در بخش بعد توضیح داده می‌شود.

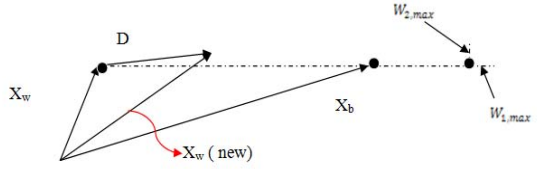
۲-۷- تصحیح کردن قانون پرش قورباغه با استفاده از روش پیشنهادی هان

پس از ارائه نسخه ابتدایی لنزی و یوسف که در قسمت قبل بیان گردید، در سال ۲۰۰۸ مقاله‌ای با عنوان «بهبود دادن الگوریتم جستجوی قورباغه برای به دست آوردن ضرائب کنترولر PID» توسط هان مطرح شد که به اصلاح قانون پرش قورباغه‌ها منجر گردید (رابطه (۳)). بدین گونه که به دلیل درک ناقص، قورباغه با شایستگی بدتر نمی‌تواند دقیقاً در موقعیت قورباغه بهتر قرار گیرد و این حرکت غیر دقیق، باعث می‌شود که قورباغه با شایستگی بدتر نتواند به طور مستقیم به موقعیت هدف پرش کند. این مورد در شکل (۲) برای یک مسأله دو بعدی نشان داده شده است.



شکل (۲): حرکت غیر دقیق قورباغه بدتر به سمت قورباغه بهتر

با توجه به این عدم قطعیت ، موقعیت قورباغه جدید با شایستگی کمتر لزوما به مسیر مستقیمی که بین موقعیت جدیدش و موقعیت بهترین قورباغه است، محدود نمی‌شود. بنابراین قورباغه بدتر می‌تواند از روی خط مستقیم پرش کند. این ایده به یک قانون پرش جدید منتهی می‌شود که در شکل (۳) نشان داده شده است.



شکل (۳): پرش قورباغه بدتر از روی خط مستقیم

قانون جدید پرش قورباغه پیشنهادی‌هـان بـه صـورت زیر است:

$$D = r \times c \times (X_b - X_w) + W \quad (۵)$$

که مقدار W از رابطه (۶) تعیین می‌گردد:

$$W = [r_1 W_{1,max}, r_2 W_{2,max}, \dots, r_s W_{s,max}]^T \quad (۶)$$

که r یک مقدار تصـادفی بـین ۰ و ۱ اسـت و c ثـابتی است که مقـداری بـین ۱ و ۲ دارد و $r_i\ (1 \le i \le S)$ یـک عدد تصادفی بین ۱- و ۱ می‌باشـد. $W_{i,max}\ (1 \le i \le S)$ بیشترین درک اجازه داده شده و عـدم قطعیـت در i-امـین بعد از فضای جستجوست که به صورت رابطه زیر تعریـف می شود:

$$W \quad (۷)$$

W_{max} برای هر متغیر مستقل S، ۰.۱ تا ۰.۲ بازه مسـأله در نظر گرفته می‌شود. در هر حـال، اگـر $\|W_{max}\|$خیلـی بزرگ باشد، قانون جهش قورباغه مشخصه هدایتی خود را از دست می‌دهد.

سپس مقدار پرش جدید در رابطه (۵) به مقدار بدترین قورباغه طبق رابطه (۸) اضافه می‌شود:

$$X_w(new) = \begin{cases} X_w + D & \text{if} \|D\| \le D_{max} \\ X_w + \frac{D}{\sqrt{D^T D}} & \text{if} \|D\| > D_{max} \end{cases} \quad (۸)$$

بقیه الگوریتم همانند SFL استاندارد است. در قسمت بعد به بیان الگوریتم پیشنهادی این مقاله پرداخته می‌شود.

۳- معرفی الگوریتم جدیـد پیشـنهادی ایـن مقاله برای جایگاه یابی مسـائل چنـد مـدی (NSFL [۲۳])

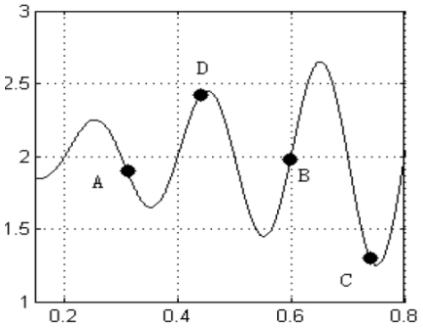
در ایـن بخـش ابتـدا راهکـاری بـرای دسـته بنـدی قورباغه‌ها برای رفع نخستین عیب الگـوریتم SFL (بخـش ۲-۵)ارائه می شود. سپس روش‌هان توسعه داده می شـود و راه حلی برای رفع دومین عیـب الگـوریتم SFL پیشـنهاد می شود.

۳-۱- دسته بندی قورباغه‌ها با استفاده از تـابع قله-دره[۲۴]

بـه طـور کلـی، مشـخص کـردن شـعاع همگرایـی در روش‌های به دسـت آوردن جایگـاه بسـیار سـخت اسـت. بنابراین، اگر دو نقطه قله مربوط به تابع چند بعـدی متعلـق به فضای جستجو را داشته باشیم، دیگر به شعاع همگرایـی نیازی نیسـت. ایـن روش در ابتـدا در سـال ۱۹۹۹ توسـط ارسـم[۲۵] معرفـی گردیـد [۲۱]. روش پیشـنهادی ارسـم در شکل (۴) نشان داده شده است.

```
Hill_valley(ip,iq,samples)
minfit=min(fitness(ip), fitness(iq))
for j=1 to samples.length
 Calculate point iinterior on the line between the pints
                    ip and iq
                If(minfit>fitness(iinterior))
                    return 1
                  end if
                 end for
                 return 0
```

شکل (۴): شبه کد تابع قله-دره



شکل (۵): ایده جدا سازی قله‌ها به وسیله تابع قله-دره

همـان گونـه ذکـر شـد، در الگـوریتم SFL جمعیـت متناسب با شایستگی مرتب می شـود. در روش پیشـنهادی، ابتدا دو قورباغه A وB در نظـر گرفتـه، سـپس قورباغـه‌ای تصادفی بین A وB به نام D با توجـه بـه رابطـه (۹)تولیـد می‌شود.

$$D = A + rand() \times (B - A) \quad (۹)$$

اگر مینیمم شایستگی A وB از شایستگی D بیشتر شـد می‌توان این نتیجه را گرفت که A وB در قله‌هـای متفـاوت هستند، در غیر این صورت، قورباغـه B ام و قورباغـه A ام مربوط به یک قله هستند و در یک زیر مجموعه قرار مـی-

گیرند. ممکن است قورباغه D در موقعیتی همانند شکل (۵) باشد.

همان طور که در شکل (۵)دیده می‌شود، مینیمم شایستگی A وB از D کمتر است و همچنان A وB در قله‌های متفاوت هستند. در صورت بروز چنین حالتی باید تعداد عددهای تصادفی بین A وB را زیاد در نظر گرفت تا شایستگی عدد تولید شده D از A وB کمتر شود. در اینجا این نکته قابل اهمیت است که در الگوریتم جهش قورباغه متحرک با توجه به مرتب کردن جمعیت قورباغه‌ها متناسب با شایستگی همیشه شایستگی قورباغه B از A کمتر است (مسأله بیشینه سازی). پس از دسته بندی قورباغه‌ها با توجه به جایگاه‌ها، بر روی هر کدام از مجموعه‌ها یک جستجوی محلی انجام می‌شود. در قسمت بعد با ارائه روشی جدید به بهبود الگوریتم‌هان پرداخته می‌شود که الگوریتم پیشنهادی بهتر از الگوریتم ارائه شده توسط‌هان عمل می نماید.

۳-۲- بهبود روش پیشنهادی هان

هرچند روش پیشنهادی هان تا حدودی الگوریتم SFL را بهبود بخشید اما الگوریتم هنوز از مکانیزم یادگیری چندان موثری برخوردار نیست، از آنجایی که پرش مد نظر از اختلاف بین قورباغه‌های بهتر و بدتر حاصل می‌شود و از یادگیری بقیه قورباغه‌ها به نحوی جلوگیری می‌کند، بنابراین، در این مقاله، قانون پرش به صورت زیر تصحیح می‌شود؛ بدین گونه که پرش مورد نظر از اختلاف تک تک اعضای قورباغه‌ها در هر زیر مجموعه و بدترین عضو از همان مجموعه، به دست می‌آید:

$$D(i) = r \times c \times (X_i - X_{worst}) + W \qquad (۱۰)$$

که در آن i=1:P و X هر عضو جمعیت یا قورباغه در هر مجموعه و X_{worst} بدترین عضو جمعیت در هر مجموعه است، r یک مقدار تصادفی بین ۰ و ۱ است و c ثابتی است که مقداری بین ۱ و ۲ دارد.

سپس موقعیت جدید قورباغه توسط رابطه زیر به دست می‌آید:

$$X_{new\ worst} = current\ position \quad X_w + D \qquad (۱۱)$$

اگر این تغییر موقعیت، قورباغه‌ای با شایستگی بهتر تولید کرد این قورباغه جایگزین قورباغه بدتر می‌شود در غیر این صورت X_g جایگزین X_i می شود. X_g بیانگر بهترین قورباغه در کل مجموعه‌هاست. اگر قورباغه بهتری تولید شد جایگزین بدترین قورباغه می‌گردد. در غیر این صورت، قورباغه‌ای به صورت تصادفی تولید و با توجه به رابطه زیر جایگزین بدترین قورباغه می‌گردد.

$$X_{new\ worst} = X_{Low} + rand() \times (X_{hi} - X_{Low}) \qquad (۱۲)$$

که rand() یک عدد یکنواخت بین ۰ و ۱ است.

اما الگوریتم پس از مدتی دچار رکود می‌گردد، چون یادگیری قورباغه بدتر از بهترین‌ها متوقف می‌شود. بنابراین، نیاز به حرکت دوباره قورباغه‌ها در جهت دیگر در فضای جستجو است. بنابراین، پس از آنکه یادگیری قورباغه بدتر از بهترین قورباغه در کل فضای جستجو پایان یافت از رابطه (۱۳) برای پرش قورباغه استفاده می‌شود.

$$D(i) = r \times c \times (X_g - X_i) + W \qquad (۱۳)$$

حال این پرش طبق رابطه (۱۴) به X_i اضافه می‌گردد و اگر شایستگی قورباغه تولید شده از i–امین قورباغه بهتر شد جایگزین قورباغه‌ای که دچار رکود شده است.

$$X_i(new) = \begin{cases} X_i + D & \text{if } \|D\| \le D_{max} \\ X_i + \frac{D}{\sqrt{D^T D}} & \text{if } \|D\| \ge D_{max} \end{cases} \qquad (۱۴)$$

شایان ذکر است که بقیه الگوریتم همانند SFL استاندارد است. برتری روش پیشنهادی نسبت به الگوریتم هان در پیوست مقاله نشان داده شده است.

در بخش بعد با پیاده سازی الگوریتم پیشنهادی بر روی توابع محک استاندارد، توانایی آن در پیدا کردن مکان تمامی جایگاه‌ها و رسیدن به جواب بهینه سنجیده می‌شود و با روش‌های مطرح در این زمینه مقایسه می شود.

۴- آزمایش‌ها ونتایج و مقایسه با سایر روش‌ها

در این قسمت نتایج حاصل از پیاده سازی الگوریتم پیشنهادی باچند روش دیگر ارائه شده در مقالات گوناگون، برای حل تعدادی از مسائل چند مدی که در آنها هدف یافتن همه بهینه‌های محلی و فرامحلی است، ارائه شده است. توابع F1 الی F5 توسط گلدبرگ[۲۶] و ریچاردسون[۲۷] معرفی شده‌اند. این توابع برای آزمودن الگوریتم جایگاه‌یابی توسط محققان مختلف استفاده شده‌اند [۱۷و۸و۳]. این توابع توسط روابط (۱۵) الی (۲۱) تعریف می‌شوند. توابع F1 الی F4 در یک بعد تعریف شده و در بازه [۰ ۱] دارای پنج نقطه بهینه است. تابع F1 و F2 دارای پنج نقطه بهینه در مکان‌های ۰.۱،۰.۳،۰.۵،۰.۷، و ۰.۹ هستند. برای تابع F1 مقدار بهینه‌ها برابر ۱ است، ولی برای تابع F2 مقدار بهینه‌ها با یکدیگر متفاوت است. اما مکان بهینه‌ها در توابع F3 و F4 ۰.۹۳۰ ، ۰.۶۸۱ ، ۰.۴۵۰ ،۰.۲۴۶ و ۰.۰۷۹ است. برای تابع F3 مقدار بهینه‌ها برابر ۱، ولی برای تابع F4 مقدار بهینه‌ها با یکدیگر متفاوت است. تابعF5 به تابع هیملبلائو[۲۸] مشهور است، در فضای دو بعدی تعریف می‌شود و در ناحیه تعریف شده چهار بهینه در نقاط (۳و۲)، (۳.۷۷۹۰- و۳.۲۸۳۴-)، (۲.۸۰۶۴- و۳.۱۳۰۴) و (۳.۵۸۴۰ و ۱.۸۴۸۰-) دارد.

$$F_1(x) = \sin^6(5\pi x) \quad 0 < x < 1 \tag{۱۵}$$

$$F_2(x) = e^{2\log(2)\times\left(\frac{x-0.1}{0.8}\right)^2} \sin^6(5\pi x) \quad 0 < x < 1 \tag{۱۶}$$

$$F_3(x) = \sin^6\left(5\pi(x^{\frac{3}{4}} - 0.05)\right) \quad 0 < x < 1 \tag{۱۷}$$

$$F_4(x) = e^{2\log(2)\times\left(\frac{x-0.08}{0.864}\right)^2} \sin^6\left(5\pi\left(x^{\frac{3}{4}} - 0.05\right)\right) \quad 0 < x < 1 \tag{۱۸}$$

$$F_5(x_1, x_2) = 200 - (x_1^2 + x_2 - 11)^2 - (x_1 + x_2^2 - 7)^2 \quad -5 < x_1, x_2 < 5 \tag{۱۹}$$

تابع F6 معروف به تابع فریبنده، توسط عمرانی[۲۹] [۱۹] و عالمی[۳۰] [۲۰] به کار گرفته شده است. این تابع در یک بعد و در فاصله [۶ ۰] تعریف می‌شود و دارای دو بهینه مرزی است. این تابع دارای یک بهینه محلی در X=۳ با مقدار ۰.۶ و دو بهینه فرامحلی با مقدار ۱ در X=۰ و X=۶ است.

تابع F7 توسط مایکولز[۳۱] معرفی و توسط ژانگ استفاده شده است [۲۸]. این تابع در یک بعد در فاصله [۱۰ ۱۰-] تعریف می‌شود. این تابع ۱۹ بهینه دارد که ۳ تا از آنها بهینه‌های فرامحلی هستند. در این تابع تعداد بهینه‌ها افزایش یافته است تا الگوریتم پیشنهادی جهت یافتن تعداد بیشتر جایگاه‌ها محک زده شود. توابع F6 و F7 توسط روابط (۲۰) و (۲۱) تعریف می شوند.

$$F_6(x) = \begin{cases} -x + 1 & \text{if } 0 \le x < 1 \\ 0.4(x - 1) & \text{if } 1 \le x < 2 \\ 0.24x - 0.08 & \text{if } 2 \le x < 3 \\ -0.24x + 1.36 & \text{if } 3 \le x < 4 \\ -0.4x + 2 & \text{if } 4 \le x < 5 \\ x - 5 & \text{if } 5 \le x \le 6 \end{cases} \quad 0 \le x \le 6 \tag{۲۰}$$

$$F_7(x) = \sum_{i=1}^{5} i\cos[(i+1)x + i] \quad -10 \le x \le 10 \tag{۲۱}$$

شایان ذکر است که جمعیت اولیه برای توابع F1 تا F6 برابر ۵۰ و برای تابع F7 برابر ۱۵۰ در نظر گرفته شده است. مقدار D_{max} هم برای تمامی توابع برابر با بی نهایت در نظر گرفته شده است.

درصد موفقیت برای حل مسائل چند مدی به صورت نسبت تعداد بهینه‌های یافته شده توسط الگوریتم به کل بهینه‌های موجود در فضای مسأله در همه تکرارها تعریف می‌شود [۱]. جدول (۱) درصد موفقیت در یافتن نقاط بهینه طی ۳۰ بار اجرای مستقل الگوریتم را با تعدادی از الگوریتم‌های دیگر مقایسه می‌کند. جدول (۲) خطای میانگین الگوریتم پیشنهادی را نشان می‌دهد. برای هر جواب، خطای میانگین به صورت میانگین فاصله اقلیدسی جواب پیدا شده توسط الگوریتم تا مکان بهینه واقعی تعریف می‌شود. این خطا طبق رابطه (۲۲) محاسبه می

شود. در این رابطه N_i مکان واقعی بهینه i ام و $\hat{N}_i$ مکان بهینه یافته شده توسط الگوریتم و M تعداد بهینه‌های یافت شده است [۲].

$$\text{Mean Erorr} = \frac{\sum_{i=1}^{M}\|N_i - \hat{N}_i\|}{M} \quad (۲۲)$$

نتایج جدول (۲) بیانگر آن است که کارایی الگوریتم پیشنهادی در مقایسه با الگوریتم‌های دیگر بهتر بوده است. به عبارت دیگر، الگوریتم پیشنهادی توانسته است در هر بار اجرای برنامه جایگاه نقاط بهینه را پیدا کند.

جدول (۱): درصد موفقیت الگوریتم پیشنهادی

تابع	NISFL	[۲]NGSAV	[۱۷]GCPSO	Deterministic [۱۷]Crowding	[۲]Fitness sharing
F1	۱۰۰	۱۰۰	۱۰۰	۱۰۰	82.7
F2	۱۰۰	۹۹,۳	93	93	68
F3	۱۰۰	۱۰۰	۱۰۰	90	80
F4	۱۰۰	۹۹,۳	93	90	66.7
F5	۱۰۰	۱۰۰	۱۰۰	90	53.3
F6	۱۰۰	۹۸,۸	-	-	63.7
F7	۱۰۰	۱۰۰	-	-	32.3

جدول (۲): بهترین جواب دیده شده در ۳۰ بار اجرای الگوریتم

تابع	Best_Niche					Mean Error
F1	0.100	0.2999	0.4995	0.700	0.9000	3.4e -5
F2	0.100	0.2994	0.4983	0.6983	0.8976	1.1e -3
F3	0.0796	0.2466	0.450	0.6810	0.9338	1.6e -3
F4	0.0796	0.2462	0.4494	0.6791	0.9303	3.9e -4
F5	3.00 1.9993	-3.7821 -3.2783	3.5843 1.8482	-2.8056 2.1276		2.7e -3
F6	0.000	2.999	6.00			3.8e -3
F7	-					2.2e -2

۵- نتیجه گیری

تاکنون برای حل مسائل چند مدی به وسیله الگوریتم جهش قورباغه متحرک روشی ارائه نشده است. الگوریتم جهش قورباغه، از جمله الگوریتم‌های تکاملی است که می‌تواند در حل مسائل چند مدی استفاده شود. در بهینه سازی مسائل چند مدی، همیشه این سوال مطرح بوده است که یافتن تمامی نقاط بهینه بسیار دشوار است. همان طور که بیان شد، استفاده از پارامترهای جایگاه، از جمله پارامترهای شعاع همگرایی عملکرد بدی در مسائل مربوط به جایگاه دارد. با روش پیشنهادی علاوه بر بهبود الگوریتم SFL اقتباش شده از مقاله هان، می‌توان به روش تابع فله-دره قله‌ها را تشخیص داد و جمعیت را متناسب با قله‌ها در مجموعه‌های جداگانه دسته بندی کرد. برای ارزیابی توانایی روش پیشنهادی، نتایج بهینه سازی چند تابع محک استاندارد با روش‌های مطرح در این زمینه مقایسه شده است. نتایج آزمایش‌ها بهبود کارایی روش پیشنهادی را تأیید می‌کند.

مراجع

[۱] سجاد یزدانی، حسین نظام آبادی پور، (۱۳۸۸). «حل مسائل چند مدی با استفاده از الگوریتم جستجوی گرانشی»، در پانزدهمین کنفرانس سالانه انجمن کامپیوتر ایران، مجموعه مقالات رایانش نرم.

[۲] سجاد یزدانی، حسین نظام آبادی پور ، (۱۳۸۹). « یک راه حل گرانشی جدید برای بهینه یابی مسائل چند

مدی»، در هجدهمین کنفرانس مهندسی برق ایران ،مجموعه مقالات کامپیوتر.

[3] Mahfoud, S. W., "Niching methods for genetic algorithms",Ph.D. dissertation, Urbana, IL, USA, 1995.

[4] Horn, J., Nafpliotis, N., and Goldberg, D. E., "A Niched Pareto Genetic Algorithm for Multiobjective Optimization", in Proc. of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, vol. 1. Piscataway, New Jersey: IEEE Service Center, pp. 82–87 ,1994.

[5] De Jong, K. A., "An analysis of the behavior of a class of genetic adaptive systems", Ph.D. dissertation, University of Michigan, 1975.

[6] Mahfoud, S. W., "Crowding and preselection revisited", in Parallel problem solving from nature 2, R. M¨anner and B. Manderick, Eds. Amsterdam: North-Holland, pp. 27–36, 1992.

[7] Goldberg D. E., Richardson, J., "Genetic algorithms with sharing for multimodal function optimization", in Proc. of the Second International Conference on Genetic Algorithms, J. Grefenstette, Ed., pp. 41–49,1987.

[8] Beasley, D., Bull, D. R., Martin, R. R., "A sequential
niche technique for multimodal function optimization", Evolutionary Computation, vol. 1, no. 2, pp. 101–125, 1993.

[9] Harik, G. R., "Finding multimodal solutions using restricted tournament selection", in Proc. of the Sixth International Conference on Genetic Algorithms, L. Eshelman, Ed. San Francisco, CA: Morgan Kaufmann, pp. 24–31,1995.

[10] Bessaou, M., P´etrowski, A., and Siarry, P., "Island model cooperating with speciation for multimodal optimization", in Parallel Problem Solving from Nature - PPSN VI 6th International Conference, H.-P. S. et al., Ed. Paris, France: Springer Verlag, 16-20,2000.

[11] Yin, X., Germay, N., "A fast genetic algorithm with sharing scheme using cluster analysis methods in multi-modal function optimization", in the International Conference on Artificial Neural Networks and Genetic Algorithms, pp. 450–457, 1993.

[12] P´etrowski, A., "A clearing procedure as a niching method for genetic algorithms", in Proc. of the 3rd IEEE International Conference on Evolutionary Computation, pp. 798–803, 1996.

[13] Li, J.P., Balazs, M. E., Parks, G. T., Clarkson, P. J., "A species conserving genetic algorithm for multimodal function optimization", Evol. Comput., vol. 10, no. 3, pp. 207–234, 2002.

[14] Brits, A. E. R., van den Bergh, F., "A niching particle swarm optimizer", in Proc. of the 4th Asia-Pacific Conference on Simulated Evolution and Learning 2002(SEAL 2002), pp. 692–696, 2002.

[15] Li, X., "Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization", in Proc. of Genetic and Evolutionary Computation Conference 2004(LNCS3102), K. Deb, Ed, pp. 105–116, 2004.

[16] Parrott, D., Li, X., "Locating and tracking multiple dynamic optima by a particle swarm model using speciation", IEEE Trans. on Evol. Comput., vol. 10, no. 4, pp. 440–458, August 2006.

[17] Brits, R., Negelbrecht, A., van den Bergh, F., "Locating multiple optima using particle swarm optimization", Applied Mathematics and Computation, vol. 189, pp. 1859–1883, 2007.

[18] Sareni, B., Krahenbuhl, L., "Fitness sharing and niching methods revisited", IEEE Trans. on Evol. Comput., vol. 2, no. 3, pp. 97 – 106, Sep. 1998.

[19] El Imrani, A., Bouroumi, A., Zine El Abidine, H., Limouri, M., Essaid, A., "A fuzzy clustering-based niching approach to multimodal function optimization", Cognitive System Research, vol 1, pp 119-133, 2000.

[20] Alami, J., El Imrani, A., Bouroumi, A., "Multi-population cultural algorithm using fuzzy clustering", Applied Soft Computing ,vol. 7 ,pp. 506–519,2007.

[21] R.K. Ursem, "Multinational evolutionary algorithms", in proceedings of congress of Evolutionary Computation (CEC-99), vol. 3, Washington, pp 1633–1640, 1999.

[22] Eusuff, M., Lansey, K., Pasha, F., "Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization", Engineering Optimization, vol. 38, No. 2, pp. 129-154, 2006.

[23] Eusuff, M., Lansey, K., "Optimization of water distribution network design using the shuffled frog leaping algorithm", Journal of Water Resources Planning and Management, vol. 129, no. 2, pp. 210–25, 2003.

[24] Zhen, Z., "A Novel Memetic Algorithm for Global Optimization Based on PSO and SFLA", ISICA, LNCS 4683, pp. 126–135, 2007.

[25] Y. Li and et al, “The Chaos-based Shuffled Frog Leaping Algorithm and Its Application,” Fourth International Conference on Natural Computation,pp.481-485,2008.

[26] lttipong, P., " solving non-linear continous mathematical using shuffled frog leaping and memetic algorithm", Department of Computer Science and information technology, faculty of science, Naresuan university,2008.

[27] Huynh, T.H., "A Modified Shuffled Frog Leaping Algorithm for Optimal Tuning of Multivariable PID Controllers", ICIT IEEE Industrial Technology ,2008 .

[28] Zhang, J., "A novel adaptive sequential niche technique for multimodal function optimization",,Neuro Computing,vol. 69, pp 2396-2401,2006.

پیوست ۱: نتایج روش هان با روش پیشنهادی روی توابع محک استاندارد در ادامه مقایسه شده است. نتایج آزمایش‌ها بهبود توانایی الگوریتم پیشنهادی و ضعف الگوریتم هان را تایید می‌کند. این نتایج نشان می دهد که الگوریتم هان از مکانیزم یادگیری مناسبی برخوردار نیست.

جدول پ- ۱-توابع محک کمینه شونده چند مد و با بعد متغیر

Test Function	S
$F_1(X)=\sum_{i=1}^{n} x_i^2$	$[-100,100]^n$
$F_2(X)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	$[-10,10]^n$
$F_3(X)=\sum_{i=1}^{n}(\sum_{j=1}^{i} x_j)^2$	$[-100,100]^n$
$F_4(X)=\max_i\{\lvert x_i\rvert,1\le i\le n\}$	$[-100,100]^n$
$F_5(X)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+(x_i-1)^2\right]$	$[-30,30]^n$
$F_6(X)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	$[-100,100]^n$
$F_7(X)=\sum_{i=1}^{n} ix_i^4+random[0,1)$	$[-1.28,1.28]^n$

جدول پ- ۲- توابع چند مدی که با افزایش بعد، تعداد بهینه‌های آن افزایش می یابد.

Test Function	S
$F_8(X)=\sum_{i=1}^{n} -x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	$[-500,500]^n$
$F_9(X)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	$[-5.12,5.12]^n$
$F_{10}(X)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	$[-32,32]^n$
$F_{11}(X)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$[-600,600]^n$
$F_{12}(x)=\frac{\pi}{n}\{10\sin(\pi(x)_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi(x)_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n}u(x_i,10,100,4)$ $y_i=1+\frac{x_i+1}{4}$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0 & -a<x_i<a\\ k(-x_i-a)^m & x_i<-a\end{cases}$	$[-50,50]^n$

$$F_{13}(X) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4) \qquad [-50,50]^n$$

جدول پ- (۳): نتاج پیاده سازی الگوریتم‌هان و الگوریتم پیشنهادی روی توابع جداول (پ- ۱ و پ- ۲) میانگین بهترین جواب دیده شده(Mean_best)، بهترین جواب دیده شده(Best_answer)،بدترین جواب دیده شده(Worst_answer) ، انحراف معیار (SD)، تعداد ارزیابی(evaluate) و تعداد تکرار ۵۰۰ با بعد ۳۰ با ۱۰ بار اجرای مستقل الگوریتم.

Function	الگوریتم‌هان					الگوریتم پیشنهادی				
	Mean_best	Best_answer	Worst_answer	SD	evaluate	Mean_best	Best_answer	Worst_answer	SD	evaluate
F1	5.9526e-05	2.6461e-۰5	7.2310e-۰5	3.177e-5	50000	0	0	0	0	35000
F2	0.0045	0.0024	0.0075	0.0018	50000	2.3207e-213	1.8278e-222	2.3188e-212	0	50000
F3	4.4987e-15	1.0080e-017	3.1450e-014	9.5e-15	50000	0	0	0	0	45000
F4	6.0352e-14	2.2777e-014	1.5774e-013	3.78e-14	50000	5.67393e-237	1.30685e-241	4.55618e-236	0	50000
F5	92.7849	26.2614	419.5713	117.7615	50000	0	0	0	0	4000
F6	0	0	0	0	8500	0	0	0	0	600
F7	0.0235	0.0038	0.1415	0.0419	50000	2.4646e-04	7.1464e-05	6.82636e-04	2.2e-04	50000
F8	-7.6244e+03	-9.2729e+03	-7.0617e+003	688.4169	50000	-1.25694e+04	-1.25694e+04	-1.25694e+04	0	1000
F9	46.1882	16.9143	70.6420	19.4128	50000	0	0	0	0	4000
F10	0.8419	0.0035	1.6484	0.7565	50000	0	0	0	0	35000
F11	0.0441	1.4482e-004	0.1058	0.0360	23000	0	0	0	0	3000
F12	1.0121	1.8416e-006	2.0943	1.6030	50000	1.57054e-32	1.57054e-32	1.57054e-32	2.88e-48	4000
F13	9.3631e-16	1.3934e-016	1.2814e-015	5.13e-16	50000	1.34978e-32	1.34978e-32	1.34978e-32	2.88e-48	4000

زیرنویس‌ها

[1] -Multimodal
[2] -Crowding
[3]-Deterministic crowding
[4] -Fitness sharing
[5] -Sequential niche
[6] - Restricted tournament selection
[7] - parallelization
[8] -Clustering
[9] -Clearing
[10] -Speciation
[11] Shuffled frog leaping (SFL)
[12] Lansey
[13] -Eussuf
[14] - Memplex
[15] -Culture
[16]-Heuristic

[17]-random
[18] Global best
[19] Shuffled of population
[20] - Zhen
[21] - Li
[22] - Huynh
[23] New shuffled frog leaping (NSFL)
[24] -Hill- valley function
[25] -Ursem
[26] - Goldberg
[27] - Richardson
28- Himmelblau
[29] - Imrani
[30] -Alami
[31] - Michols