

# طراحی الگوریتم های بهینه سازی مبتنی بر CLA-EC برای محیط های پویا

مژده خاکسار منشاد<sup>۱</sup>، محمد رضا میدی<sup>۲</sup>

## چکیده

بسیاری از تحقیقات در محاسبات تکاملی روی بهینه سازی مسائل ایستا بحث می کند. در حالی که بسیاری از مسائل بهینه سازی دنیای واقعی پویا هستند و روش های بهینه سازی نیاز است، که قادر باشد به طور پیوسته راه حل را با محیط در حال تغییر وفق دهد. محاسبات تکاملی بر پایه آتاماتای یادگیر سلولی (CLA-EC) یک مدل محاسبات تکاملی است که از ترکیب آتاماتای یادگیر سلولی (CLA) و مدل محاسبات تکاملی (EC) به وجود آمده است. در این مقاله چهار ورژن از مدل CLA-EC برای یک نوع از مسائل بهینه سازی پویا که مسئله قله های متغیر ک نامیده می شود، بکار گرفته شده است. سپس دو رویکرد جدید معرفی شده است، رویکرد اول بر پایه حافظه دار کردن مدل CLA-EC است و در دیگری که CLA-EC جستجو/حافظه نامیده می شود، از ساختار چند جمعیتی استفاده شده است. برای نشان دادن تاثیر CLA-EC های معرفی شده، آزمایشات متعددی طراحی شد. این آزمایشات نشان می دهد CLA-EC معرفی شده، نسبت به رویکردهایی که تا کنون برای محیط های پویا طراحی شده است دارای دقت بالاتری است.

## کلمات کلیدی

محاسبات تکاملی، محیط های پویا، آتاماتای یادگیر سلولی، CLA-EC، مسئله قله های متغیر، جستجو/حافظه.

## Designing Optimization Algorithms based on CLA-EC for Dynamic Environment

Mojdeh khaksar manshad; mohammadreza meybodi

### ABSTRACT

Most research in evolutionary computation focuses on optimization of static, non-changing problems. Many real-world optimization problems, however, are dynamic, and optimization methods are needed that are capable of continuously adapting the solution to a changing environment. Cellular Learning Automata based Evolutionary Computing (CLA-EC) is an evolutionary computing model obtained by combining cellular learning automata (CLA) model and evolutionary computing (EC) model. This paper uses four version of CLA-EC model for a kind of dynamic optimization problem that called moving peaks problem. Then we proposed two new approaches, the first approach is based on memorization the CLA-EC model, the other one that called memory-search CLA-EC is uses a novel multi-population structure. In order to show the effectiveness of the proposed CLA-EC extensive computer experimentation were conducted. It has been shown that the proposed CLA-EC outperforms existing approaches reported for dynamic environments in terms of accuracy.

### KEYWORDS

Evolutionary Computation, Dynamic Environments, Cellular Learning Automata, CLA-EC, Moving Peaks Problem, Memory-Search CLA-EC.

<sup>۱</sup> دانشجوی کارشناسی ارشد دانشگاه آزاد اسلامی واحد قزوین، دانشکده برق، ریانه و فن آوری اطلاعات، پست الکترونیک:

khaksar\_mojdeh@yahoo.com

<sup>۲</sup> عضو هیئت علمی دانشگاه امیرکبیر، دانشکده کامپیوتر، پست الکترونیک: mmeybodi@aut.ac.ir

## ۱. مقدمه

هر گاه یک تغییر در هدف بهینه سازی ، نمونه مسئله یا بعضی تغییرات محدودیت<sup>۱</sup> یک مسئله بهینه سازی رخ دهد ، ممکن است بهینه آن مسئله تغییر کند. اگر این حالت رخ دهد و فق دادن راه حل با راه حل قدیمی ضروری است. از آن جا که الگوریتم های تکاملی به طور رایج در حال تکامل هستند، و ذاتاً به طور پیوسته در حال وفق دادن هستند، یک کاندیدای مناسب به نظر می آیند. مشکل اصلی الگوریتم های تکاملی استاندارد، این هست که آن ها عاقبت به یک بهینه همگرا<sup>۲</sup> می شوند و بنابراین تنو<sup>۳</sup> لازم را برای اکتشاف<sup>۴</sup> به صورت کارا از دست می دهند. بنابراین هنگامی که جمعیت الگوریتم تکاملی همگرا شد ، توانایی اش برای وفق پذیری با تغییرات محیط را از دست می دهد. روش هایی برای مناسب ساختن الگوریتم های تکاملی برای محیط های پویا وجود دارد. برخی از روش هایی که به صورت گسترده استفاده شده اند عبارتند از: روش های شروع دوباره [۱،۲،۳،۴،۵،۶،۷]، جهش و فرقی [۸،۹،۱۰،۱۱،۱۲،۱۳،۳۰،۳۱] و حافظه اضافی [۱۴،۱۵،۱۶،۱۷،۱۸،۱۹،۲۰]، انتخاب اصلاحی<sup>۵</sup> [۲۱،۲۲،۲۳،۲۴،۲۵،۲۶]، روش های چند جمعیتی<sup>۶</sup> [۲۷،۲۸،۲۹] . همه محیط های پویا یکسان نیستند و محیط های پویای متفاوت نیاز به روش های بهینه سازی متفاوت دارند. معیارهای لازم برای تشخیص این که یک مسئله محک مناسب است ، عبارت است از: باید این امکان را داشته باشد تا بسیاری از متغیرهای محیطی را تغییر دهد، نهایتاً باید محک هایی برای انکد باینری و مقادیر حقیقی وجود داشته باشد، آن ها باید برای پیاده سازی و توصیف ساده باشند، آن ها برای تحلیل باید ساده باشند، باید از لحاظ محاسباتی کارا باشند. جفت کردن پویای بیت<sup>۷</sup>، سهمی متحرک<sup>۸</sup>، مسئله کوله پشتی با زمان متغیر<sup>۹</sup>، تابع قله های متحرک<sup>۱۰</sup>، مسائل برنامه ریزی<sup>۱۱</sup>، قله های در حال نوسان<sup>۱۲</sup>، مسائلی هستند که به عنوان مسائل محک مناسب شناخته شده اند.

تاكنون CLA-EC برای مسائل بهینه سازی استاتیک به کار گرفته شده است. بعضی از اشکالات CLA-EC استاندارد سرعت همگرایی پایین و دقت کم برای بعضی از مسائل بهینه سازی است. و همین طور CLA-EC توانایی دنبال کردن تغییرات محیط های پویا را ندارد. بنابراین نیاز است تا تغییراتی در CLA-EC ایجاد شود تا توانایی دنبال کردن تغییرات را داشته باشد. یکی از دلایل این عدم توانایی از دست رفتن تنو<sup>۱۳</sup> برای محیط های پویا جمعیت بعد از گذشت چند نسل است. در این مقاله هدف اصلی طراحی الگوریتم های بهینه سازی مبتنی بر CLA-EC است. الگوریتم های طراحی شده در این مقاله با الگوریتم های جهش و فرقی و الگوریتم های تکاملی مبتنی بر حافظه مورد مقایسه قرار خواهد گرفت. در همین راستا دو معیار بهینگی یعنی «کارایی برون خطی» و «خطای برون خطی» مورد توجه قرار گرفته است. مسائل محکی که در این مقاله مورد ارزیابی قرار می گیرد مسئله قله های متحرک خواهد بود. تمام پیاده سازی ها در محیط MATLAB انجام خواهد شد. مسئله محک قله های متحرک شامل  $m$  قله در یک فضای پارامتر مقدار حقیقی  $n$  بعدی است که چشم انداز شایستگی به عنوان ماکریزم روی همه تابع های قله ای تعریف می شود و به صورت زیر فرموله می شود.

$$f(\vec{x}, t) = \max_{i=1 \dots m} (B(\vec{x}), \max P(\vec{x}, h_i(t), W_i(t), \vec{P}_i(t))) \quad (1)$$

که  $B(\vec{x})$  نا متغیر با زمان که پایه واساس چشم انداز است و  $p$  تابعی است که شکل قله را تعیین می کند، که هر یک از  $m$  قله پارامترهای متغیر با زمان شامل ارتفاع  $h$ ،عرض  $w$ ، و مکان  $p$  است. بعد از هر  $\Delta e$  ارزیابی ، ارتفاع، عرض و مکان هر قله تغییر می کند. ارتفاع و عرض هر قله به وسیله اضافه کردن متغیر گوسین تصادفی تغییر می کند. مکان هر قله به وسیله بردار  $\vec{s}$  با طول ثابت  $S$  حرکت خواهد کرد. بنابراین پارامتر  $s$  اجازه کنترل شدت یک تغییر را می دهد،  $\Delta e$  فرکانس تغییر را تعیین می کند. یک پارامتر جدید  $\lambda$ ، تعیین می کند که تغییر مکان قله چه مقدار بستگی به حرکت قبلی اش دارد. اگر  $\lambda = 0$  باشد هر حرکت کاملاً تصادفی است، و برای  $\lambda = 1$  قله همیشه در جهت یکسان حرکت خواهد کرد. فرمول هایی که باعث تغییر یک قله می شوند به صورت زیر هستند.

$$\begin{aligned} \sigma &\in N(0,1) \\ h_i(t) &= h_i(t-1) + height\_severity.\sigma \\ w_i(t) &= w_i(t-1) + width\_severity.\sigma \\ \vec{p}_i(t) &= p_i(t-1) + \vec{v}_i(t) \end{aligned} \quad (2)$$

## 2. الگوریتم تکاملی مبتنی بر آناتماتای یادگیر سلوالی

الگوریتم های تکاملی از جمله الگوریتم مطرح شده در این مقاله امکان جستجو را در هر فضای جستجوی گسسته متناهی دلخواه دارند، اما به منظور ساده کردن ارائه الگوریتم فرض می کنیم فضای جستجوی مورد نظر یک فضای متناهی دودویی باشد. بنابراین مساله بهینه سازی به فرم زیر قابل طرح است.

$$\max\{f(\underline{X}) \mid \underline{X} \in B^n\} \quad (3)$$

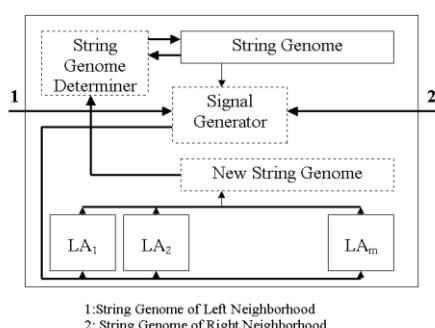
به طوریکه  $(.)f$  یک تابع حقیقی و  $B^n = \{0,1\}^n$  فضای جستجوی دودویی مورد نظر است. فرض کنید هر ژنوم مورد بحث در این الگوریتم دارای دو مولفه رشته ژنومی و مدل ژنومی باشد. رشته ژنومی همان راه حل های میانی مساله مورد نظر می باشند. مدل ژنومی مشکل از تعدادی اتوماتای یادگیر می باشد که بر اساس تجارت گذشته خود و ژنومهای دیگر آموزش می بینند. و بدین ترتیب فرایند تکامل به سمتی هدایت می شود تا ارزش رشته ژنومی بر اساس تابع ارزیابی بهبود یابد. نحوه انتساب اتوماتاهای یادگیر مدل ژنومی به بیتها رشته ژنومی به گونه های مختلفی قابل طرح است. برای مثال می توانیم رشته ژنومی را به دسته های دو بیتی تقسیم کرده و به ازای هر دسته یک اتوماتای یادگیر در نظر بگیریم. به این ترتیب با فرض دودویی بودن فضای جستجو و رشته ژنومی به طول  $n$  (با فرض زوج بودن)، تعداد  $\frac{n}{2}$  اتوماتای یادگیر، هر کدام با چهار عمل خواهیم داشت. برای مثال در این حالت اگر یک اتوماتای یادگیر عمل ۳ را انتخاب کند به معنای آن است که بیت ۱ و ۲ دسته متناظر آن مقادیر ۱ و ۱ را می پذیرند. در این پایان نامه با فرض مستقل بودن متغیرهای مساله ساده ترین شکل مدل ژنومی را در نظر می گیریم. به عبارت دیگر به ازای هر بیت در رشته ژنومی یک اتوماتای یادگیر در مدل ژنومی خواهیم داشت که مقدار آن بیت را تعیین می کند. تا به اینجا ساختار مورد نیاز در الگوریتم را شرح دادیم. همانطور که آشکار است این ساختار مانند ساختار سلول در اتوماتای یادگیر سلولی است.

حال برای ادامه بحث یک اتوماتای یادگیر سلولی،  $CLA(\underline{L}_1, \dots, \underline{L}_k)$ ، با  $k$  سلول که هر سلول مجهز به  $n$  اتوماتای یادگیر است در نظر بگیرید. در هر سلول یک رشته به طول  $n$  نشان دهنده حالت آن سلول می باشد که در الگوریتم مورد نظر ما این رشته همان رشته ژنومی می باشد که در قسمت قبل آن را شرح دادیم. واضح است که فضای حالت هر سلول  ${}^2$  می باشد. با فرض سنکرون بودن اتوماتای یادگیر سلولی، در زمان  $t$  هر سلول،  $t$ ، رشته های ژنومی خود و همسایگان خود را مورد بررسی قرار داده و بر اساس تابع ارزیابی از میان آنها دسته ای را به عنوان ژنومهای مناسب انتخاب می کند. این فرایند معادل انتخاب زوج در طبیعت می باشد. این شیوه انتخاب دو طرفه نمی باشد. به عبارت دیگر اگر یک ژنوم، ژنوم دیگری را به عنوان یکی از کاندیدهای خود انتخاب نماید، تضمینی برای انتخاب شدن از طرف ژنوم مقابل وجود نخواهد داشت. البته این روش معادل عینی در بین سیستمهای طبیعی ندارد. ژنوم بر اساس ژنومهایی که انتخاب نموده است، بردار سیگنال تقویتی را ساخته و به اتوماتاهای یادگیر خود می دهد. این مکانیسم در واقع معادل جفت گیری ژنوم مورد نظر می باشد. نکته ای که باید به آن توجه داشته باشیم این است که نحوه انتخاب ژنومها و تولید بردار تقویتی که قانون اتوماتای یادگیر سلولی را نشان می دهد، از پارامترهای بسیار مهم الگوریتم می باشد.

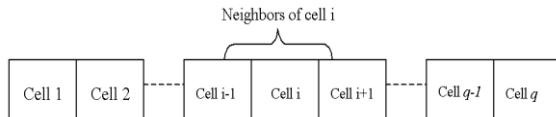
پس از آن برای تولید  $X_{t+1}^i$ ، هر کدام از اتوماتاهای یادگیر مقدار مورد نظر خود را برای بیت متناظر خود در ژنوم اعلام می کنند. در صورتی که ژنوم جدید،  $new_{t+1}^i$ ، ارزش بیشتری نسبت به ژنوم قبلی داشته باشد، جایگزین آن می شود، در غیر این صورت سلول رشته ژنومی قبلی را حفظ می کند.

$$X_{t+1}^i = \begin{cases} X_t^i & f(X_t^i) > f(new_{t+1}^i) \\ new_{t+1}^i & f(X_t^i) \leq f(new_{t+1}^i) \end{cases} \quad (4)$$

این بخش از الگوریتم معادل یادگیری از تجارت قبلی ژنوم و حفظ آنها در نسلهای بعدی می‌باشد. این مکانیزم را می‌توان به نوعی معادل تاثیر یادگیری بر تکامل از دیدگاه ثئوری بالدوین دانست. در شکل ۱(نمایی از یک سلول الگوریتم در یک شبکه تک بعدی (شکل ۲) دیده می‌شود.



شکل (1) نمایی از یک سلول در الگوریتم تکاملی مبتنی بر آتماتاتیک پادگیر سلولی،



شکل (2) نمایی از یک شبکه سلولی استفاده شده الگوریتم تکاملی مبتنی بر آتاماتیک یادگیرسlovی

قانون تعريف شده برای اتوماتیک یادگیر سلولی مورد استفاده در الگوریتم پیشنهادی از دو بخش تشکیل شده است. بخش اول استراتژی انتخاب می باشد. منظور از استراتژی انتخاب سلول  $i$  ام، نحوه انتخاب تعداد مشخص  $Se \leq m$ .  $Se \leq m$  رشتہ ژنومی از میان  $m$  ژنوم همسایه،  $P^i_{Se}$  می باشد. در استراتژی مورد استفاده،  $Se$  ژنوم از بهترین ژنوهای  $P^i$  را انتخاب می کنیم و آن را  $P^i_{Se}$  می نامیم.

بخش دوم قانون استراتژی تولید سیگنال تقویتی با استفاده از مجموعه  $P^i_{Se}$  می باشد. بردار سیگنال تقویتی به روشهای گوناگون قابل محاسبه است. در این بخش به شرح روش مورد استفاده در این پایان نامه می پردازیم. فرض کنید  $\bar{\beta}_t^i = (\beta_t^{i,1}, \beta_t^{i,2}, \dots, \beta_t^{i,n})$  بردار مورد نظر باشد. در این صورت هرگاه  $\bar{\beta}_t^i \in \{0,1\}^n$  محیط از نوع  $P$  می باشد. در چنین محیطی مقدار  $1$  به عنوان جرمیه و  $0$  به عنوان پاداش در نظر گرفته می شود. در محیط  $Q$ ,  $\bar{\beta}_t^i$  می تواند به طور گستته یک بردار از بردارهای محدود در فاصله  $[0,1]^n$  و در محیط از نوع  $S$ , بردار تصادفی در فاصله  $[0,1]^n$  باشد. در ادامه کار فرض ما بر  $P$  بودن محیط است.

اگر  $(X_t^i = (X_t^{i,1}, \dots, X_t^{i,n})$  رشتہ ژنومی سلول  $i$  ام در زمان  $t$  باشد. برای هر متغیر  $j \leq n$ ,  $X_t^{i,j} = 1$  و به ازای مقادیر  $1, k = 0, 1, \dots, n$  را به صورت زیر محاسبه می کنیم.

$$N_{i,j}(k) = \sum_l \delta_l(X_t^{i,j} = k | P_{Se}^i) \quad (5)$$

به طوریکه اگر متغیر  $X_t^{i,j}$  در  $i$  امین سلول، مقدار  $k$  را داشته باشد  $\delta_l(X_t^{i,j} = k | P_{Se}^i) = 1$  و در غیر این صورت برابر صفر است. سپس با استفاده از  $N_{i,j}(k)$  مقدار  $\beta_t^{i,j}$  را بدست می آوریم.

$$\begin{aligned} \beta_t^{i,j} &= u(N_{i,j}(1) - N_{i,j}(0)) \quad \text{If } X_t^{i,j} = 0 \\ \beta_t^{i,j} &= u(N_{i,j}(0) - N_{i,j}(1)) \quad \text{If } X_t^{i,j} = 1 \end{aligned} \quad (6)$$

که  $u(\cdot)$  تابع پله است. قانونی که در بالا آن را شرح دادیم، به فرم قوانین کلی می باشد. در شکل (3) مجازی الگوریتم ارائه شده است.

Initialize.

While not done do

For each cell  $i$  in CLA do in parallel

Generate a new string genome

Evaluate the new string genome

If  $f(\text{new string genome}) > f(\text{old string genome})$  then

Accept the new string genome

End if

Select  $Se$  cells from neighbors of cell  $i$

Generate the reinforcement signal vector

Update LAs of cell  $i$

End parallel for

End while

شکل (3) الگوریتم تکاملی مبتنی بر اتوماتیک یادگیر سلولی

الگوریتم CLA-EC دارای مدل های متفاوتی است که هر یک از مدل ها در زیر توضیح داده شده است:

CLA-EC همکارانه [۳۳]: کارایی CLA-EC بستگی زیادی به نرخ های یادگیری آتاماتیک یادگیر در سلول های CLA-EC دارد. اگر نرخ یادگیری آتاماتیک یادگیر بالا انتخاب شود، معمولاً الگوریتم به سرعت با بهره برداری از جواب های موجود به سمت جواب حرکت می کند ولی به

دلیل این که فضای جستجو به طور کامل کاوش نمی شود ممکن است در نقاط بھینه محلی به دام بیفتد. در مقابل اگر نرخ یادگیری پایین انتخاب شود به دلیل این که کاوش در فضای جستجو به طور کامل صورت می پذیرد پاسخ به دست آمده توسط الگوریتم از دقت خوبی برخوردار است ولی در عوض سرعت همگرایی پایین است. یک روش برای حل مشکل فوق الذکر استفاده از چندین CLA-EC با نرخ های یادگیری متفاوت است. CLA-EC های همکار از طریق به اشتراک گذاشتن جواب های میانی هم از سرعت بالا و هم از دقت بالا بهره می برد. CLA-EC همکارانه شامل دو CLA-EC با مشخصات کاملاً یکسان ولی نرخ های یادگیری متفاوت برای آتماتای یادگیر مستقر در سیستم می باشد. هر CLA-EC همکار به طور دوره ای جواب میانی به دست آمده در هر سلول را با جواب میانی سلول متناظر در CLA-EC دیگر، تعویض می کند. به این ترتیب یکی از CLA-EC ها عمل کاوش و دیگری عمل بهره برداری را انجام می دهد. همکاری CLA-EC ها حرکت سریع و دقیق به سمت جواب بھینه را باعث می شود.

CLA-EC ناهنگام [۳۲]: در CLA-EC همگام تمام سلول ها به طور همزمان فعال می شوند و هر سلول با مقادیر سلول های همسایه در تکرار قبلی سروکار دارد و وضعیت نهایی سلول ها فقط به آتماتاهای یادگیر بستگی دارد. با استفاده از بروز رسانی ناهمگام وضعیت نهایی به ترتیب فعال سازی سلول ها نیز وابسته خواهد بود. همچنین در روش های به روز رسانی ناهمگام برخلاف روش به روز رسانی همگام، هر سلول با مقادیر سلول های همسایه در تکرار فعلی سروکار دارد. روش های فعال سازی به دو گروه روش های مبتنی بر زمان<sup>۱۴</sup> و روش های مبتنی بر گام<sup>۱۵</sup> تقسیم بندی می شوند. در روش های مبتنی بر زمان، زمان فعال سازی به طور صریح مشخص می شود. در روش های مبتنی بر گام، ترتیب فعال سازی بر اساس یک ترتیب ثابت و از پیش تعیین شده فعال می شوند.

ترتیب تصادفی: در این روش ترتیب فعال سازی به این صورت ساخته می شود که اولین سلول به طور تصادفی از میان  $n$  سلول انتخاب می شود. سپس دومین سلول به طور تصادفی از میان  $n-1$  سلول باقی مانده انتخاب شده و به همین ترتیب ادامه می یابد تا تمام سلول ها انتخاب شوند. نکته ای که باید به آن توجه کرد این است که هنگام انتخاب، تمام سلول های باقی مانده شناسی یکسانی برای انتخاب شدن دارند. به عبارت دیگر، انتخاب بر اساس توزیع یکنواخت بدون جایگذاری است.

ترتیب جدید تصادفی: در این روش برای هر تکرار از یک ترتیب فعال سازی تصادفی بر مبنای توزیع یکنواخت با جایگذاری استفاده می شود. در این روش برخلاف روش ترتیب تصادفی ترتیب فعال سازی در تمام تکرارها یکسان نمی باشد.

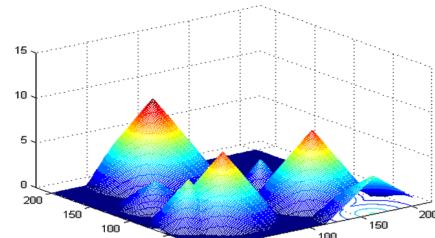
ترتیب یکنواخت: در این روش انتخاب سلول برای فعال سازی به طور تصادفی با توزیع یکنواخت انجام می گیرد. به عبارت دیگر از یک انتخاب تصادفی یکنواخت با جایگذاری استفاده می شود. در  $k$  گام ممکن است بعضی از سلول های بیش از یکبار فعال شوند و بعضی اصلاً فعال نشوند. در این روش برخلاف روش های قبلی، برای انتخاب سلول بعدی به حافظه اضافی نیاز نمی باشد.

CLA-EC بازترکیبی [۳۳]: در طول عملیات CLA-EC استاندارد، آتماتاهای یادگیر موجود در هر سلول، بردارهای احتمال خود را تغییر می دهند تا رشته های ژنومی محلی خوبی را برای نسل بعدی تولید کنند. فرض می شود که همه بیت های این رشته ژنومی محلی خوب بوده و برای یادگیری آتماتاهای یادگیر استفاده خواهد شد. در اینجا رشته هایی وجود دارند که در حالت کلی کیفیت بالایی ندارند ولی جواب های جزی خوبی را نگه می دارند. CLA-EC استاندارد مکانیزمی برای جایی رشته های جزی میان سلول هایی که حاوی این رشته ها هستند ندارد. برای حل این مشکل از بازترکیب میان چند رشته در سلول ها استفاده شده است. در ضمن این الگوریتم به جستجوی بهتر فضا کمک می کند.

# کنفرانس داده کاوی ایران

جدول (1) پارامترهای تابع قله های متحرک

Parameter	Value
number of peaks $m$	10
Frequency of change $f$	every 500 evaluations
height severity	7.0
width severity	1.0
peak shape	cone
shift length $s$	1.0
number of dimensions $D$	5
cone height range $H$	[30.0, 70.0]
cone width range $W$	[1, 12]
cone standard height $I$	50.0
Search space range $A$	[0, 100]



شکل (4) نمایی از قله های متحرک

در محیط های پویا برای فهمیدن این که کدام رویکرد بر رویکرد دیگر اولویت دارد ، تنها لازم است که همگرایی را نمایش دهیم(بهترین و متوسط شایستگی هر نسل) و آن ها را به صورت بصری مقایسه کنیم. بسته به مسئله مورد نظر یکی از اندازه گیری های شمارشی ممکن است انتخاب خوبی برای مقایسه باشد.

کارایی درون خطی  
به صورت میانگین همه ارزیابی ها بر روی کل اجرا محاسبه می شود.

$$x = \frac{1}{T} \sum_{t=1}^T e_t \quad (7)$$

کارایی برون خطی

\*  
 $x$  به عنوان بهترین میانگین بدست آمده در هر مرحله محاسبه می شود.

$$x^* = \frac{1}{T} \sum_{t=1}^T e_t^* \text{ with } e_t^* = \max\{e_1, e_2, \dots, e_t\} \quad (8)$$

عموماً فرض می شود که بهینه سازی در محیط شبیه سازی شده انجام می شود و تنها بهترین راه حل به دنیای واقعی وارد می شود. کارایی برون خطی باید تنها به افرادی رسیدگی کند که تا آخرین تغییر در محیط ارزیابی شده اند.

$$x^* = \frac{1}{T} \sum_{t=1}^T e_t^* \text{ with } e_t^* = \max\{e_\tau, e_{\tau+1}, \dots, e_t\} \quad (9)$$

$\tau$  being the last time step < t

که در آن یک تغییر در محیط روی داده است. البته این نیازمند این است که تغییرات محیطی برای مشاعده کننده شناخته شده باشد.

#### 4. آزمایشات انجام شده

یک استراتژی تکاملی کاملاً رایج تطبیق نرخ جهش بعد از تغییر محیط است. حرکت به این صورت است که نرخ جهش باید بعد از یک تغییر در محیط افزایش یابد تا به جمعیت همگرا شده کمک کند تا خارج گردد و جستجو از ابتدا آغاز شود در این مقاله در قسمت اول به حل مساله قله

های متحرک با استفاده از الگوریتم ژنتیک وفقی می پردازیم و سپس مساله قله های متحرک را با انواع مدل های CLA-EC حل کرده و به مقایسه جواب های بدست آمده می پردازیم. سپس برای بهبود عملکرد الگوریتم های CLA-EC به معرفی مدل حافظه/جستجو می پردازیم. برای آزمایشات گزارش شده در این مقاله الگوریتم های CLA-EC طراحی شده با الگوریتم های ژنتیک با جهش وفقی مورد مقایسه قرار می گیرند. در اینجا برای الگوریتم ژنتیک مورد استفاده از کدینگ حقیقی، جایگزینی تولیدی با نخبه گرایی ۱ و تقاطع دو نقطه ای با احتمال ۰.۰۵ و

جهش وفقی با احتمال  $\frac{1}{n}$  که  $n$  تعداد ابعاد است ( $n=5$ ) استفاده می کنیم. در الگوریتم CLA-EC از ۱۰۰ سلول استفاده می شود. هر یک از توابع دارای ۵ بعد هستند و برای انکد کردن هر بعد از ۲۷ بیت استفاده شده است، که این، به این معنی است که هر سلول دارای ۱۳۵ آتماتای یادگیر است. برای سهولت در ارائه از نماد CLA-EC(LRI(a,b),r,se,q) با سلول، شاعع همسایگی  $r$ ، تعداد سلول انتخابی  $se$ ، آتماتای یادگیر با پارامترهای پاداش  $a$  و حریمه  $b$  است. در آزمایشات انجام شده توسط CLA-EC، از یک اتوماتای یادگیر شاععی دارای ۱۰۰ سلول و همسایگی با شاعع یک دارای دور استفاده شده است. همچنین از اتوماتای یادگیر LRI با نرخ پاداش ۰.۰۱ (یعنی نرخ حریمه صفر است) استفاده شده است. تعداد گام در هر دو الگوریتم، ۵۰۰۰ تکرار است. پیاده سازی ها روی ۳۰ اجرا میانگین گرفته شده است. از آن جایی که برای توابع شایستگی پویا گزارش بهترین راه حل به دست آمده، مناسب نخواهد بود، در اینجا میانگین خطای برون خطی و کارایی برون خطی گزارش شده است. CLA-EC همکارانه شامل دو CLA-EC<sub>L</sub> خطی ساده و CLA-EC<sub>H</sub> هر کدام با شاعع باشد.

	SEA	CLA-EC	Cooperative CLA-EC	Recombinative CLA-EC	Uniform CLA-EC	CLA-EC <sub>H</sub>	همسایگی ۱ می باشد.
شامل آتماتاهای	۱۶.۷۳۵۸۲	۴۵.۱۵۶۵۷	۱۳.۴۱۴۷	۳۹.۹۷۷۳۸	۴۴.۹۷۶۹۸		
یادگیری بالا و							$L_{RI}$ با نرخ

CLA-EC<sub>L</sub> شامل آتماتاهای یادگیر با نرخ یادگیری پایین می باشد. طول هر دوره ۱۰ تکرار در نظر گرفته شده است. هر CLA-EC<sub>H</sub> همکارانه با  $q/2$  سلول، شامل دو CLA-EC ساده با ۲ سلول است و پس از گذشت هر دوره زمانی از پیش تعیین شده، هر  $X_i$  خود را با سلول  $CLA-EC_L$  متناظر در  $CLA-EC_L$  معاوضه می کند. در مرحله دوم از آزمایشات انواع مدل های دیگر CLA-EC را در محیط های پویا استفاده کردیم. نتایج حاصل در جدول (۲) نمایش داده شده است. همان طور که مشاهده می شود، کارایی الگوریتم CLA-EC استاندارد و CLA-EC ابتداء از آن را ندارند، هر چند که در [۳۲] در بیشتر موارد کارایی CLA-EC ناهمگام در مسائل با محیط ایستا بالاتر از CLA-EC استاندارد بوده است. و همین طور نشان داده شده است که در مسائل ایستا با این که CLA-EC<sub>L</sub> ناهمگام همگرا می شود ولی برای این کار به تعداد سلول بیشتری نیاز دارد و غیر این صورت در بهینه های محلی گیر سریعتر از CLA-EC استاندارد و ناهمگام مثل یکدیگر است و هیچ یک توانایی ورق پذیری با تغییرات محیط را ندارند.

الگوریتم دیگری که در اینجا مورد مقایسه قرار گرفته است، الگوریتم CLA-EC بازترکیبی است. این الگوریتم نسبت به الگوریتم CLA-EC استاندارد و CLA-EC ناهمگام کارایی بالاتری داشته و در نتیجه دارای خطای کمتری است. دلیل این امر، عملگر بازترکیبی است که به الگوریتم CLA-EC اضافه شده است. عملگر بازترکیبی قدرت اکتشاف الگوریتم را بالا می برد و همین طور اجازه می دهد که ساختارهای جزئی میان کروموزوم های افراد جایه جا شود. این تغییر در الگوریتم CLA-EC باعث می شود تا فضای خوبی جستجو شود و الگوریتم در بهینه خای محلی گیر نیفتد.

بهترین الگوریتمی که در میان الگوریتم های CLA-EC وجود دارد، الگوریتم CLA-EC همکارانه است. این الگوریتم دارای قدرت اکتشاف بسیار خوبی است و با تغییرات محیطی خود را وفق می دهد. البته در نسل های ابتدایی این الگوریتم نسبت به الگوریتم ژنتیک با جهش وفقی بد تر عمل می کند ولی در نسل های انتهایی از لحاظ کارایی به الگوریتم ژنتیک با جهش وفقی می رسد و کم کارایی آن بالاتر نیز می رود.

جدول (2) مقایسه خطای برون خطی روی تابع قله متحرک با الگوریتم ژنتیک با جهش وفقی و انواع مدل های CLA-EC بدون

حافظه در ۵۰۰۰ تکرار (الف) سناریو (ب) سناریو ۲

## 5. الگوریتم های جستجو/حافظه

در ادامه بحث الگوریتم های چند جمعیتی در این بخش برای بهبود الگوریتم CLA-EC میکنیم. این الگوریتم اولین بار در [۲۵] مورد استفاده قرار گرفت و به طور چشم گیری باعث بهبود عملکرد الگوریتم های ژنتیک با جهش و فقی شد. ایده اضافه کردن حافظه به الگوریتم های تکاملی به قدری کارا است که به وسیله بسیاری از نویسنده ها پیشنهاد شده است. دلیل این امر این است که وقتی که بهینه در یک مکان مشاهده می شود، حافظه می تواند آن مکان را به خاطر داشته باشد و جمعیت به سمت بهینه جدید حرکت داده شود. حافظه همچنین در نگه داری تنوع می تواند مفید باشد. وقتی تصمیم به اضافه کردن حافظه به الگوریتم کردیم، سوالی که مطرح می شود این است که از حافظه آشکارا یا ضمنی استفاده کنیم؟ در این مقاله از حافظه آشکارا استفاده کردیم به دلیل این که درک تاثیر آن ساده و بنابر آن چه که تا کنون گزارش شده است، حافظه ضمنی برای مسائلی با بیش از دو وضعیت مناسب نمی باشد. با توجه به این که از حافظه آشکارا استفاده خواهیم کرد سوالات زیر مطرح می شود:

چه هنگام و کدام یک از افراد جمعیت باید در حافظه ذخیره شوند؟

چه تعداد از افراد باید در حافظه ذخیره شوند و کدام یک باید جایگزین شوند تا فضای کافی برای افراد جدید به وجود آید؟

کدام یک از افراد باید از حافظه بازیابی شوند و دوباره به جمعیت اضافه شوند؟

در پاسخ به سوال اول باید گفت که، افرادی که در حافظه ذخیره می شوند باید دارای شایستگی ای بالاتر از میانگین شایستگی کل جمعیت داشته باشند، نباید بسیار قدیمی باشند، و باید در مکان های مختلف از فضای جستجو قرار گرفته باشند. پس به نظر می رسد که در هر زمان باید بهترین افراد در حافظه قرار گیرند.

در اینجا به عنوان استراتژی جایگزینی، گزینه های زیر مورد آزمایش قرار گرفته است:

- فرد آن از حافظه حذف می شود، هنگامی که ماکریم واریانس ( $i$ ) در حافظه باقی بماند. که واریانس کل برابر با جمع واریانس آلل ها در حافظه است.

$$V(i) = \sum_{i=1}^m \sum_{k \in M \setminus \{i\}} (x_{jk} - \bar{x}_j)^2 \quad (10)$$

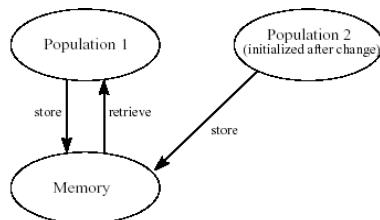
برای نگه داری تنوع، می توان دو فرد  $i$  و  $j$  با کمترین فاصله را در نظر گرفته و فردی که شایستگی کمتری دارد را جایگزین کرد. ایده مورد نظر این است که یک فرد در یک ناحیه مشخص کافی است و اگر دو فرد نزدیک به یک دیگر در یک ناحیه وجود دارند، تنها فرد شایسته تر باید باقی بماند (Mindist).

استراتژی دیگری که similar نامیده می شود به این صورت است که فرد جدید جایگزین شبیه ترین فرد قدیمی نسبت به خودش می شود.

بازیابی از حافظه تنها بعد از این که محیط تغییر کرد اتفاق می افتد. برای بازیابی افراد از حافظه  $n$  فرد جمعیت و  $m$  فرد موجود در حافظه ترکیب شده و تنها  $n$  نفر از بهترین افراد برای جمعیت جدید انتخاب می شوند.

## 6. رهیافت پیشنهادی

در این بخش الگوریتم اصلی مورد استفاده برای مناسب سازی CLA-EC به نام حافظه/جستجو<sup>۱۶</sup> مورد بررسی قرار گرفته است. رویکرد حافظه/جستجو دو مفهوم حافظه آشکارا و تنوع پذیری در مسیری شخص را ترکیب کرده است. آزمایشات نشان داده اند که اضافه کردن حافظه به الگوریتم های تکاملی ریسک گمراه کردن تکامل و جلوگیری از کشف نواحی جدید در فضای جستجو را دارد. شروع دوباره تکامل از ابتدا، هرگاه که تغییری در محیط روی می دهد به طور آشکار شناس خوبی برای پیدا کردن قله های جدید می دهد ولی ممکن است زمان طولانی برای پیدا کردن بهینه صرف شود و پرش های ناگهانی غیر ممکن است. در اینجا برای این که از مزیت هر دو ساختار استفاده کنیم، جمعیت CLA-EC را به دور زیر جمعیت تقسیم کرده ایم. جمعیت حافظه که مسئول نگه داری راه حل های خوب قدیمی و بهره برداری و تصحیح دانش موجود در فضای جستجو است. جمعیت دیگر که جمعیت جستجو نامیده می شود، مسئول کشف قله های جدید و ارائه آن ها به حافظه است ولی هرگز از حافظه اطلاعاتی بازیابی نمی کند. برای کشف قله های جدید جمعیت جستجو بعد از این که تغییری در محیط رخ داد به صورت تصادفی دوباره مقداردهی می شود. در شکل (۵) نمایی از الگوریتم جستجو/حافظه نمایش داده شده است.



شکل (5) نمایی از الگوریتم جستجو حافظه

در این روش ابتدا دو جمعیت CLA-EC هر یک با تعداد سلول ۴۵ را ایجاد می کنیم. سپس هر یک از این دو جمعیت به صورت موادی با هم بر روی محیط عمل می کنند و در پایان هر نسل یک کپی از بهترین سلول هر یک از CLA-EC ها در حافظه قرار می گیرد. سپس بعد از این که تغییری در محیط اتفاق افتاد، یکی از CLA-EC ها سلول های داخل حافظه را بازیابی کرده و آن ها را جایگزین بدترین سلول های خودش می کند و جمعیت CLA-EC دیگر اعمال تمام آنماها را یادگیر خود را به صورت تصادفی مقداردهی می کند. در شکل (6) شبه کد الگوریتم CLA-EC جستجو/حافظه نشان داده شده است.

```

Initialize
While not done do
    For each cell i in both  $CLA-EC_1$  and  $CLA-EC_2$  do in
        parallel
            Generate a new string genome
            Evaluate the new string genome
            If  $f(\text{new string genome}) > f(\text{old string genome})$  then
                Accept the new string genome
            End if
            Select  $S_e$  cells from neighbors of cell  $i$ 
            Generate the reinforcement signal vector
            Update LAs of cell  $i$ 
            If cell  $i$  is the fittest cell of generation &
            If  $\text{memory\_counter} < \text{max\_memory\_size}$ 
                Store cell  $i$  in memory
            Else if  $\text{memory\_counter} > \text{max\_memory\_size}$ 
                Store cell  $i$  with replacement strategy
            End if
        End parallel for
        If environment change
            Insert memory_population in to  $CLA-EC_1$ 
            Randomly reinitial action of Las of  $CLA-EC_2$ 
        End if
    End while

```

شکل (6) الگوریتم CLA-EC جستجو/حافظه

جدول (3) مقایسه خطای برون خطی روی تابع قله متحرک با الگوریتم ژنتیک Mem\_Search با جهش وفقی و انواع مدل های ۲ تکرار (الف) سناریو (ب) سناریو ۵۰۰۰

Memo ry schem e	SEA	M/S SEA	M/S CLA- EC	M/S Cooperative CLA-EC	M/S Recombinativ e CLA-EC	M/S Uniform CLA-EC
Mindis t	۱۵.۴۵۵۸	۳.۳۵۵۰۲	۴.۵۸۵۷۲۷	۱۱.۲۰۴۰۵	۱.۹۳۳۵۷۳	۴.۷۵۵۵۶۹
Simila r	۱۶.۲۴۵۷	۸.۱۳۸۸۱۳	۴.۵۸۵۷۲۷	۱۱.۰۷۹۷۴	۱.۹۳۳۵۷۳	۴.۷۵۵۵۶۹

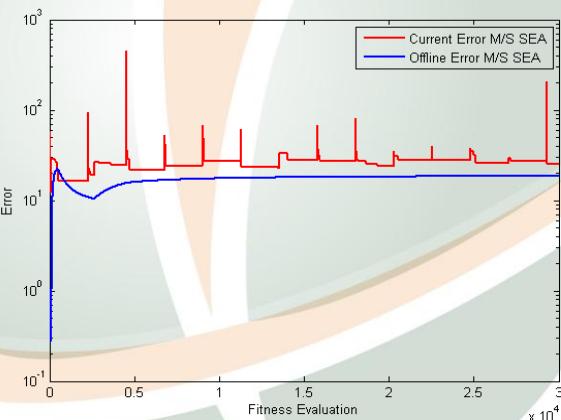
<b>Varian ce</b>	۱۷.۳۳۸۵	۳.۵۹۴۴۷۳	۶.۸۰۲۸۱۴	۱۱.۰۷۹۷۴	۱.۹۴۳۷۲	۵.۴۳۷۰۰۹
------------------	---------	----------	----------	----------	---------	----------

همان طور که در جدول (۳) مشاهده می شود روش CLA-EC جستجو/حافظه بازتر کیمی دارای بالاترین کارایی است و میزان خطای خوبی رسانده است. ولی استفاده از مکانیزم جستجو/حافظه برای CLA-EC همکارانه و همکارانه بهبود یافته تاثیر خوبی نداشته است.

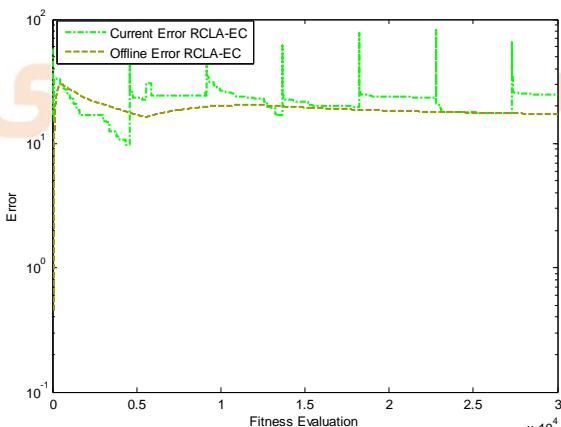
## 7. بررسی سرعت همگرایی در الگوریتم CLA-EC همکارانه و الگوریتم ژنتیک با جهش و فرقه

در این بخش هدف این است که با بررسی سرعت همگرایی به این نتیجه برسیم کدام الگوریتم پس از بروز تغییراتی در سیستم می تواند سریعتر بهترین پاسخ خود را تولید نماید و آنرا در اختیار سیستم قرار دهد که دارای کمترین خطای باشد. این پارامتر در سیستم های پویا به دلیل شرایط خاص این محیط ها از اهمیت بالایی برخوردار است اما به دلیل پیچیدگی هایی که دارد کمتر مورد بررسی قرار گرفته است. برای این منظور از میزان خطای جاری که توسطتابع استفاده شده است. در هر بار گردش برنامه این میزان خطای در متغیری ذخیره شده سپس این مقادیر بصورت نموداری نشان داده خواهند شد. هر چه این مقدار خطای کمتر بوده و به صفر نزدیکتر باشد به آن معنی است که قله یافته شده به بلندترین قله نزدیکتر است، حال هر الگوریتمی که بتواند با شبیه تندری به کمترین مقدار تولیدی توسط الگوریتم برسد می توان نتیجه گرفت سرعت این الگوریتم بهتر بوده و می تواند بعد از بروز تغییرات جوابهای بهتری را با سرعت بیشتر تولید نماید.

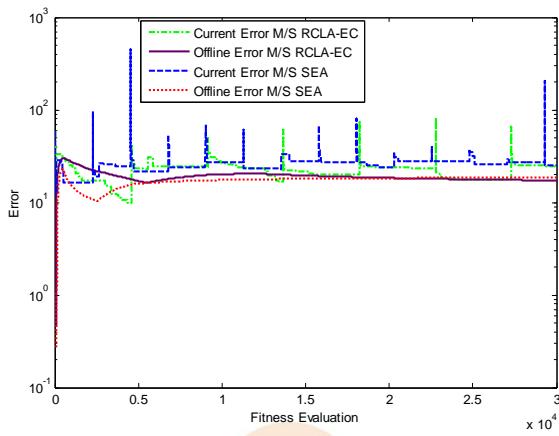
در شکل (۷) (به بررسی خطای جاری و Offline Memory/search SEA) پرداخته ایم. در هر یک از نمودارها میزان خطای جاری و Offline در یک گردش سی هزار بار برای ارزیابی برآش تابع نشان داده شده است، پالس های نامنظم نشان دهنده ایجاد یک تغییر و حرکت برای رسیدن به جواب است تا بروز تغییر بعدی، این عمل در هر ۵۰۰۰ بار ارزیابی تابع یکبار رخ می دهد که بصورت بازه هایی در شکل ها نشان داده شده است. در شکل (۹) نیز خطای برون خطی و جاری الگوریتم های M/S SEA و M/S CLA-EC مقایسه شده است.



شکل (۷) ارزیابی خطای جاری و Offline برای M/S SEA در F=۵۰۰۰ ارزیابی اول



شکل (۸) ارزیابی خطای جاری و Offline برای M/S CLA-EC در F=۵۰۰۰ ارزیابی اول



شکل (9) مقایسه خطای الگوریتم M/S CLA-EC و M/S SEA برای ۳۰۰۰۰ ارزیابی اول

همان طور که در شکل (9) نشان داده شده است میزان خطای جاری الگوریتم M/SCLAE-EC در اکثر موارد در زیر نمودار SEA حرکت می کند بنابراین میزان خطای بروز خطی آن با شیب بیشتری به سمت صفر حرکت کرده و بنابراین دارای کارایی بالاتری است.

## 8. نتیجه گیری

در این مقاله مدل محاسبات تکاملی بر پایه اتوماتای یادگیر سلولی معرفی شد و مورد بررسی قرار گرفت. این مدل برای حل مسئله قله های متجرک به کار گرفته شد. نتایج آزمایشات صورت گرفته بر روی این مدل نشان دادند که مدل CLA-EC در حل مسایلی با فضای حالت بزرگ به ویژه زمانی که فضای جستجو خلوت باشد، عملکرد مناسبی ندارد و نسبت به الگوریتم ژنتیک با جهش وقیعی بدتر عمل میکند. نکته قابل توجه این است که ایجاد تغییرات گسترده در ساختار ژنومها در هر گام، به دلیل ماهیت این مدل است. در الگوریتمهای ژنتیک با انتخاب روش‌های درست بازترکیبی و جهش، سعی میشود که تغییرات ژنومها، بسیار نرم و به آهستگی صورت گیرند. به عنوان مثال، در روش بازترکیبی تک نقطه ای، بخش از ژنومهای والد، ثابت میمانند و فقط قسمتی از ژنوم تغییر میکند. ولی در مدل CLAEC، پیش از همگرا شدن اتوماتای یادگیر سلولها، در هر گام کل ژنوم به صورت تصادفی تغییر میکند. این مساله باعث میشود که همگرا شدن الگوریتم نرخی بسیار کند. داشته باشد و در مقایسه با الگوریتم ژنتیک بدتر عمل کند. مطالعات انجام شده نشان داد که الگوریتم CLA-EC استاندارد، بازترکیبی، CLA-EC آسنکرون دارای عملکرد خوبی نیستند ولی CLA-EC همکارانه به دلیل توازنی که میان خاصیت اکتشاف و بهره برداری ایجاد می کند، دارای کارایی بالاتر است. الگوریتم های CLA-EC استاندارد، بازترکیبی و آسنکرون به دلیل این که نرخ همگرای اشان بسیار کند قدرت ورق پذیری با تغییرات محیط را ندارند. ولی در الگوریتم CLA-EC همکارانه نرخ همگرای افزایش پیدا کرده و نسبتاً بهتر می توانند با تغییرات محیط تطبیق پیدا کنند. در واقع CLA-EC همکارانه شامل یک حافظه ضمنی است که باعث نگه داری تنوع در جمعیت و تطبیق بهتر با تغییرات می شود. در گام دوم یک مدل جدید با نام Memory/search CLA-EC معرفی شد که نسبت به مدل های ارائه شده دارای کارایی بهتری است. از آزمایشات انجام شده به این نتیجه می رسیم که اگر بهینه به طور مکرر به مکان قبلی باز گردد برای رویکرد مبتنی بر حافظه مناسب خواهد بود. استفاده از استراتژی مبتنی بر حافظه مناسب در این کار دارای اهمیت است. برای این که بتوانیم از مزایای حافظه آشکاری که به الگوریتم CLA-EC اضافه کردیم استفاده کنیم باید از یک استراتژی جایگزینی مناسب استفاده کنیم.

## 9. مراجع

- [1] Louis, S. J., and XU, Z., "Genetic Algorithms for Open Shop Scheduling and Re-scheduling," In M. E. Cohen and D. L. Hudson , Editors, ISCA 11<sup>th</sup> International Conference on Computers and Their Application, pp. ۹۹-۱۰۲, ۱۹۹۶.
- [2] Cartwright, H. M., and Tusen, A. L., "Genetic Algorithms and Flow Shop Scheduling: Towards the Development of a Real-time Process Control System," T. C. Fogarty, Editor, AISB Workshop on Evolutionary Computing, vol. ۸۶۵ of LNCS, pp. ۲۷۷-۲۹۰, Springer Verlag, ۱۹۹۴.
- [3] Bierwirth, C., and Kopfer, H., "Dynamic Task Scheduling with Genetic Algorithms in Manufacturing Systems," Technical Report, Department of Economics, University of Bremen, Germany, ۱۹۹۴.
- [4] Bierwirth, C., Kopfer, H., Mattfeld, D. C., and Rixen, I., "Genetic Algorithm Based Scheduling in a Dynamic Manufacturing Environment," Proceedings of IEEE Conference on Evolutionary Computation, IEEE press, ۱۹۹۵.

- [5] Pico, C. A. G., and Wainwright, R. L., "Dynamic Scheduling of Computer Tasks Using Genetic Algorithms," Proceedings of First IEEE Conference on Evolutionary Computation, vol. 2, pp. 829-832, 1994.
- [6] Fang, H. -L., Ross, P., and Corn, D., "A Promising Genetic Algorithm Approach to Job-shop Scheduling, and Open-shop Scheduling Problems," In S. Forrest, Editor, Fifth International Conference on Genetic Algorithms, pp. 375-382, 1993.
- [7] Krishnakumar, K., "Micro Genetic Algorithms for Stationary and Non-stationary Function Optimization," In intelligent control and adaptive systems, of the SPIE, vol. 1196, pp. 289-296, 1989.
- [8] Cobb, H. G., "An Investigation into the Use of Hyper Mutation as an Adaptive Operator in Genetic Algorithms Having Continuous, Time-dependent Non stationary," Technical Report AIC-90-001, proc in. Naval Research Laboratory, Washington, USA, 1990.
- [9] Grefenstette, J. J., "Genetic Algorithms for Changing Environments," In R. Maenner and B. Manderick, Editors, Parallel Problem Solving From Nature 2, pp. 137-144, North Holland, 1992.
- [10] Cobb, H. G., and Grefenstette, J. J., "Genetic Algorithms for Tracking Changing Environment," Proceedings of 5<sup>th</sup> International Conference on Genetic Algorithms, pp. 523-530, 1993.
- [11] Vavak, F., Jukes, K. A., and Fogarty, T. C., "Performance of a Genetic Algorithms with Variable Local Search Range Relative to Frequency for the Environmental Changes," Proceedings of Conference on Genetic Programming, 1998.
- [12] Vavak, F., Fogarty, T. C., and Jukes, K., "Learning the Local Search Range for Genetic Control of Dynamic Systems," Proceedings of Pre-Conference Workshop on Evolutionary Computing and Machine Learning, pp. 143-150, 1996.
- [13] Back, T., "On the Behavior of Evolutionary Algorithms in Dynamic Environments," Proceedings of IEEE International Conference on Evolutionary Computation, pp. 446-451. IEEE, 1998.
- [14] Ng, K. P., and Wong, K. C., "A New Diploid Scheme and Dominance Change Mechanism for Non-stationary Function Optimization," Proceedings of 6<sup>th</sup> international conference on genetic algorithms, pp. 159-166, 1995.
- [15] Ryan, C., "Diploidy Without Dominance," In J. T. Alander, Editor, Third Nordic Workshop Mon Genetic Algorithms, pp. 53-56, 1997.
- [16] Dasgupta, D., and McGregor, D. R., "Non stationary Function Optimization Using the Structured Genetic Algorithm," In R. Manner and B. Manderick, Editors, Parallel Problem Solving From Nature, pp. 145-154, Elsevier science publisher, 1992.
- [17] Dasgupta, D., "Incorporating Redundancy and Gene Activation Mechanisms in Genetic Search," In L. chambers, Editor, Practical Handbook of Genetic Algorithms, vol. 2, pp. 203-216, CRC Press, 1995.
- [18] Ohkura, K., and Ueda, K., "Adaptation in Dynamic Environment Using Genetic Algorithms with Redundant Representation and Additional Genetic Operators," In Dagli and Cihan, Editors, Intelligent Engineering Systems through Artificial Neural Networks, pp. 291-298, 1994.
- [19] Trojanowski, K., and Michalewicz, Z., "Searching for Optima in Non-stationary Environments," Proceedings of Congress on Evolutionary Computation, vol. 3, pp. 1843-1850, IEEE, 1999.
- [20] Trojanowski, K., and Michalewicz, Z., "Searching for Optima in Non-stationary Environments," Proceedings of Congress on Evolutionary Computation, vol. 3, pp. 1843-1850, IEEE, 1999.
- [21] Goldberg, D. E., and Richardson, J., "Genetic Algorithms With Sharing for Multimodal Function Optimization," Proceedings of Second International Conference on Genetic Algorithms, pp. 41-49, 1987.
- [22] Andersen, H. C., "An Investigation in to Genetic Algorithms, and the Relationship between Speciation and the Tracking of Optima in Dynamic Functions," Honours thesis, Queensland university of technology, Brisbane, Australia, November 1991.
- [23] Hocaglu, C., and Sanderson, A. C., "Planning Multi-paths Using Speciation in Genetic Algorithms," Proceedings of 4<sup>rd</sup> IEEE Intl. Conf. on Evolutionary Computation, pp. 378-383, 1996.
- [24] Cedeno, W., and Vemuri, V. R., "On the Use of Niching for Dynamic Landscapes," Proceedings of International Conference on Evolutionary Computation, IEEE, 1997.

- [25] Liles, W., and Jong, D., "The Usefulness of Tag Bits in Changing Environments," Proceedings of Congress on Evolutionary Computation, vol. ۳, pp. ۲۰۵۴-۲۰۶۰. IEEE, ۱۹۹۹.
- [26] Spears, W., "Simple Subpopulation Schemes," Proceedings of evolutionary programming conference, pp. ۲۹۶-۳۰۷, World Scientific, ۱۹۹۴.
- [27] Oppacher, F., and Wineberg, M., "The Shifting Balance Genetic Algorithm: Improving the GA in a Dynamic Environment," Proceedings of Genetic and Evolutionary Computation Conference, vol. ۱, pp. ۵۰۴-۵۱۰, ۱۹۹۹.
- [28] Wineberg, M., and Oppacher, F., "Enhancing the GA's Ability to Cope with Dynamic Environments," Proceedings of Genetic and Evolutionary Computation Conference, pp. ۳-۱۰, ۲۰۰۰.
- [29] Ursem, R. k., "Multinational GA Optimization Techniques in dynamic Environments," Proceedings of Genetic and Evolutionary Computation Conference, pp. ۱۹-۲۶, ۲۰۰۰.
- [30] Stephens, C. R., Olmedo, I. G., Vargas, J. M., and Waelbroeck, H., "Self-adaptation in Evolving Systems," In Artificial life, ۶(۲), ۱۹۹۸.
- [31] Weiker, K., and Weiker, N., "Dynamic Rotation Partial Visibility," Proceedings of congress on evolutionary computation, pp. ۱۱۲۵-۱۱۳۱, ۲۰۰۰.
- [32] Masoodifar, B., Meybodi, M. R., and Rastegar, R., "Asynchronous CLA-EC," Proceedings of ۱۱<sup>th</sup> Annual CSI Computer Conference of Iran, Fundamental Science Research Center (IPM), Computer Science Research Lab., Tehran, Iran, pp. ۴۴۷-۴۵۸, Jan. ۲۴-۲۶, ۲۰۰۶.
- [33] Masoodifar, B., Meybodi, M. R., and Hashemi, M., "Cooperative CLA-EC," Proceedings of ۱۲<sup>th</sup> Annual CSI Computer Conference of Iran, Shahid Beheshti University, Tehran, Iran, pp. ۵۵۸-۵۵۹, Feb. ۲۰-۲۲, ۲۰۰۷.
- [34] Jafarpourand, B., and Meybodi, M. R., "Recombinative CLA-EC," Proceedings of Annual IEEE International Conference on Tools with Artificial Intelligence (ICTAI ۲۰۰۷), pp. ۴۱۵-۴۲۲, Achaia, Greece, Oct. ۲۰۰۷.
- [35] Branke, J., "Memory-Enhanced Evolutionary Algorithms ,," Congress on Evolutionary Computation (CEC'۹۹), IEEE, vol.۳ pp. ۱۸۷۵-۱۸۸۲, ۱۹۹۹.

زیرنویس ها

- ` Restriction
- ˘ Optimum
- ˘ Converge
- ˘ Diversity
- Exploring
- ˘ Modifying selection
- ˘ Multi-population approaches
- ˘ Dynamic bit-matching
- ˘ Moving parabola
- ˘ Time-varying knapsack problem
- ˘ Moving peaks function
- ˘ Scheduling problem
- ˘ Oscillating peaks
- ˘ Time-driven
- ˘ Step-driven
- ˘ Memory/Search