بهبود الگوریتم رقابتی کلونی استعمارگران با استفاده از عملگر یادگیری استعمارگران وکاربرد آن در حل مساله فروشنده دوره‌گرد

حسن حاله[۱]، دانیال اسماعیلی علی آبادی[۲،*]

[۱] استادیار، دانشگاه آزاد اسلامی، واحد قزوین، گروه مهندسی صنایع، قزوین، ایران

[۲] دانشجو دکتری، دانشگاه سابانجی، گروه مهندسی صنایع، استانبول، ترکیه (عهده‌دار مکاتبات)

تاریخ دریافت: فروردین۱۳۹۴، اصلاحیه: خرداد ۱۳۹۴، پذیرش: مرداد ۱۳۹۴

چکیده

در این مقاله سعی شده تا با بهبود الگوریتم رقابتی کلونی استعمارگران (ICA) در قالب مساله فروشنده دوره گرد (TSP)، پاسخ مسأله بهبود یابد و میزان میزان این بهبود مورد بررسی و تحقیق قرار گیرد. با افزودن عملگر یادگیری، استعمارگران از کلونی‌هایی که تا حد مطلوبی پیشرفت حاصل نموده‌اند، خواصی را در جهت‌هایی یاد می‌گیرند که باعث بهبود پاسخ مسأله شود. با توجه به یادگیری کنترل شده استعمارگران از کلونی‌های مطلوب میزان بهبود پاسخ بیشتر از زمانی است که کلونی‌ها بصورت غیر کنترل شده از استعمارگران خود خواصی را برداشت می نمایند. در این مقاله سعی شده تا عملکرد عملگر یادگیری را با مثال‌های متعدد از TSPLIB نمایش دهد و بیان نماید که الگوریتم رقابتی کلونی استعمارگران با عملگر یادگیری استعمارگران نتایج بهتری را هم در کیفیت پاسخ و هم در زمان حل نسبت به زمانی که این عملگر استفاده نشود، ارائه می دهد.

کلمات کلیدی: مسأله فروشنده دوره گرد، الگوریتم کلونی استعمارگران، الگوریتم فرا ابتکاری، کتابخانه مسائل فروشنده دوره گرد.

۱- مقدمه

مسئله فروشنده دوره گرد با فرض طول مسیر به صورت اقلیدسی یکی از مسائل کلاسیک بهینه‌سازی ترکیباتی[۱] است. مسئله شامل یافتن کوتاهترین مسیر حرکتی است که از N شهر بگذرد و در آن فاصلهٔ بین شهرها اقلیدسی باشد. بعلت NP-Complete بودن مسئله TSP[۲] برای سه دهه موضوع تحقیقات گسترده ای بوده است. پیچیدگی زیاد این مسئله سبب می‌شود که به علت رشد غیرخطی زمان حل برای TSPهایی با اندازه بیش از ۱۰۰۰ شهر قابلیت حل به صورت بهینه وجود نداشته باشد [۵]. الگوریتم‌های ابتکاری و فراابتکاری رویکردهایی هستند که قادر به تولید دورهای نزدیک بهینه در زمان قابل قبول می‌باشند.

فرا ابتکاری‌ها روش‌هایی هستند که از لحاظ آماری یا تجربی تضمین می‌کنند که جواب‌های خوبی را بیابند، اما هیچ اثبات ریاضی برای کارایی آنها هنوز وجود ندارد[۵]. الگوریتم‌های بهینه‌سازی الهام گرفته از طبیعت (فراابتکاری) به عنوان روشهای هوشمند بهینه‌سازی در کنار روش‌های کلاسیک موفقیت خوبی از خود نشان داده‌اند. از جمله این روش‌ها می‌توان به الگوریتم‌های ژنتیک[۳] (الهام گرفته از تکامل بیولوژیکی انسان و سایر موجودات)، بهینه سازی کلونی مورچه‌ها[۴] (بر مبنای حرکت بهینه مورچه‌ها) و روش بازپخت شبیه سازی شده[۵] (با الهام‌گیری از فرایند تبرید فلزات) اشاره نمود.

این روش‌ها در حل بسیاری از مسائل بهینه سازی در حوزه‌های مختلفی چون تعیین مسیر بهینه عامل‌های خودکار[۶]، طراحی بهینه کنترل کننده برای پروس‌های صنعتی، حل مسائل عمده مهندسی صنایع همانند طراحی چیدمان[۷] بهینه برای واحدهای صنعتی، حل مسائل صف[۸] و نیز در طراحی عامل‌های هوشمند استفاده شده‌اند.

الگوریتم‌های فراابتکاری معرفی شده، به طور عمده الهام گرفته از فرایندهای طبیعی می‌باشند و در ارائه این الگوریتم‌ها به سایر نمودهای تکامل انسانی توجهی نشده است. الگوریتم رقابت استعماری[۹] ، که در این مقاله به کار گرفته شده است، از یک پدیده اجتماعی – انسانی الهام گرفته شده است. بطور ویژه این الگوریتم به فرایند استعمار، به عنوان مرحله‌ای از تکامل

* Danial@gohararya.com

1.Combinatorial Optimization Problem

2 .Travel Salesman Problem

3 .Genetic Algorithm

4 . Ant Colony Algorithm

5. Simulated Annealing

6 .Optimal Path for Automated Agents

7. Layout Design Problem

8 .Queue Problem

9. Imperialist Competitive Algorithm

اجتماعی - سیاسی بشر نگریسته و با مد لسازی ریاضی این پدیده تاریخی، از آن به عنوان منشأ الهام یک الگوریتم قدرتمند در زمینه بهینه‌سازی بهره می‌گیرد. تاکنون از این الگوریتم در مقالات متعددی جهت بهینه‌سازی استفاده شده است بطور مثال استفاده از الگوریتم رقابت استعمارگران در مباحث کنترل و رباتیک که می توان به طراحی کنترل PID مبتنی بر الگوریتم رقابت استعمارگران در کنترل فرایندهای صنعتی و شیمیایی [۱و۲] و کنترل خطی موتور القایی [۶] اشاره نمود. همچنین در مباحث هوش مصنوعی و آنالیز تصویر نیز در تطبیق الگوهای بصری در تصاویر و تشخیص مکان الگو در تصاویر بزرگتر بهره برداری شده است[۴].

همچنین الگوریتم رقابت کلونی استعمارگران برای تطبیق با مسائل بزرگ بهبود داده شد و در مساله بهبود محدود[1] نیز بکار برده شد تا کارایی آن به اثبات برسد[۱۰]. با اینکه این الگوریتم اخیراً بوجود آمده ولی کاربرد آن در بازه گسترده‌ای از مسائل نشان از کارایی آن دارد بطور مثال می‌توان به آرایه‌های آنتنهای انطباقی اشاره نمود[۹]. در مباحث برنامه ریزی تولید نیز از الگوریتم رقابت استعمارگران برای تصمیم‌گیری در خصوص برونسپاری یا تولید محصولات متفاوت بمنظور افزایش بهره وری استفاده شده است [۷]. در این تحقیق عملگر یادگیری برای استعمارگران[2] به الگوریتم رقابت استعماری اضافه شده تا آنها نیز تلاشی از سمت مقابل برای رسیدن به جواب‌های بهینه حتی برای کلونی‌های سایر استعمارگران داشته باشد. در دنیای واقعی این اتفاق در قالب جاسوسی اطلاعات کسب می‌شود و شاید به همین علت است که اکثر استعمارگران دارای سیستم‌های جاسوسی و اطلاعاتی قوی می‌باشند.

جهت بررسی صحت عملکرد این عملگر ابتدا عملگر پیشنهادی را برروی مثالهای متعدد از مجموعهٔ TSPLIB [۸] آزمایش نموده و نتایج آن مورد آنالیز و بررسی قرار خواهند گرفت تا مشخص شود که الگوریتم بهبود یافته قادر به حل مسائل با کیفیت بیشتر در زمان کمتر می‌باشد. مجموعهٔ TSPLIB مجموعه‌ای است از مسایل حل شده TSP که به صورت استاندارد در اختیار محققین جهت بررسی روش‌های ابتکاری و فرابتکاری قرار داده شده است.

ادامه مقاله بصورت زیر سازماندهی شده است: در بخش ۲ اصول الگوریتم رقابت استعماری استاندارد بیان و در بخش ۳ عملگر یادگیری استعمارگران تشریح می‌شود در بخش ۴ گزارشی از نتایج استفاده از این عملگر ارائه شده است، در بخش ۱.۴ برحسب تعداد باززایی برابر و در بخش ۲.۴ بر اساس زمان اجرای برابر. در نهایت در بخش ۵ نتیجه‌گیری از نتایج و تحقیقات آتی مورد بحث واقع می‌گردد.

۲- اصول الگوریتم رقابت استعماری

همانند دیگر الگوریتم‌های تکاملی، این الگوریتم، نیز با تعدادی جمعیت اولیه تصادفی که هر کدام از آنها یک "کشور" نامیده می شوند؛ شروع می‌شود[۳]. تعدادی از بهترین عناصر جمعیت (معادل نخبه‌ها در الگوریتم ژنتیک) به عنوان استعمارگر انتخاب می‌شوند. باقیمانده جمعیت نیز به عنوان مستعمره، در نظر گرفته می‌شوند.

استعمارگران بسته به قدرتشان، این مستعمرات۳ را با یک روند خاص که در ادامه می آید؛ به سمت خود می‌کشند. قدرت کل هر امپراطوری، به هر دو بخش تشکیل دهنده آن یعنی کشور استعمارگر (به عنوان هسته مرکزی) و مستعمرات آن، بستگی دارد. در حالت ریاضی، این وابستگی با تعریف قدرت امپراطوری به صورت مجموع قدرت کشور امپریالیست، به اضافه درصدی از میانگین قدرت مستعمرات آن، مدل شده است.

با شکل‌گیری امپراطوری‌های اولیه، رقابت استعماری میان آنها شروع می‌شود. هر امپراطوری که در رقابت استعماری، نتواند موفق عمل کرده و بر قدرت خود بیفزاید (و یا حداقل از کاهش نفوذش جلوگیری کند)، از صحنه رقابت استعماری، حذف خواهد شد. در نتیجه، در جریان رقابت‌های استعماری، به تدریج بر قدرت امپراطوری‌های بزرگتر افزوده شده و امپراطوری‌های ضعیف تر، حذف خواهند شد. با گذشت زمان، مستعمرات، از لحاظ قدرت به امپراطوری‌ها نزدیک‌تر خواهند شد و شاهد یک نوع همگرایی خواهیم بود. حد نهایی رقابت استعماری، زمانی است که یک امپراطوری واحد در دنیا داشته باشیم، با مستعمراتی که از لحاظ موقعیت، به خود کشور استعمارگر، خیلی نزدیک هستند.

۲-۱- *انتخاب استعمارگران*

در بهینه سازی، هدف یافتن یک جواب بهینه بر حسب متغیرهای مسئله، است. با ایجاد یک آرایه از متغیرها، مسئله جهت بهینه سازی، ایجاد می شود. در الگوریتم ژنتیک این آرایه، کروموزوم ولی در اینجا کشور نامیده می‌شوند. می‌توان جواب‌ها را که در اینجا کشورها هستند بصورت تصادفی ایجاد نمود. جهت پیاده سازی هزینه هر پاسخ را محاسبه و آنها را بر اساس هزینه پاسخ اشان از کوچک به بزرگ مرتب نموده و سپس به تعداد مورد نظر از استعمارگران از بالای لیست که دارای هزینه‌های کمتری هستند انتخاب و مابقی را به عنوان مستعمره می‌بایست بین این استعمارگران پخش شوند.

۲-۲- *تعیین تعداد مستعمره‌های هر امپراطوری*

جهت تعیین تعداد مستعمره‌های هر امپراطوری می‌بایست با توجه به هزینه هر استعمارگر قدرتی را برای هر امپراطوری تعیین نمود و سپس متناسب با قدرت هر امپراطوری تعدادی مستعمره را به آن اختصاص داد و ما آن را بصورت زیر انجام می‌دهیم.

قدم اول: محاسبه هزینه نرمالیزه شده استعمارگر n ام:

1. Constrained Optimization
2 .Imperialists
3.Colonies

h_j : هزینه استعمارگر jام

H_j : هزینه نرمالیزه شده استعمارگر jام

$$H_j = \max_i\{h_i\} - h_j \quad (۱)$$

همانگونه که در فرمول فوق مشاهده می نمایید[۳] بدترین هزینه از هزینه هر استعمارگر کسر شده تا میزان انحراف از بدترین استعمارگر یافته شود.

قدم دوم : محاسبه قدرت نسبی نرمالیزه‌ی استعمارگر nام [3]

n : تعداد استعمارگران

E_n : قدرت نسبی نرمالیزه‌ی استعمارگر nام

$$E_j = \left|\frac{H_j}{\sum_{i=1}^{n} H_i}\right| \quad (۲)$$

قدم سوم: محاسبه تعداد مستعمره‌های استعمارگر nام [۳]

$N.I.C_j$: تعداد مستعمره‌های استعمارگر j ام

N_{Colony} : تعداد کل مستعمره‌ها

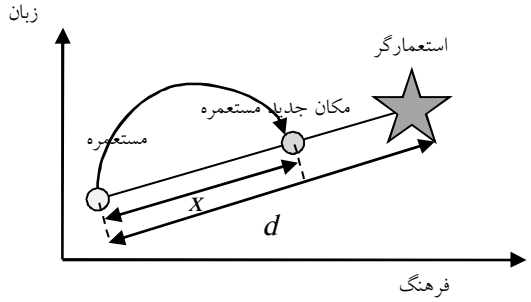
$$N.I.C_j = \text{round}\{E_j \times N_{Colony}\} \quad (۳)$$

حال با توجه به تعداد مورد نظر که از فرمول ۳ بدست آمد، می‌بایست بصورت تصادفی مستعمره‌ها را انتخاب و به هر استعمارگر، اختصاص داد.

۲-۳- حرکت مستعمره‌ها به سمت امپراطوری

سیاست همگون سازی (جذب) با هدف تحلیل فرهنگ و ساختار اجتماعی مستعمرات در فرهنگ حکومت مرکزی انجام می‌گرفت[۳].

این بخش از فرایند استعمار در الگوریتم بهینه سازی، به صورت حرکت مستعمرات به سمت کشور استعمارگر، مدل شده است. شکل (۱) شمای کلی این حرکت را نشان می دهد.



شکل (۱): سیاست جذب امپراطوری‌ها بر مستعمره‌هایشان

مطابق این شکل کشور امپریالیست کشور مستعمره را در راستای محورهای فرهنگ و زبان به سمت خود جذب می کند. همانگونه که در این شکل نشان داده شده است، کشور مستعمره، به اندازهٔ x واحد در جهت خط واصل مستعمره به استعمارگر، حرکت کرده و به موقعیت جدید، کشانده می‌شود .در این شکل، فاصله میان استعمارگر و مستعمره با d نشان داده شده است.x نیز عددی تصادفی با توزیع یکنواخت می‌باشد یعنی برای x داریم:

$$x \sim U(0, \beta \times d) \quad (۴)$$

که در آن β عددی بزرگتر از ۱ و کمتر از ۲ است و بزرگتر از ۱ بودن آن نیز به علت نزدیک شدن مستعمره از جهات مختلف به سمت استعمارگر می‌باشد. با بررسی تاریخی پدیده همگون‌سازی، یک حقیقت آشکار در این زمینه این است که علی رغم اینکه کشورهای استعمارگر بطور جدی پیگیر سیاست جذب بودند، اما وقایع به طور کامل مطابق سیاست اعمال شده آنها پیش نمی‌رفت و انحرافاتی در نتیجه کار وجود داشت.

در الگوریتم ICA، این انحراف احتمالی با افزودن یک زاویه تصادفی به مسیر جذب مستعمرات، انجام می‌گیرد .بدین منظور، در حرکت مستعمرات به سمت استعمارگر، کمی زاویه تصادفی نیز به جهت حرکت مستعمره، اضافه می‌کنیم که البته در این بخش بعلت پیچیدگی در بیش از دو بعد، بهبودهایی در آن صورت پذیرفته است [۱۰].

۲-۴- جابجایی بین استعمارگران و مستعمره‌ها

عین نابودی ساختارهای اجتماعی سیاسی کشور مستعمره در بعضی موارد نتایج مثبتی را نیز برای آنها در پی‌داشت. بعضی از کشورها در نتیجه اعمال این سیاست به نوعی از خودباوری عمومی دست یافتند و پس از مدتی همان نخبگان (به عبارت دیگر جذب شدگان فرهنگ استعماری) بودند که به رهبری ملت خود برای رهایی از چنگال استعمار پرداختند. در مدلسازی این واقعه تاریخی ICA به اینصورت عمل شده است که در حین حرکت مستعمرات به سمت کشور استعمارگر، ممکن بعضی از این مستعمرات به موقعیتی بهتر از استعمارگر برسند (به نقاطی در تابع هزینه برسند که هزینه کمتری را نسبت به مقدار تابع هزینه در موقعیت امپریالیست، تولید می‌کنند) در این حالت، کشور استعمارگر و کشور مستعمره، جای خود را با همدیگر عوض کرده و الگوریتم با کشور استعمارگر در موقعیت جدید ادامه یافته و این بار این کشور استعمارگر جدید است که شروع به اعمال سیاست همگون‌سازی بر مستعمرات خود می‌کند.

۲-۵- قدرت امپراطوری

قدرت یک امپراطوری برابر است با قدرت کشور استعمارگر، به اضافه درصدی از قدرت کل مستعمرات آن. بدین ترتیب هزینه کل یک امپراطوری از عبارت زیر محاسبه می‌شود:

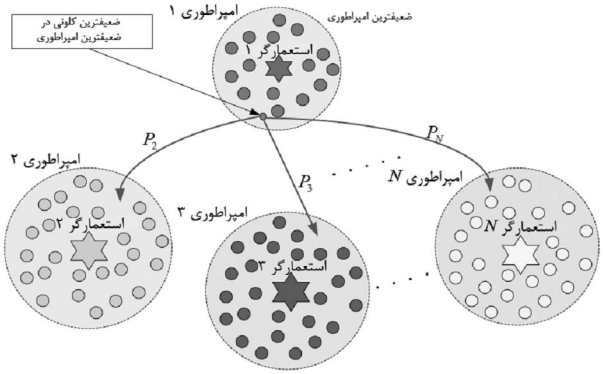
(۵)

$$T.C_n = \text{هزینه}\left(\text{استعمارگر}_n\right) + \xi \times \text{میانگین}\left\{\text{هزینه}\left(\text{مستعمرات یک امپراطوری}_n\right)\right\}$$

که در آن $T.C_n$ هزینه کل امپراطوری n ام و ξ عددی مثبت بین صفر و یک است و کوچک قرار دادن این پارامتر معادل اهمیت بیشتر استعمارگر مرکزی می‌باشد و بالعکس.

۲-۶- رقابت استعمارگران

همانگونه که قبلاً نیز بیان شد، هر امپراطوری ای که نتواند بر قدرت خود بیافزاید و قدرت رقابت خود را از دست بدهد، در جریان رقابت‌های امپریالیستی، حذف خواهد شد. برای مدل کردن این واقعیت، فرض می‌کنیم که امپراطوری در حال حذف، ضعیف ترین امپراطوری موجود است. بدین ترتیب، در تکرار الگوریتم، یکی یا چند تا از ضعیف ترین مستعمرات ضعیف ترین امپراطوری را برداشته و برای تصاحب این مستعمرات، رقابتی را میان کلیه امپراطوری‌ها ایجاد می‌کنیم. مستعمرات مذکور، لزوماً توسط قویترین امپراطوری، تصاحب نخواهند شد، بلکه امپراطوری‌های قویتر، احتمال تصاحب بیشتری دارند. شکل (۲) رقابت استعمارگران را نشان می‌دهد.



شکل (۲): ضعیفترین امپراطوری، ضعیفترین عضو خود را در رقابت از دست می‌دهد

جهت تعیین امپراطوری غالب می‌بایست بصورت زیر اقدام نماییم:

قدم اول : محاسبه هزینه نرمالیزه شده استعمارگر n ام:

$N.T.H_n$: هزینه کل نرمالیزه شده امپراطوری n ام

$$N.T.H_j = \max_i\{T.H_i\} - T.H_j \qquad (۶)$$

قدم دوم : قدرت نسبی نرمالیزه‌ی امپراطوری n ام برای تصاحب مستعمره

n' : تعداد امپراطوری‌ها

E'_n : قدرت نسبی نرمالیزه‌ی امپراطوری n ام برای تصاحب مستعمره

$$\acute{E}_j = \left| \frac{N.T.H_j}{\sum_{i=1}^{\acute{n}} N.T.H_i} \right| \qquad (۷)$$

بردار E' دارای اندازه $n' \times 1$می‌باشد و از مقادیر احتمال تصاحب امپراطوری‌ها تشکیل شده است. بردار تصادفیR هم اندازه با بردار E' می‌باشد که آرایه‌های این بردار اعدادی، تصادفی با توزیع یکنواخت در بازه بسته صفر تا یک می‌باشند. بردار D نیز از تفریق درایه‌های این دو بردار به صورت زیر ایجاد می‌شود.

$$D = \acute{E} - R = [D_1, D_2, \dots, D_{\acute{n}}] = [\acute{E}_1 - R_1, \acute{E}_2 - R_2, \dots, \acute{E}_{\acute{n}} - R_{\acute{n}}] \qquad (۸)$$

با داشتن بردار D مستعمرات مذکور به امپراطوری اختصاص می‌یابد که اندیس مربوط به آن در بردار D بزرگتر از بقیه باشد.

۲-۷- نابودی امپراطوری‌ها

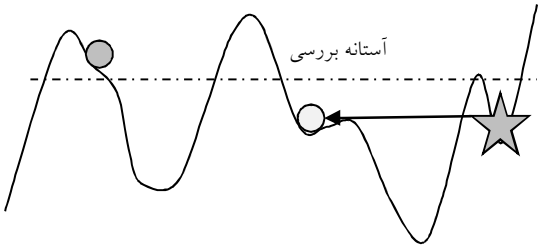
همانگونه که بیان شد، در جریان رقابت‌های امپریالیستی، خواه ناخواه، امپراطوری‌های ضعیف به تدریج سقوط کرده و مستعمراتشان به دست امپراطوری‌های قوی‌تر می‌افتد. یک امپراطوری زمانی حذف شده تلقی می‌شود که مستعمرات خود را از دست داده باشد.

۳- عملگر یادگیری استعمارگران

در این مقاله کوشیده شده تا عملگر یادگیری استعمارگران تعریف، تشریح، ایجاد و به عملگرهای قبلی افزوده شود. در نهایت نتایج این عملگر در کیفیت پاسخ و زمان حل مورد بررسی واقع شود. عملگر یادگیری استعمارگران در واقع انتقال اطلاعات مفید بصورت کنترل شده از سمت مستعمره‌ها به سمت استعمارگران است بنحوی که باعث پیشرفت استعمارگران و در واقع دستیابی به جواب بهینه جدید می‌باشد. بطور مثال می‌توان به انتقال تکنولوژی عطر از ایران به انگلیس یا انتقال تکنولوژی کنکورد از فرانسه به روسیه را نام برد که منجر به توسعه استعمارگران شد.

البته بعضی از این اطلاعات باعث بدتر شدن جواب می‌شود ولی چون اطلاعات بصورت کنترل شده از مستعمرات به امپراطوری‌ها انتقال می‌یابد قابل چشم پوشی می‌باشد. از طرفی اگر از همه کشورها اطلاعات دریافت شود شاید فقط هزینه انتقال اطلاعات (جاسوسی) را افزایش دهد و کمکی به بهبود ننماید پس نیاز به یک پالایش اولیه می‌باشد تا فقط کشورهایی که دارای وضعیت نزدیک به بهینه هستند مورد ارزیابی قرار گیرند. در عملگر یادگیری افزایش بازه جستجو باعث افزایش زمان جستجوی غیر مفید خواهد بود که با صاف نمودن می‌توان آن را محدود نمود.

طبق این عملگر جدید استعمارگران بدون توجه به اینکه کدام کلونی مستعمره کیست، شروع به یادگیری از کلونی‌های مختلف می‌نمایند و با یک حد آستانه‌ای مشخص شده، تعیین می‌نمایند که کدام مستعمره دارای پیشرفت قابل قبولی بوده است و سعی می‌نمایند به سمت آن کلونی‌ها با یک نرخ خاص حرکت نمایند و اگر بهبودی حاصل شد حالت استعمارگر بروز شود. این حد قابل قبول درصدی از بهترین جواب از بهترین استعمارگر می‌باشد. همانگونه که در شکل (۳) مشاهده می‌نمایید با حرکت استعمارگر (ستاره) به سمت مستعمره سفید از استعمارگر دیگر می‌تواند وضعیت استعمارگر را بهبود دهد.



شکل (۳) :یادگیری استعمارگر از مستعمره سفید

در تصویر (۳) حد آستانه با خط چین نمایش داده شده است و بر طبق این آستانه بررسی، مستعمره تیره مورد بررسی قرار نمی‌گیرد و مستعمره سفید، مورد بررسی قرار می‌گیرد. البته با استفاده از این عملگر سرعت همگرا شدن استعمارگران بیشتر می‌شود و درنتیجه می‌توان امپراطوری‌های مشابه را ادغام نمود که این خود باعث بالا رفتن سرعت و کمتر شدن زمان اجرا خواهد شد.

محل قرارگیری این عملگر در الگوریتم ICA نیز از اهمیت برخوردار است و می‌بایست قبل از عملگرهایی باشد که بر طبق هزینه استعمارگر فعالیت خاصی را به انجام می رسانند، باشد. مکان پیشنهادی بعد از عملگر «حرکت مستعمره‌ها به سمت استعمارگران» و قبل از عملگر «جابجایی بین استعمارگران و مستعمره‌ها» می‌باشد.

۴- مطالعه قیاسی

در این بخش نتایج الگوریتم ICA بهبود داده شده با عملگر یادگیری، با حالتی که بهبود نیافته (عملگر یادگیری استفاده نشده) مورد مقایسه و بررسی قرار می‌گیرد تا مشخص شود آیا تغییرات معنی‌دار است یا خیر. همچنین میزان بهبود در زمان‌های مساوی و تعداد اجراهای مساوی در مسائل TSPLIB مورد مقایسه قرار می‌گیرند. جهت چنین کاری می‌بایست از تعدادی مسایل TSP که شناخته شده می‌باشند استفاده نماییم که در جدول (۱) لیست آنها خواهد آمد.

جدول (۱): مسائل مورد بررسی TSPLIB

ردیف	نام مساله	اندازه مساله	مقدار بهینه
۱	Bays29	۲۹ شهر	۲۰۲۰
۲	Eil51	۵۱ شهر	۴۲۶
۳	Eil76	۷۶ شهر	۵۳۸
۴	rd100	۱۰۰ شهر	۷۹۱۰
۵	ch130	۱۳۰ شهر	۶۱۱۰

۴-۱- مطالعه در تعداد اجرای مساوی

در این بخش تمرکز مطالعات بر کیفیت پاسخ بعد از تعداد باززایی‌های برابر خواهد بود و نتایج الگوریتم بهبود یافته را با الگوریتم ICA بدون عملگر یادگیری مقایسه خواهد شد. همچنین میزان معنی دار بودن این بهبود مورد بررسی واقع می‌شود.

در جدول زیر تعداد باززایی برابر هر دو الگوریتم و زمان‌های خاتمه آنها آمده است. در اجراهای زیر تعداد جمعیت کلونی‌ها ۶۰۰ و تعداد استعمارگران ۶۰ فرض شده همچنین عدد *Gamma*، ۱.۵ برابر بهترین هزینه (هزینه بهترین استعمارگر) و عدد جهش ۱ درصد فرض شده است.

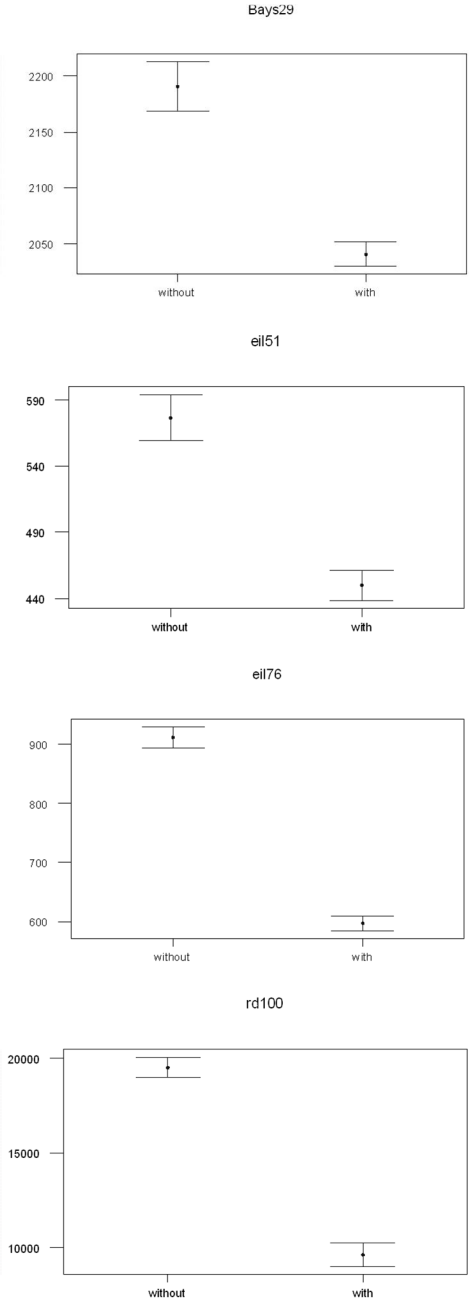
این مطالعات بر روی ۵ مساله از TSPLIB انجام پذیرفته و تعداد شهرها (گره‌ها) نیز در این مثال‌ها از ۲۹ تا ۱۳۰ می‌باشند. البته می‌توان مثالهای بیشتری را نیز از این دست آورد که مطالعات در مورد آنها نیز صدق می‌نماید همانند berlin52 و شهرهایی با بیش از ۱۳۰ شهر. کارایی الگوریتم ICA نسبت عکس با اندازه مساله دارد.

جدول (۲): نتایج مقایسه در ۸ اجرا و ۶۰۰ باززایی

	الگوریتم رقابت استعمارگران بدون یادگیری		الگوریتم رقابت استعمارگران با یادگیری	
نام مساله	میانگین هزینه	میانگین زمان اجرا	میانگین هزینه	میانگین زمان اجرا
bays29	۲۱۹۰.۶۲	۶۹.۱۲۸	۲۰۴۰.۳۷	۵۹.۵۰۹
eil51	۵۷۶.۹۶۱	۱۵۳.۱۱۰	۴۴۹.۶۸۶	۱۶۱.۵۷۶
eil76	۹۱۱.۸۹۱	۳۵۹.۲۹۲	۵۹۵.۸۲۷	۵۳۷.۷۰۳
rd100	۱۹۵۲۳.۴	۴۴۸.۳۹۳	۹۶۱۳.۵۹	۱۲۶۰.۳۷۸
ch130	۱۹۰۵۸.۵	۷۶۲.۰۲۲	۸۷۳۴.۸۵۵	۴۵۹۲.۲۱۳

در شکل (۴) بازه‌های تغییرات را برای هر مساله در ۶۰۰ فرایند باززایی و ۸ اجرا، نشان می‌دهد و همانطور که مشاهده می‌شود در بازه ۹۵ درصد تمام گرافها با عملگر یادگیری وضعیت بهتری از لحاظ هزینه نسبت به آنهایی که از عملگر یادگیری بهره‌مند نشده‌اند، نشان می‌دهد و موثر بودن فرایند یادگیری را اثبات می‌نمایند.

بازه تغییرات در سمت چپ میزان برازش را بدون عملگر یادگیری و در بخش راست تصویر میزان برازش با استفاده از عملگر یادگیری را در الگوریتم ICA نشان می‌دهد. همانگونه که مشاهده می‌شود حداکثر بازه تغییرات با عملگر یادگیری کمتر از حداقل بازه بدون استفاده از عملگر یادگیری می‌باشد که نشان از معنی‌دار بودن این تغییرات در سطح ۹۵٪ دارد.



شکل (۴): نمودار بازه‌ای از عملکرد عملگر یادگیری در موارد مختلف در تعداد اجرای برابر

حال این سوال مطرح می‌شود که آیا الگوریتم ICA استاندارد بدون افزودن عملگر جدید نیز در فرصت مشابه همین هزینه را حاصل می‌نمود یا خیر. پس می‌بایست مطالعات در بخش ۴-۲ بر روی زمان عملیات متمرکز شود و محدودیت زمان اجرا در الگوریتم ICA استاندارد در نظر گرفته شود.

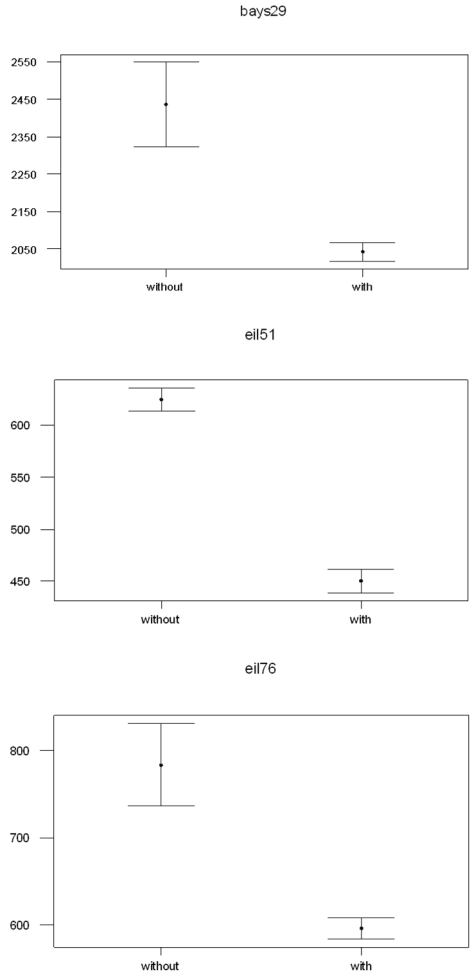
۴-۲- مطالعه در زمان اجرای مساوی

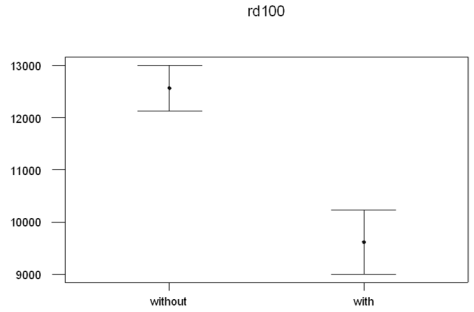
در این بخش مقاله، تمرکز مطالعات بر کیفیت پاسخ با توجه به زمان اجرای برابر خواهد بود و نتایج الگوریتم بهبود یافته با الگوریتم ICA بدون افزودن عملگر یادگیری استعمارگران مقایسه می‌شود و در نهایت میزان معنی‌دار بودن تغییرات بررسی خواهد شد.

در جدول (۳) میانگین ۸ اجرای هر دو الگوریتم را بر روی مسایل جدول (۱)، در زمان‌های اجرای برابر درج شده است. همانگونه که از جدول (۳) استنباط می شود دو ستون میانگین زمان اجرا در هر دوحالت (با عملگر یادگیری و بدون آن) به علت افزودن محدودیت زمان اجرا و حذف محدودیت تعداد باززایی برابر شده‌اند.

جدول (۳): نتایج مقایسه در زمان اجرای برابر

نام مساله	الگوریتم رقابت استعمارگران بدون یادگیری		الگوریتم رقابت استعمارگران با یادگیری	
	میانگین هزینه	میانگین زمان اجرا	میانگین هزینه	میانگین زمان اجرا
bays29	۲۴۳۶.۱۲	۵۹.۵۰۹	۲۰۴۰.۳۷	۵۹.۵۰۹
eil51	۶۲۴.۶۸۲	۱۶۱.۵۷۶	۴۴۹.۶۸۶	۱۶۱.۵۷۶
eil76	۷۸۳.۴۰۵	۵۳۷.۷۰۳	۵۹۵.۸۲۷	۵۳۷.۷۰۳
rd100	۱۲۵۶۳.۲	۱۲۶۰.۳۷۸	۹۶۱۳.۵۹	۱۲۶۰.۳۷۸
ch130	۹۷۵۷.۶۸	۴۵۹۲.۲۱۳	۸۷۳۴.۸۵۵	۴۵۹۲.۲۱۳





شکل (۵): نمودار بازه‌ای از عملکرد عملگر یادگیری در موارد مختلف در زمان اجرای برابر

همانگونه که در جدول (۳) و شکل (۵) (سطح اطمینان ۹۵ درصد) می‌توان مشاهده نمود الگوریتم رقابت استعمارگران در شرایط کاملاً مساوی تاثیر مثبت خود را نشان داده و این میزان بهبود بسته به نوع، اندازه و فضای پاسخ مساله می‌تواند متفاوت باشد.

در جدول (۴) میزان بهبود میانگین و میزان بهبود بهترین پاسخ آورده شده است. معادلات ۹ و ۱۰، *%PDM* و *%PDB* را بدین‌گونه تعریف می‌نمایند [۵].

$$\%PDM = \frac{\left(\text{پاسخ بهینه} - \text{میانگین پاسخها}\right) \times 100}{\text{پاسخ بهینه}} \qquad (۹)$$

$$\%PDM = \frac{\left(\text{پاسخ بهینه} - \text{بهترین پاسخ}\right) \times 100}{\text{پاسخ بهینه}} \qquad (۱۰)$$

جدول (۴): نتایج میزان بهبود در زمان اجرای برابر

نام مساله	الگوریتم رقابت استعمارگران با یادگیری	
	%PDM	*%PDB*
bays29	۰.۹۹	۰.۰۰
eil51	۵.۵۶	۲.۸۸
eil76	۱۰.۷	۷.۲۷
rd100	۲۱.۵۳	۱۱.۶
ch130	۴۲.۹۵	۲۶.۹۶
میانگین	۱۶.۳۴۶	۹.۷۴۲

۵- نتیجه‌گیری

با توجه به نتایج بدست آمده در بخش ۴ می‌توان نتیجه‌گیری نمود که یادگیری استعمارگران از کلونی‌های منتخب می‌تواند باعث بهبود پاسخ در زمان مشابه شود و این میزان بهبود بسته به نوع و اندازه مساله و یا فضای پاسخ می‌تواند متفاوت باشد. پس با افزودن این عملگر به الگوریتم ICA باعث افزایش سرعت همگرایی و حذف سریعتر استعمارگران در عملگر اتحاد را باعث می‌شود. در تحقیقات آتی سعی خواهد شد تا انتخاب استعمارگران توسط الگوریتم‌های هوشمندانه‌تری انجام گیرد تا سرعت همگرایی اولیه را افزایش دهد و مدت زمان اجرای الگوریتم رقابت کلونی استعمارگران بهبود یابد. جهت نایل شدن بدین هدف می‌توان از الگوریتم‌هایی بر مبنای شبکه عصبی (نقشه خودسازمانده) استفاده نمود و با تغییر اندکی در عملگر یادگیری استعمارگران و حرکت کلونی‌ها، آن را بهبود داد. در این صورت الگوریتم ارائه شده برای مسائل فروشنده دوره‌گرد با فواصل اقلیدسی قابل اجرا خواهد بود.

۶- منابع و مأخذ

[1] Atashpaz,E.G. , Hashemzadeh, F., Rajabioun, R., Lucas, C., (2008), **Colonial Competitive Algorithm: A Novel Approach for PID Controller Design in MIMO Distillation Column Process**, International Journal of Intelligent Computing and Cybernetics, 1(3), 337-355.

[2] Atashpaz, E.G., Hashemzadeh, F., Lucas, C., (2008), **Designing MIMO PID Controller using Colonial Competitive Algorithm: Applied to Distillation Column Process**, 2008 IEEE Congress on Evolutionary Computation, 1929 – 1934.

[3] Atashpaz, E.G., Lucas, C., (2007), **Imperialist Competitive Algorithm: An Algorithm for Optimization Inspired by Imperialistic Competition**, 2007 IEEE Congress on Evolutionary Computation, 4425083, 4661 – 4667.

[4] Haibin, D., Chunfang, X., Senqi, L., Shan, S., (2009), **Template matching using chaotic imperialist competitive algorithm**, Pattern Recognition Letters, Article in Press.

[5] Jean-CharlesCre´put Abderrafia˜a, K., (2009), **A memetic neural network for the Euclidean traveling salesman problem** ,Neurocomputing 72, 1250– 1264.

[6] Lucas, C., Nasiri-Gheidari, Z., Tootoonchian, F., (2010), **Application of an imperialist competitive algorithm to the design of a linear induction motor**, Energy Conversion and Management, 51(7), 1407-1411.

[7] Nazari-Shirkouhi, S., Eivazy, H., Ghodsi, R., Rezaie, K., Atashpaz-Gargari, E., (2010), **Solving the integrated product mix-outsourcing problem using the Imperialist Competitive Algorithm** , Expert Systems with Applications 37, 7615–7626.

[8] Reinelt, G., (1991),**A traveling salesman problem library**, ORSAJ. Comput, 3, 376–384.

[9] Roshanaei, M., Atashpaz-Gargari, E., Caro, L., (2008), **Adaptive Beam forming Using Colonial Competitive Algorithm**, 2nd International Joint Conference on Computational Engineering,