الگوریتم فراابتکاری دسته والد-فرزند مبتنی بر حافظه و خوشه‌بندی جهت بهینه‌سازی پویا

محسن مرادی[۱]، صمد نجاتیان*[۲]، حمید پروین[۳]، کرم‌الله باقری‌فرد[۴] و وحیده رضایی[۵]

[۱و۴]گروه کامپیوتر، واحد یاسوج ، دانشگاه آزاد اسلامی، یاسوج، ایران

[۲]گروه برق، واحد یاسوج، دانشگاه آزاد اسلامی، یاسوج، ایران

[۳]گروه کامپیوتر، واحد نورآباد ممسنی، دانشگاه آزاد اسلامی، نورآباد ممسنی، ایران

[۵] گروه ریاضی، واحد یاسوج، دانشگاه آزاد اسلامی، یاسوج، ایران

چکیده

تاکنون روش‌های مختلفی برای بهینه‌سازی ارایه شده است و یکی از معروف‌ترین روش‌های بهینه‌سازی، الگوریتم‌های هوش‌جمعی هستند. بسیاری از مسائل بهینه‌سازی اخیر در دنیای واقعی طبیعت پویا دارند؛ بنابراین، الگوریتم بهینه‌سازی برای حل مسائل در محیط‌های پویا مورد نیاز است. الگوریتم دستهٔ والد-فرزند مبتنی بر حافظه و خوشه‌بندی (CMPCS)، گونه‌ای از الگوریتم‌های هوش جمعی و برگرفته شده از طبیعت است، که در این مقاله ارایه شده است. این روش به رفتار فردی و گروهی وابسته است، در این الگوریتم برای افزایش کارآیی از یک حافظه با خوشه‌بندی و دافعه استفاده شده است. روش CMPCS پیشنهاد شده بر روی محک قله‌های متحرک (MPB) آزمایش شده است. MPB یک محک خوب برای ارزیابی کارایی الگوریتم‌های بهینه‌سازی در محیط‌های پویا است. نتایج تجربی در MPB نشان می‌دهد که روش پیشنهادی CMPCS کارایی مناسب‌تری نسبت به روش‌های دیگر حل مسائل بهینه‌سازی پویا دارد.

واژگان‌کلیدی: بهینه‌سازی پویا، محیط‌های پویا، حافظه، محک قله‌های متحرک.

Clustering and Memory-based Parent-Child Swarm Meta-heuristic Algorithm for Dynamic Optimization

Mohsen Moradi[1], Samad Nejatian*[2], Hamid Parvin[3], Karamolla Bagherifard[4] & Vahideh Rezaie[5]

[1,4]Department of Computer Engineering Yasooj Branch, Islamic Azad University, Yasooj, Iran

[2]Department of Electrical Engineering, Yasooj Branch, Islamic Azad University, Yasooj, Iran

[3]Department of Computer Engineering, Nourabad Mamasani Branch, Islamic Azad University, Nourabad Mamasani,Fars, Iran

[5]Department of Mathematics, Yasooj Branch,Islamic Azad University, Yasooj, Iran

Abstract

In the real world, we face some complex and important problems that should be optimized, most of the real-world problems are dynamic. Solving dynamic optimization problems are very difficult due to possible changes in the location of the optimal solution. In dynamic environments, we are faced challenges when the environment changes. To respond to these changes in the environment, any change can be considered as the input of a new optimization problem that should be solved from the beginning, which is not suitable because it is time consuming. One technique for improving optimization and learning in dynamic environments is by using information from the past. By using solutions from previous environments, it is often easier to find promising solutions in a new environment. A common way to maintain and exploit information from the past is the use of memory, where solutions are stored periodically and can be retrieved and refined at the time that the environment changes. Memory can help search respond quickly and efficiently to change in a dynamic problem. Given that a memory has a

* نویسندهٔ عهده‌دار مکاتبات

* Corresponding author

● تاریخ ارسال مقاله: ۱۳۹۸/۳/۷ ● تاریخ پذیرش: ۱۳۹۸/۱۱/۲ ● تاریخ انتشار: ۱۴۰۰/۱۰/۲۷ ● نوع مطالعه: پژوهشی

finite size, if one wishes to store new information in the memory, one of the existing entries must be discarded. The mechanism used to decide whether the candidate entry should be included in the memory or not, and if so, which of the old entries should be replaced it, is called the replacement strategy. This paper explores ways to improve memory for optimization and learning in dynamic environments. In this paper, a memory with clustering and new replacement strategy for storing and restoring memory solutions has been used to enhance memory performance. The evolutionary algorithms that have been presented so far have the problem of rebuilding populations when multiple populations converge to an optimum. For this reason, we proposed algorithm with exclution mechanism that have the ability to explore the environment (Exploration) and extraction (Explitation). Thus, an optimization algorithm is required to solve the problems in dynamic environments well. In this paper, a novel collective optimization algorithm, namely the Clustering and Memory-based Parent-Child Swarm Algorithm (CMPCS), is presented. This method relies on both individual and group behavior. The proposed CMPCS method has been tested on the moving peaks benchmark (MPB). The MPB is a good Benchmark to evaluate the efficiency of the optimization algorithms in dynamic environments. The experimental results on the MPB reveal the appropriate efficiency of the proposed CMPCS method compared to the other state-of-the-art methods in solving the dynamic optimization problems.

Keywords: Dynamic Optimization, Dynamic Environments, Memory, Moving Peaks Benchmark.

۱- مقدمه و پیشینه

در دنیای واقعی به‌طورمعمول با مسائلی پیچیده و مهمی روبه‌رو هستیم که می‌بایست بهینه شوند. به‌طور کلی، مسائل بهینه‌سازی به دو گروه اساسی تقسیم می‌شوند: مسائل بهینه‌سازی ایستا و مسائل بهینه‌سازی پویا. در مسائل بهینه‌سازی ایستا، بهینه مسأله ثابت بوده و تغییر نمی‌کند. در مسائل پویا، بهینه مسأله در طول زمان تغییر می‌کند، به‌طوری که ردیابی بهینه برای الگوریتم کار دشواری خواهد بود و این باعث می‌شود، این مسائل نیازمند الگوریتم‌های بهینه‌سازی باشند، که علاوه‌بر یافتن بهینه، بتوانند بهینه‌های متغیر را دنبال کند. همچنین در محیط‌های پویا با سه چالش اساسی روبه‌رو هستیم: نخستین چالش، شناسایی زمان تغییرات محیط است، به‌طوری‌که الگوریتم بتواند به‌موقع پاسخ مناسبی به این تغییرات از خود نشان دهد و در کمترین زمان به‌سمت بهینه هدایت شود. دومین چالش، حافظه منسوخ‌شده ذرات است؛ زیرا با تغییر محیط بهترین تجربه شخصی یک ذره و بهترین تجربه جمعی یک گروه، ممکن است دیگر معتبر و صحیح نباشد. سومین چالش، ازدست‌دادن تنوع[1] است؛ زیرا متنوع‌سازی یک گروه هم‌گراشده برای یافتن بهینه متحرک و سپس هم‌گرایی آن به بهینه مورد نظر به‌شدت کارایی الگوریتم را کاهش می‌دهد؛ بنابراین هرگاه یک تغییر در هدف، نمونه یا محدودیت یک مسأله بهینه‌سازی رخ دهد، ممکن است، بهینه آن مسأله تغییر کند. برای روبه‌روشدن با این پویایی و واکنش به تغییرات محیط، می‌توان هر تغییر را به‌عنوان ورودی یک مسأله بهینه‌سازی جدید در نظر گرفت، که می‌بایست از ابتدا حل شود، که این روش مناسب نیست و زمان‌بر است. یک راه دیگر برای بهینه‌سازی پس از تغییرات محیط، استفاده از اطلاعات مربوط به فضای جستجوی قبلی برای پیشروی در جستجو پس از تغییر است؛ بنابراین با استفاده از حافظه، راه‌حل‌های فضای مسأله به‌صورت دوره‌ای ذخیره می‌شوند و در صورت نیاز یا تغییر محیط، دوباره بازیابی می‌شوند، که این روش کمک فراوانی به جستجوی سریع مسائل بهینه‌سازی می‌کند. همچنین در بهینه‌سازی مسائل پیچیده، استفاده از روش‌های بهینه‌سازی دقیق غیرممکن و فاقد کارایی است. در حل مسائل بهینه‌سازی نیاز به روشی است که به‌صورت هوشمندانه عمل کند و درنهایت به جواب بهینه برسد، برای این منظور از روش‌های جستجوی تصادفی برای رسیدن به یک پاسخ نزدیک به بهینه استفاده می‌شود [1, 2].

در میان روش‌های جستجوی تصادفی، الگوریتم‌های تکاملی برگرفته از طبیعت دارای جایگاه ویژه‌ای هستند. تاکنون الگوریتم‌های مختلفی برای بهینه‌سازی ارایه شده است، اما در این مقاله به نکاتی می‌پردازیم، که باعث شد یک الگوریتم بهینه‌سازی فراابتکاری جدید ارایه کنیم.

۱. الگوریتم‌های بهینه‌سازی بایستی بتواند با کمترین خطا به سمت بهینه هم‌گرا شوند و بهینه محلی و سراسری را دنبال کنند و تنوع را در تمام اجرا برقرار نمایند.
۲. استفاده از حافظه و سازوکار جایگزینی حافظه (عمل ذخیره و بازیابی راه‌حل‌ها در حافظه)، می‌تواند باعث افزایش کارایی الگوریتم‌های بهینه‌سازی شود.

[1] Diversity

۳. در محیط‌هایی که چندین بهینه وجود دارد، نیاز به الگوریتمی است که بتواند چند جمعیتی باشد و چندین بهینه را دنبال کند.
۴. استفاده از سازوکار انحصار در زمان هم‌گراشدن چند دسته به یک بهینه (حذف دسته با برازش کمتر و راه‌اندازی مجدد آن) کارایی الگوریتم را کاهش می‌دهد.

بنابراین در این مقاله یک الگوریتم بهینه‌سازی چندجمعیتی بر اساس دسته والد-فرزند ارایه شده، که دارای سازوکار دافعه است؛ به‌طوری که دسته‌ها یکدیگر را دفع می‌کنند و این باعث جلوگیری از راه‌اندازی دوباره می‌شود؛ همچنین در این الگوریتم، از حافظه برای نگهداری اطلاعات فضای جستجوی قبلی استفاده شده و جهت افزایش کارایی حافظه و حفظ تنوع، از یک سازوکار جایگزینی جدید بهره گرفته شده است. در این مقاله جهت آزمایش‌ها، از محک قله‌های متحرک استفاده شده است و نتایج آزمایش‌ها بر روی محک قله‌های متحرک نشان می‌دهد که الگوریتم پیشنهادی در حل مسائل بهینه‌سازی پویا دارای کارایی قابل قبولی است؛ به‌طوری که سریع‌تر به سمت بهینه هم‌گرا می‌شود و به‌دلیل تنوع زیاد، به‌طورتقریبی بیش‌تر بهینه‌ها مورد پوشش قرار می‌گیرند، و وجود دافعه باعث افزایش سرعت الگوریتم می‌شود. در ادامه به برخی از روش‌ها بهینه‌سازی انجام‌شده، پرداخته می‌شود. در [3]، یک مدل چندجمعیتی ازدحام ذرات (FTMPSO[1]) برای محیط‌های پویا ارایه شده است. در این روش، هدف رسیدن به تنوع بیشینه در محیط است. در [4]، یک الگوریتم به‌نام CESO[2] پیشنهاد شده است، این روش از دو جمعیت برای شناسایی و دنبال‌کردن بهینه استفاده می‌کند، که یکی مسئول حفظ تنوع و دیگری مسئول پیداکردن بهینه سراسری است. در [5]، یک الگوریتم چندجمعیتی مبتنی بر الگوریتم کرم شب‌تاب ارایه شده است. در این روش از نگاشت آشوب برای تولید جمعیت اولیه استفاده شده است. در [6]، یک روش مبتنی بر بهینه‌سازی جمعی ذرات ارایه شده، که در این الگوریتم از جابه‌جایی تصادفی ذرات برای حفظ تنوع استفاده شده است. در [7]، یک الگوریتم مبتنی بر ازدحام ذرات ارایه شده است؛ در این روش از حافظه صریح و سازوکار به‌روزرسانی حافظه برای ردیابی سریع بهینه استفاده شده است. در [8]، یک الگوریتم ازدحام ذرات مبتنی بر خوشه‌بندی ارایه شده است، در این الگوریتم ذرات به خوشه‌هایی تقسیم می‌شوند و هر ذره در خوشه مربوط به خود به جستجوی محلی می‌پردازد. در [9]، یک روش بهینه‌سازی جمعی ذرات مبتنی بر اتوماتای سلولی به نام CellularPSO پیشنهاد شده است. ایده اصلی در این رویکرد، بهره‌گیری از تعاملات محلی در آتوماتای سلولی[3] (CA) و تقسیم‌کردن جمعیت ذرات در داخل سلول‌های آتوماتای سلولی است. هر گروه برای یافتن بهینه محلی تلاش می‌کند، که این کار باعث کشف بهینه سراسری می‌شود. در [10]، یک الگوریتم تطبیقی Adaptive mQSO ارایه شده است. در الگوریتم A mQSO تعداد دسته‌ها از ابتدا معین نیست و با تغییر در محیط و پیداکردن قله‌های جدید، تعداد دسته‌ها افزایش می‌یابد. در این الگوریتم، از یک عملگر ضد هم‌گرایی در زمان هم‌گراشدن تمام دسته‌ها استفاده می‌کند، به‌طوری که یک دسته آزاد جدید ایجاد می‌کند، که به پیداکردن بهینه محلی جدید کمک می‌کند. در [11]، الگوریتم‌های mQSO و FMSO ارایه شده است، که نوع خاصی از الگوریتم ازدحام ذرات برای محیط‌های پویا است. در این الگوریتم، ذرات به دو دسته ذرات خنثی و ذرات کوانتوم تقسیم می‌شوند. در این الگوریتم، کل جمعیت به چند گروه تقسیم می‌شوند و شامل سه عمل‌گر تنوع با نام‌های ذرات کوانتوم، دفع و ضد هم‌گرایی است. در [12]، الگوریتم بهینه‌سازی ذرات چنددسته‌ای[4] (MPSO) ارایه شده است. در این الگوریتم، بیشینه تعداد دسته فعال در محیط محدود شده و در هر فاز، دسته‌ای که دارای مقدار برازش بهتر است، فعال می‌شود. این الگوریتم دارای عملیات‌های مختلف از جمله عملیات ضدهم‌گرایی، گروه غیرفعال، شعاع حذف و گروه فعال است. در [13]، الگوریتم بهینه‌سازی کلونی زنبور مصنوعی یادگیر جامع[5] (CLABC) ارایه شده است. که یک مدل ABC بسیار منطقی (حساس) است. انگیزه اصلی CLABC، توانگرکردن رفتارهای کاوش زنبور عسل مصنوعی در مدل ABC با ترکیب اولیه جمعیت بر اساس روش مربع متعامد، روش جستجوی الگوی پائول، چرخه زندگی و استراتژی یادگیری اجتماعی مبتنی بر تزویج است. در [14]، یک الگوریتم ABC چنددسته‌ای ارایه شده است. ABC، یک الگوریتم چندجمعیتی است که، تنوع را حفظ می‌کند و اکتشاف‌گرا است. در الگوریتم ABC چنددسته‌ای، تعداد دسته‌ها در گذر زمان تغییر می‌کنند.

[1] Finder–tracker multi-swarm PSO
[2] Collaborative Evolutionary-Swarm Optimization
[3] Cellular Automata
[4] Multiswarm Particle Swarm Optimization algorithm
[5] Comprehensive Learning Artificial Bee Colony Optimizer

این الگوریتم شامل یک طرح پاک‌سازی جمعیت برای حل مسائل بهینه‌سازی پویا است. در [15]، یک الگوریتم مبتنی بر ماهی مصنوعی[1] به نام mNAFSA پیشنهاد شده است. در این الگوریتم، رفتارها، پارامترها و فرآیندها اجرای الگوریتم استاندارد ماهی مصنوعی استاندارد (AFSA) تغییر کرده‌اند تا الگوریتم پیشنهادی بتواند چندین قله را کشف و پس از تغییر محیط، آنها را دنبال کند. در [16]، یک روش مبتنی بر حافظه برای حل مسائل بهینه‌سازی پویا پیشنهاد شده است. در این روش، یک حافظه با ترکیبی از سه الگوریتم: (۱) الگوریتم ژنتیک، (۲) الگوریتم بهینه‌سازی دسته ذرات و (۳) جستجوی محلی تپه‌نوردی استفاده شده که از حافظه پیشنهادی برای حفظ سطح تنوع مناسب استفاده شده است. در [17]، یک الگوریتم مبتنی بر گونه ارایه شده که در آن از رفتار تغذیه و به اشتراک‌گذاری اطلاعات در کلونی زنبور عسل بهره گرفته شده است. در [18]، یک الگوریتم ژنتیک با حافظه پیشرفته (MEGA) ارایه شده که این الگوریتم، ترکیبی از حافظه با الگوریتم ژنتیک است. در [19]، یک روش به‌نام CPSO[2] ارایه شده است، که در این روش ذرات به خوشه‌هایی تقسیم می‌شوند و در هر خوشه، یک جستجوی محلی صورت می‌پذیرد. در [20]، یک الگوریتم چنددسته‌ای ازدحام ذرات مبتنی بر ردیاب و یابنده ارایه شده است. که در این روش برای کنترل دسته‌ها از سازوکار یابنده و دنبال‌کننده (ردیاب) استفاده شده است. به‌طوری که شامل یک دسته یابنده و چندین دسته ردیاب است. دسته یابنده، دارای تعداد ذرات بیشتری نسبت به دسته ردیاب و مسئول یافتن بهینه‌ها (قله‌ها) است. هنگامی که یابنده به یک بهینه هم‌گرا می‌شود، یک دسته ردیاب را فعال می‌کند تا در موقعیت یابنده جایگزین شود. پس از فعال‌سازی و قرارگرفتن در بهینه، دسته ردیاب، بهینه را پوشش می‌دهد و مسئول پیگیری این بهینه پس از تغییر محیط است. در [21]، یک مدل همکاری ترکیبی جمعیتی مبتنی بر تکامل تفاضلی ازدحام و بهینه‌سازی ذرات پیشنهاد شده است. در این روش، یک جمعیت با استفاده از تکامل تفاضلی ازدحام، مسئول مکان‌یابی چندین ناحیه امیدبخش از فضای جستجو و حفظ تنوع در طول اجرای است. جمعیت‌های دیگر با استفاده از بهینه‌سازی ذرات، عمل بهره‌برداری اطراف بهترین موقعیت یافت‌شده را انجام می‌دهند. در [22]، یک الگوریتم بهینه‌سازی چنددسته‌ای مبتنی بر خواب پیشنهاد شده که این روش از یک دسته والد و تعدادی دسته فرزند تشکیل شده است. دسته والد مسئول پیداکردن نواحی امیدبخش در فضای جستـجو است، که بر اساس آن یک دسته فرزند جهت بهره‌برداری از نواحی جدید امیدبخش ایجاد می‌شود. الگوریتم پیشنهادی از خواب زمستانی حیوانات برگرفته شده است. در الگوریتم پیشنهادی، هر دسته فرزند به‌عنوان یک جستجوگر غذا در نظر گرفته می‌شود و تا زمانی که یک راه‌حل بهتر پیدا شود، فعال می‌ماند و همچنان جستجو می‌کند. با این حال، اگر نتواند یک راه‌حل بهتر را پیدا کند، دسته فرزند فعالیت خود را قطع می‌کند؛ بنابراین دسته به حالت خواب می‌رود و این حالت تا زمانی که دوباره مفید باشد، ادامه می‌یابد و آن زمان وقتی است که تغییر در محیط تشخیص داده شود، که این شبیه، تغییر فصل در طبیعت است که حیوانات از خواب زمستانی بیدار می‌شوند. در [23]، یک روش مبتنی بر حافظه[3] برای مسائل پویای چندهدفه ارایه شده است. در این روش از یک مکانیزم پذیرش پایدار[4] استفاده شده است. سازوکار پذیرش پایدار، یک الگوریتم کارآمد برای مسائل چندمنظوره ایستا است. در ابتدا، محیطی بهبودیافته برای انطباق با تغییرات پویا، شناسایی می‌شود. سپس، از حافظه تقویت‌شده برای تطبیق و به یادآوردن راه‌حل‌های قبلی استفاده شده است. در [24]، الگوریتم DPSABC ارایه شده است، که این روش، یک مدل ترکیبی از الگوریتم‌های بهینه‌سازی ازدحام ذرات (PSO) و الگوریتم کلونی زنبور عسل (ABC) است. در این الگوریتم، از PSO برای بهینه‌سازی مقدار برازش جمعیت در الگوریتم ABC استفاده می‌شود. در این الگوریتم، ابتدا جمعیت به‌طور تصادفی تولید می‌شود. سپس، در هر سیکل، پس از محاسبه برازش همه زنبوران در همان جمعیت، زنبورهای کارگر مشخص می‌شوند، که با PSO افزایش داده می‌شوند و بهترین ذره سراسری جمعیت بر اساس مقادیر برازش مرتب‌شده تعیین می‌شود. در [25] با استفاده از خوشه‌بندی تجمعی روشی جهت تشخیص ارتباط‌های بین سلولی و بین بافتی در بیماری‌های مختلف ارایه شده است که در ابتدا چندین مدل خوشه‌بندی به‌منظور تشخیص ارتباط‌های اولیه بین

[1] Multi-Artificial Fish-Based Algorithm
[2] Clustering Particle Swarm Optimizer
[3] Memory-Based Method
[4] Stable Adoption Mechanism

سلول‌ها یا بافت‌ها ترکیب می‌شوند و تشابه بین سلول‌ها یا بافت‌ها در هر خوشه با استفاده از یک معیار شباهت مبتنی بر ساختار توپولوژیکی گراف محاسبه می‌شود و درنهایت از بیشینه شباهت‌های بین سلول‌ها یا بافت‌ها در هر خوشه برای کشف ارتباطات بین بیماری‌ها استفاده می‌شود. در [26] یک روش خوشه‌بندی ترکیبی ارائه شده است که از روش خوشه‌بندی پایه ضعیف به‌عنوان خوشه‌بندی پایه استفاده شده است. که این روش خوشه‌بندی ترکیبی دارای سرعت مناسب است و همچنین ضعف‌های عمده از جمله عدم قابلیت کشف خوشه‌های غیرکروی و غیریکنواخت را ندارد. در این مقاله برخی از کارهای انجام‌شده در زمینه بهینه‌سازی محیط‌های پویا شرح داده شد، اما خوانندگان کنجکاو می‌توانند برخی از الگوریتم‌های بهینه‌سازی فراابتکاری جدید را که درهمین‌اواخر ارایه شده است و در محدوده این مقاله نیستند در [29-26] پیدا کنند. همچنین برخی از کارهای انجام‌شده در بهینه‌سازی محیط‌های پویا به‌صورت خلاصه در جدول (۱) مشخص شده است. قابل توجه است که بیش‌تر الگوریتم‌های چنددسته‌ای الهام گرفته‌شده از طبیعت از نظر ظاهری شبیه هم هستند و تفاوت آنها در نوع حرکت و تصمیم‌گیری آنها است؛ بنابراین الگوریتم پیشنهادی ممکن است، شبیه الگوریتم‌های قبلی باشد، اما ماهیت آن به‌طورکامل متفاوت است. چون در این الگوریتم جهت تصمیم‌گیری از میانگین اندازه بردارهای والد تا فرزندانش استفاده شده است که این میانگین مشخص می‌کند که فرزندان چقدر اطراف والد پخش شده‌اند و در آن از حرکت تصادفی، موقعیت والد، بهترین موقعیت فرزند در دسته و بدترین موقعیت فرزند در دسته بهره گرفته شده است. همچنین در کارهای مورد بررسی بیان‌شده، الگوریتم‌ها از راه‌اندازی مجدد جهت دسته‌های کنار هم استفاده کرده‌اند که در روش پیشنهادی از سازوکار دافعه استفاده شده است؛ علاوه بر این در روش پیشنهادی از سازوکار جایگزینی جدیدی برای حافظه استفاده شده است و این روش به‌خوبی تنوع را حفظ می‌کند و کارایی را نسبت به روش‌های قبلی افزایش داده است. بخش‌های بعدی مقاله به‌شرح زیر ارایه خواهد شد: در بخش دو به بهینه‌سازی پرداخته می‌شود. در بخش سه محک قله‌های متحرک شرح داده شده است. در بخش چهار به تحلیل و بحث درباره نتایج پرداخته می‌شود و در بخش پنج با نتیجه‌گیری که حاصل بررسی روش پیشنهادی است مقاله پایان می‌یابد [27, 28, 29, 30].

(جدول-۱): برخی از کارهای انجام‌شده در بهینه‌سازی پویا
(Table-1): Some of dynamic environments optimization methods

الگوریتم	مزایا	معایب
FTMPSO	کشف سریع قله‌ها، دنبال‌کردن بهینه بعد از تغییر	جستجوی محلی ضعیف، عدم کاربرد واقعی، عدم استفاده از یادگیری و سازوکارهای خودسازگار، استفاده از انحصار
CESO	چندهدفه، حفظ تنوع، ردیابی بهینه سراسری و محلی، جلوگیری از هم‌گرایی زودرس	حساس به پارامتر، استفاده از سازوکار انحصار
کرم شب‌تاب	حفظ تنوع، چندجمعیتی، تنوع بالا	بهره‌وری پایین، استفاده از سازوکار انحصار
ازدحام ذرات	استفاده از حافظه، ردیابی سریع بهینه، افزایش کارایی، حفظ تنوع	استفاده از سازوکار انحصار، مشکل در تصادفی‌سازی تولید راه‌حل‌ها
EA-KDTree	کشف تغییرات و ردیابی قله‌های متحرک، پیچیدگی کم، خطاهای برون‌خطی کم	کارانبودن الگوریتم ژنتیک استفاده شده، ضعف در تنظیم پارامترها، جستجو محلی ضعیف
CellularPSO	حفظ تنوع، جلوگیری از هم‌گرایی زودرس	ناکارآمد در تعداد ابعاد و قله‌های زیاد، استفاده از سازوکار انحصار
Adaptive mQSO	تعداد دسته‌ها متغیر و افزایشی، حفظ تنوع، سرعت هم‌گرایی بالا	انعطاف‌پذیری کم، عدم تطبیق‌پذیری ذرات، استفاده از سازوکار انحصار
MPSO	چنددسته‌ای، کاوش سریع	محدودیت بر تعداد دسته، استفاده از سازوکار انحصار
CLABC	بهبود فرآیند بهره‌وری، بهبود فرآیند اکتشاف	استفاده از سازوکار انحصار
mNAFSA	چنددسته‌ای، هم‌گرایی سریع، افزایش تنوع	کاربرد محدود، عدم کاربرد واقعی، عدم استفاده از پارامترهای خودتطبیقی و یادگیری، حساس به پارامترها
مبتنی بر حافظه	استفاده از حافظه، استفاده از استراتژی جایگزینی حافظه	عدم استفاده از طرح‌های حافظه مناسب دیگر، عدم خودتطبیقی
CPSO	ردیابی چند بهینه،	عدم کشف برخی نواحی،

	کارآمد در شدت تغییرات مختلف، دارای سازوکار یادگیری	نیاز به سازوکار جستجو محلی، عدم کارایی اضافه‌کردن تعداد ذرات تصادفی، مشکل راه‌اندازی مجدد دارد
CDEPSO	چنددسته‌ای، حفظ تنوع در طول اجرا	مشکل تنظیم پارامتر، عدم تطبیق‌پذیربودن پارامترها با توجه به فرآیند جستجو
FMSO	کشف بهینه محلی و سراسری، هم‌گرایی مناسب	استفاده از سازوکار انحصار
Mqso	کاوش مناسب، حفظ تنوع و تطبیق‌پذیری، انحصار جهت ایجاد تنوع، استفاده از ضد هم‌گرایی	تعداد دسته‌ها بیشتر از تعدا قله‌ها (کاهش کارایی)، خودتطبیقی کم، استفاده از سازوکار انحصار
DPSABC	اکتشاف و بهره‌وری مناسب	سرعت پایین، استفاده از جمعیت تصادفی

۲- بهینه‌سازی

بهینه‌سازی شاخه‌ای از علوم ریاضی است و هدف آن به‌دست‌آوردن نقاط بهینه توابع با درنظرگرفتن تعدادی از محدودیت‌ها است. بهینه‌سازی به این مفهوم است، که در بین پارامترهای یک تابع به‌دنبال مقادیری باشیم، که تابع را کمینه (یا بیشینه) کنند. الگوریتم‌های بهینه‌سازی هر دو نوع مسائل بیشینه‌سازی و کمینه‌سازی را پوشش می‌دهند. همچنین به مجموعه مقادیر دامنه مسأله، راه‌حل‌های ممکن گفته می‌شود و بهترین مقدار از این مقادیر را، راه‌حل بهینه می‌نامند. هدف از بهینه‌سازی، یافتن بهترین جواب قابل قبول، با توجه به محدودیت‌ها و نیازهای مسأله است. در سال‌های اخیر یکی از مهم‌ترین و امیدبخش‌ترین پژوهش‌ها، «روش‌های ابتکاری و فراابتکاری برگرفته از طبیعت» بوده است؛ این روش‌ها شباهت‌هایی با نظام‌های اجتماعی و یا طبیعی دارند، که در آنها از معیارهای مختلفی از جمله مبتنی بر یک جواب و مبتنی بر جمعیت، الهام گرفته‌شده از طبیعت و بدون الهام از طبیعت، با حافظه و بدون حافظه، قطعی و احتمالی برای طبقه‌بندی الگوریتم‌های فراابتکاری استفاده شوند [31-24] , [27, 31, 32, 28, 29].

۲-۱- الگوریتم‌های تکاملی

الگوریتم‌های تکاملی به‌طور موفق در محدوده وسیعی از برنامه‌های کاربردی استفاده شده‌اند و برای حل مسائل بهینه‌سازی مناسب هستند. بیش‌تر برنامه‌های دنیای واقعی پویا هستند و الگوریتم‌های مورد استفاده برای حل آنها باید بتوانند با شرایط جدید سازگار شوند. برای این نوع بهینه‌سازی، یک الگوریتم تکاملی مؤثر باید بتواند تغییرات را شناسایی کند و به‌سرعت تغییرات را دنبال کند. به‌طورمعمول، الگوریتم‌های تکاملی جستجو را بر پایه جمعیت انجام می‌دهند و جستجو بر اساس جمعیت با حافظه بسیار مناسب است، به‌طوری که عناصر متعددی از حافظه می‌توانند درون فرآیند جستجو به‌کار بروند. در الگوریتم‌های تکاملی، مجموعه‌ای از راه‌حل‌های به‌صورت تصادفی ایجاد می‌شوند، راه‌حل‌های انتخاب‌شده با استفاده از عمل‌گرها، تکثیر پیدا می‌کنند و یک جمعیت جدید فرزند را تولید می‌کنند؛ درنهایت نسل بعدی جمعیت با ترکیب جمعیت والدین و فرزندان شکل می‌گیرد، این فرآیند تا زمانی که شرایط توقف خاصی، به‌عنوان مثال، تعداد نسل به‌دست آید، تکرار می‌شود. شبه‌کد عملیات‌های پایه الگوریتم تکاملی در شکل (۱) نشان داده شده است [33].

Basic operations of the evolutionary algorithm

$\boldsymbol{init(POP)}$ initialize the Population
$\boldsymbol{eval(POP)}$
WHILE (termination criteria not fulfilled)
  $\boldsymbol{POP = POP \cup Memory}$
  $S_{POP} = \boldsymbol{select(POP)}$
  $S'_{POP} = \boldsymbol{crossover}(S_{POP})$
  $S''_{pop} = \boldsymbol{mutation}(S'_{POP})$
  $\boldsymbol{Pop} = S''_{POP}$ update population
  $\boldsymbol{eval(POP)}$ evaluate individuals in the population

(شکل-۱): عملیات پایه الگوریتم تکاملی

(Figure-1): Basic operations of the evolutionary algorithm

۲-۲- محیط‌های پویا

در بسیاری از مسائل بهینه‌سازی دنیای واقعی، تابع هدف، یا محدودیت‌ها می‌توانند در طول زمان تغییر یابند، که به این مسائل، پویا گفته می‌شود. همچنین معیارهایی پیشنهاد می‌شود که محیط‌های پویا می‌توانند بر اساس آن طبقه‌بندی شوند. بر پایه این طبقه‌بندی برای محیط‌های پویا، مشخص می‌شود که کدام الگوریتم نسبت به نوع دیگر مناسب‌تر است [35 ,34]. ۱. فرکانس تغییرات: در چه

زمان‌هایی محیط تغییر می‌کند، یا چه زمانی الگوریتم بایستی به تغییرات محیط پاسخ دهد.۲. شدت تغییرات: با چه شدتی سیستم تغییر می‌کند، شدت تغییرات تا درجه زیادی بر اساس فاصله بهینه جدید از بهینه قدیم تعیین می‌شود. ۳. پیش‌بینی‌پذیری تغییر: آیا یک الگو یا یک روند در تغییرات وجود دارد، یا این که تغییرات به‌طورکامل تصادفی هستند. ۴. طول دوره و دقت دوره: آیا بهینه به محل قبلی‌اش بر می‌گردد، یا حداقل به آن نزدیک می‌شود [33, 36, 37].

۲-۳- تولید و حفظ تنوع در جمعیت

در محیط‌های پویا هنگامی که جمعیت به یک بهینه هم‌گرا می‌شود، اعضای جمعیت بسیار متراکم می‌شوند و گام حرکتی افراد جمعیت به‌دلیل جستجوی محلی بسیار کوچک می‌گردد و این باعث می‌شود که تنوع در جمعیت بسیار پایین آید. حال وقتی که محیط تغییر می‌کند و بهینه جابه‌جا می‌شود، با توجه به پایین‌بودن تنوع در جمعیت امکان تعقیب سریع بهینه مقدور نخواهد بود. بهترین روش برای مواجهه با مسأله هم‌گرایی جمعیت، به وضوح وارد کردن تنوع به فرآیند جستجو است. در [38]، روش‌های تنوع به دو دسته گروه‌بندی شده‌اند: ۱. تولید تنوع بعد از تغییر: بیشترین نیاز به تنوع زمانی است که محیط تغییر می‌کند، به‌طوری که جمعیت تنوع خود را از دست می‌دهد و می‌بایست جهت ردیابی بهینه به‌سرعت متنوع شود. ۲.حفظ تنوع در طول زمان اجرا: روش‌های وجود دارند که تنوع را در تمام طول اجرا حفظ می‌کنند. حال اگر تنوع در طول اجرا حفظ شود و جمعیت همیشه متنوع بماند، ممکن است از هم‌گرایی در تمام زمان‌ها جلوگیری شود و بهینه‌سازی نسبت به تغییرات منطبق‌تر خواهد بود. برخی از تکنیک‌های تولید تنوع را می‌توان در [39, 40, 41] و برخی از روش‌های حفظ تنوع را می‌توان در [42, 43, 44, 45, 46] مشاهده کنید.

۲-۴- حافظه

در بسیاری از مسائل پویا، تغییرات به‌صورت دوره‌ای یا مکرر اتفاق می‌افتند، یعنی بهینه ممکن است به مناطق نزدیک به مکان‌های قبلی خود بازگردد، به‌طوری که حالت فعلی محیط اغلب شبیه به حالات دیده‌شده قبلی است. استفاده از اطلاعات گذشته ممکن است به سیستم کمک کند تا با تغییرات بزرگ در محیط بهتر تطبیق پیدا کند و در طول زمان بهتر اجرا شود. یک راه برای حفظ و بهره‌برداری از اطلاعات گذشته، استفاده از حافظه است، در نتیجه حافظه محلی است که راه‌حل‌ها به‌صورت دوره‌ای در آن ذخیره می‌شوند و در زمان تغییر محیط، دوباره بازیابی می‌گردند. بنابراین حافظه از طریق بازیابی راه‌حل‌های قدیمی، تنوع را در محیط حفظ می‌کند. روش‌های حافظه برای بهینه‌سازی پویا به دو دسته حافظه ضمنی[1] و صریح[2] تقسیم می‌شوند. از حافظه ضمنی برای ذخیره تمام اطلاعات (اطلاعات اضافی) استفاده می‌شود. گفتنی است که حافظه ضمنی زیاد مورد استفاده قرار نگرفته است و عملکرد مناسبی ندارد [47, 48, 49]. در حافظه صریح، از فضای اضافی برای ذخیره اطلاعات مربوط به افراد یا محیط‌ها استفاده می‌شود. حافظه صریح اطلاعات را مجزا از جمعیت، به‌طورمعمول به‌صورت یک دسته از راه‌حل‌های خوب قبلی، ثبت می‌کند. روش‌های حافظه صریح بسیار محبوب بوده و به‌طور گسترده در بهینه‌سازی پویا استفاده می‌شوند [48]. حافظه‌ها شامل دو عمل به‌روزرسانی و بازیابی هستند، بهترین روش به‌روزرسانی حافظه می‌تواند به‌وسیلهٔ الگوهای زمانی یا تصمیمات از پیش تعیین‌شده صورت پذیرد. درصورتی‌که حافظه دارای خانه خالی باشد آن راه‌حل ذخیره می‌شود، اما درصورتی‌که خانه‌های حافظه پر باشند، می‌بایست مدخل جدید با یکی از مدخل‌های موجود جایگزین حافظه شود. به سازوکار انتخاب مدخلی که باید در حافظه اصلی تغییر کند، استراتژی جایگزینی نامیده می‌شود. قابل توجه است که نحوه به‌روزرسانی حافظه (سازوکار جایگزینی) تأثیر فراوانی بر عملکرد و کارایی حافظه و الگوریتم دارد. درعمل بازیابی حافظه، یک راه‌حل از حافظه، جایگزین یک راه‌حل جمعیت می‌شود.

۲-۵- معیارکارایی الگوریتم‌ها در محیط پویا

معیارهای مختلفی برای سنجش کارایی الگوریتم‌های تکاملی در محیط‌های پویا وجود دارند، که می‌توان به معیار: برازش کلی[3]، معیار متوسط خطا[4]، معیار دقت[5]، معیار تطبیق‌پذیری[6]، خطای درون‌خطی[7] و برون‌خطی[8] اشاره کرد [35]. بیش‌تر پژوهش‌گران از معیار خطای برون‌خطی برای سنجش کارایی روش خود و مقایسه با

[1] Implicit Memory
[2] Explicit Memory
[3] Fitness Overall
[4] Average Error
[5] Accuracy
[6] Adaptability
[7] Online
[8] Offline

دیگر روش‌ها استفاده کرده‌اند. کارایی درون‌خطی برابر است با میانگین همه مقدار ارزیابی‌هایی که در تمام مدت اجرا به‌دست آمده است. کارایی درون‌خطی بر اساس رابطه (۱) محاسبه می‌شود.

$$on-line\ (t) = \frac{1}{N}\sum_{t=1}^{N} e_t \qquad (۱)$$

که N تعداد کل ارزیابی‌ها و e_t ارزیابی در زمان t است. همچنین کارایی برون‌خطی برابر با میانگین بهترین مقدار ارزیابی پیداشده تاکنون در هر یک از مراحل است. کارایی برون‌خطی بر اساس رابطه (۲) محاسبه می‌شود:

$$off-line\ (t) = \frac{1}{N}\sum_{t=1}^{N} e_t^* \qquad (۲)$$

کهN تعداد کل ارزیابی‌ها و e_t^* ارزیابی افراد با بهترین برازش از آخرین تغییرات است.

۳- محک قله‌های متحرک

برای آزمایش عملکرد الگوریتم‌های تکاملی از محک استفاده می‌شود، معیارهای محک مختلفی وجود دارند که در آنها از ویژگی‌های متفاوت، توابع ریاضی و برنامه‌های کاربردی واقعی استفاده شده است. برخی از معیارهای محک را می‌توانید در [33, 35, 50] مشاهده کنید؛ اما یکی از محک‌های پویای معمول‌تر، که در ارزیابی الگوریتم‌های تکاملی برای بهینه‌سازی پویا استفاده می‌شود، محک قله‌های متحرک[1] (MPB) است. این محک شامل چندین قله در یک محیط چندبعدی است، که در زمان تغییر محیط، مکان، ارتفاع و عرض قله‌ها کمی تغییر پیدا می‌کنند. در قله‌های متحرک، چشم‌انداز ترکیبی از P قله در فضای D بعدی است. در هر نقطه شایستگی به‌عنوان بیشینه تمامی P توابع قله تعیین می‌شود. این شایستگی می‌تواند به‌صورت رابطه (۳) باشد:

$$F(\vec{x},t) = max_{i=1\dots P}\ P_{sh}(\vec{x}, h_i(t), w_i(t), \vec{\rho}_i(t))) \qquad (۳)$$

$P_{sh}(.)$، تابعی است که شایستگی یک نقطه داده‌شده را برای یک قله مشخص‌شده به‌وسیله ارتفاع (h_i) و طول (w_i) و موقعیت رأس (ρ_i) توصیف می‌کند. ارتفاع و پهنای هر قله به‌وسیلهٔ متغیرهای تصادفی گاوسی، تغییر یافته و به‌وسیلهٔ پارامترهای شدت ارتفاع (h_{sev}) و شدت پهنا (w_{sev})، مقیاس‌دهی می‌شوند. جهت موقعیت شیفت داده‌شده از یک متغیر طولی S_l و یک فاکتور همبستگی λ استفاده می‌شود، که طول حرکت کنترل می‌کند، قله چقدر دور شده است و فاکتور همبستگی مشخص می‌کند، حرکت تصادفی قله به چه صورت انجام پذیرد؛ بنابراین اگر $\lambda = 0$ باشد، حرکت قله به‌صورت به‌طورکامل تصادفی خواهد بود، در غیر این صورت اگر $\lambda = 1$ باشد، قله همیشه در یک سمت یکسان حرکت می‌کند تا زمانی که به یک مرز فضای مختصات برسد و راهش همانند یک شعاع از نور منعکس می‌شود. در زمان تغییر در محیط، تغییرات در یک قله می‌تواند به‌صورت رابطه (۴) توصیف شده باشد:

$$r \in N(0,1)^D \qquad (۴)$$

$$h_i(t) = h_i(t-1) + h_{sev}.r$$
$$w_i(t) = w_i(t-1) + w_{sev}.r$$
$$\vec{\rho}_i(t) = \vec{\rho}_i(t-1) + \vec{v}_i(t)$$

بردار انتقالی $v_i(t)$ ترکیبی از یک بردار تصادفی $\vec{r}$ با یک بردار انتقال قبلی $v_i(t-1)$ است. بردار تصادفی ساخته شده به‌وسیلهٔ رسم یکنواخت از $[۰,۱]$ برای هر بعد و سپس مقیاس‌دهی بردار برای داشتن طولs_l است که بر اساس رابطه (۵) است:

$$\vec{v}_i(t) = \frac{S_l}{|\vec{r}+\vec{v}_i(t-1)|}((1-\lambda)\vec{r} + \lambda\vec{v}_i(t-1)) \qquad (۵)$$

ارتفاع، پهنا و موقعیت هر قله به‌صورت تصادفی در محدوده‌ای محدود مقداردهی اولیه می‌شوند. همچنین تعدادی از توابع قله در دسترس هستند که در اینجا تابع قله به‌صورت رابطه (۶) تعریف می‌شود:

$$P_{sh}(\vec{x}, h_i(t), w_i(t), \vec{\rho}_i(t)) = h_i(t) - \qquad (۶)$$
$$w_i(t).\sqrt{\sum_{j=1\dots D}(x_j - \vec{\rho}_i(t)_j)^2}$$

۴- روش پیشنهادی

در الگوریتم‌های تکاملی، عمل راه‌اندازی دوباره دسته‌ها باعث ازدست‌رفتن ارزیابی‌های زیادی می‌شوند و ارایه الگوریتمی که بتواند از این حالت جلوگیری کند، حایز اهمیت است. همچنین بهتر است راه‌حل‌های مربوط به فضای جستجوی قبل به‌صورت دوره‌ای در حافظه ذخیره شوند و در صورت تغییر محیط از آنها استفاده شوند. حافظه‌های مختلفی در بهینه‌سازی پویا وجود دارند که باعث جستجوی سریع در الگوریتم‌ها می‌شوند، اما وجود حافظه‌ای که بتواند بر اساس مکان‌های مناسب، راه‌حل‌ها

[1] Moving Peak Benchmark

را ذخیره کند و بار محاسباتی کمتری داشته باشد، حایز اهمیت است. قابل توجه است که وجود سازوکار جایگزینی مناسب جهت ذخیره و بازیابی راه‌حل‌ها حافظه، تأثیر فراوانی بر کارایی حافظه و الگوریتم می‌گذارد. در نتیجه چالش‌های محیط‌های پویا و مشکلات روش‌های قبلی باعث شد که، این مقاله دارای اهمیت فراوانی در مسائل بهینه‌سازی پویا باشد، که بتواند با چالش‌های موجود روبه‌رو شود و برخی از مشکلات روش‌های موجود را برطرف کند و در پارامترهای مؤثر بهینه‌سازی بهبود حاصل نماید. برای این منظور در این مقاله الگوریتم چند جمعیتی والد و فرزند مبتنی بر حافظه و خوشه‌بندی[1] (CMPCS) پیشنهاد شده، که در آن از حافظه صریح، سازوکار جایگزینی حافظه و سازوکار دافعه استفاده شده است. این الگوریتم دارای خصوصیاتی از جمله سرعت هم‌گرایی بالا، انعطاف‌پذیری و تحمل‌پذیری خطا است، که آن را برای حل مسائل بهینه‌سازی قابل قبول می‌کند. در این الگوریتم سعی شده است با تنظیم مناسب پارامترها بهترین نتایج به‌دست آورده شود.

۴-۱- رفتار دسته والد و فرزندانش در طبیعت

والد و فرزندانش شامل یک عضو رهبر گروه و تعدادی فرزند است، که اطراف او پخش شده‌اند (مانند دسته مرغ و جوجه‌هایش) که به‌صورت گروهی زندگی می‌کنند و بر اساس رفتار فردی و گروهی خود تغذیه می‌کنند. فرزندان به‌صورت آزادانه در کنار والدشان غذا می‌خورند. والدها دارای رفتاری هستند که، فرزندهای خود را کنترل می‌کنند، والدها نیز فرزندان را به‌سمت آب و غذا هدایت می‌کنند و هنگامی که غذایی ببینند، آنها را صدا می‌زنند و اگر فرزندی به‌سمت غذا بیشتر رفت، بقیه دسته به آن سمت هدایت می‌شوند. والدها یک فضای شخصی را حفظ می‌کنند، که شامل اطراف سرشان و فاصله‌های خودشان با یکدیگر است. بینایی والدها در تخمین‌زدن مسافتی که می‌خواهند ببینند بسیار مهم است. اگر یک والد خیلی دور بشود، فرزندان شروع به سر و صدا کردن می‌کنند و او را دوباره به گروه بر می‌گردانند و برعکس. والدها نیز فرزندان را در کنار یکدیگر نگه می‌دارند. فرزندان به‌صورت تصادفی در محیط اطرافشان حرکت می‌کنند و به‌طورتقریبی به‌وسیلهٔ شیوه آزمایش و خطا غذای خود را پیدا می‌کنند. اگر فرزندان توسط یک والد تربیت شده باشند، مشکل غذا خوردن فرزندان از بین می‌رود.

[1] Clustering and Memory-based Hen-and-Chickens Swarm

۴-۲- الگوریتم دسته والد و فرزند

الگوریتم دسته والد و فرزند یکی از الگوریتم‌های هوش جمعی است که بر اساس جمعیت و جستجوی تصادفی کار می‌کند. گفتنی است که نزدیک‌ترین روش به الگوریتم‌های پیشنهادی دسته والد و فرزند، الگوریتم دسته ازدحام ذرات است. با توجه به توصیفات بالا می‌توان الگوریتم دسته والد و فرزندان را به‌صورت ریاضی توسعه داد؛ بنابراین این الگوریتم شامل رفتارها و قوانینی است که عبارتند از:

۱. والد و فرزندانش می‌توانند به‌صورت چنددسته‌های (چند جمعیتی) باشند، که هر دسته شامل یک والد و چندین فرزند است.
۲. والد و فرزندان در هر دسته، میدان دید خود را دارند. به‌طوری که، فرزندها نمی‌توانند از میدان دید والد خود دور شوند و در صورت خارج‌شدن، می‌بایست به درون میدان دید والد برگردانده شوند.
۳. رابطه والد-فرزند تنها ویژگی در روش پیشنهادی نیست. فرزندها می‌توانند غذای خود را پیدا کنند، که این رابطه در برخی از موجودات وجود ندارد.
۴. در هر دسته، والدها به‌صورت تصادفی محیط را جستجو می‌کنند و فرزندهای آنها بدون هیچ ترس، به جز از والدهای دیگر می‌توانند اطراف والد را کاوش کنند، که این کاوش در میدان دید آن دسته صورت می‌گیرد.
۵. والد و فرزندان آن دارای گام حرکتی هستند، که این گام حرکتی می‌تواند بر اساس شرایط محیطی، کوچک یا بزرگ شود.
۶. اگر والد بر روی بهینه (منبع غذایی) قرار نداشت، می‌بایست به فرزندهایش آزادی بیشتری دهد (گام حرکت بزرگ) تا محیط را جستجو کنند و منبع غذایی با تراکم بالا را پیدا کند، که این کار باعث اکتشاف[2] محیط می‌شود، همچنین والد می‌تواند گام حرکت فرزندهایش را کمتر کند، تا اطراف منبع غذایی کاوش بیشتری صورت بگیرد که این باعث می‌شود، بهره‌وری[3] افزایش یابد؛ بنابراین تغییرات طول گام در محیط، باعث انعطاف‌پذیری الگوریتم می‌شود.
۷. دسته والد و فرزندها به‌سختی بهینه را از دست می‌دهند، به‌طوری که حرکت‌های والد و فرزندهایش باعث می‌شوند تا بهینه سریع‌تر کشف شود.

[2] Exploration
[3] Exploitation

اساس کار الگوریتم پیشنهادی بر پایه توابعی است که از رفتارهای اجتماعی دسته والدها در طبیعت برگرفته شده‌اند. در دنیای طبیعی، والدها و فرزندهایش می‌توانند مناطقی را پیدا کنند که دارای غذای بیشتری است، که این امر با جستجوی فردی یا گروهی والد و فرزندان محقق می‌شود. مطابق با این ویژگی، مدل والد و فرزندان دارای رفتارهایی است، که به‌وسیله آنها فضای مسأله جستجو می‌شود. محیطی که والدها و فرزندها در آن زندگی می‌کند، به‌طوراساسی فضای راه‌حل و حوزه‌های والدهای دیگر است. درجه تراکم غذا در منطقه تابع هدف است. درنهایت، والد و فرزندها به مکانی می‌رسند که درجه تراکم و غلظت غذا در آنجا بیشترین (بهینه سراسری) باشد. وضعیت فعلی والدها به‌وسیلهٔ بردار $H = (H_1, H_2, \dots, H_n)$ و وضعیت فعلی فرزندان به‌وسیلهٔ بردار $CH = (CH_1, CH_2, \dots, CH_n)$ نشان داده می‌شوند. $VisualH$، برابر میدان دید والدها و $VisualCH$، برابر میدان دید فرزندان است و H_V، موقعیتی در میدان دید والد و Hc، موقعیتی در میدان دید فرزند است، که می‌خواهند به آنجا بروند. مدل والد و فرزندها شامل دو بخش متغیرها و توابع است که متغیرها شامل تنظیمات مربوط به آنها است. همچنین توابع، شامل رفتار حرکت در محیط جهت والدها و فرزندها هستند. اندازه جمعیت برابر POP و فضای مسأله D بعدی است. فاصله بین دو موقعیت X_i و X_j با $Dis_{ij} = |X_i - X_j|$ (فاصله اقلیدوسی) نشان داده می‌شود، که با استفاده از رابطه (۷) محاسبه می‌شود:

$$Dis_{ij} = \sqrt[2]{\sum_{K_{hc}=1}^{d} \left(X_{jK_{hc}} - X_{iK_{hc}}\right)^2} \qquad (۷)$$

که k_{hc}، جمعیت فرزندان به‌ازای هر والد است. در اینجا برای والدها نیز $PH_{::}$ جمعیت والدها، $H_{i:}$ والد i-ام، $POP - ch_{POP}$ تعداد والدها (POP اندازه تمام جمعیت و ch_{POP} اندازه فرزندان است)، H_{ij} مؤلفه j-ام والد i-ام در نظر گرفته می‌شود؛ همچنین برای هر نقطه در فضا نیز $X_{:}$ یک نقطه دلخواه در محیط است و $F(X_{:})$ میزان برازش است. برای حافظه نیز Q_m طول حافظه $(M \ll POP)$، $M_{::}$ حافظه، $M_{i:}$ حافظه i-ام، M_{ij} مؤلفه j-ام از حافظه i-ام و $POP_{i:}$ نیز i-امین عنصر جمعیت، POP_{ij} نیز j-ام مؤلفه از i-امین عنصر جمعیت در نظر می‌گیریم. $F(H_i)$ میزان برازش والد i-ام، $F(CH_i)$ میزان برازش فرزند i-ام و K تعداد خوشه‌ها بر اساس تعداد جمعیت است. اگر K_j^{Pop} خوشه j-ام جمعیت است، آنگاه رابطه (۸) برقرار است.

$$\begin{cases} \forall\, j, i:\ K_j^{POP} \cap K_i^{POP} = \emptyset \\ \cup_{i=1}^{h} K_i^{POP} = \{1,2,\dots,N\} \end{cases} \qquad (۸)$$

رابطه (۸) مشخص می‌کند که هیچ دو خوشه یکسانی در جمعیت وجود ندارد و اشتراک آنها تهی است و همچنین اجتماع خوشه‌ها (بر اساس تعداد والدها)، تعداد افراد جمعیت را مشخص می‌کند که می‌تواند از ۱ تا N باشد.

جمعیت ch_{POP} را به‌عنوان فرزندان و بقیه جمعیت به‌عنوان والدها تعریف می‌کنیم، که بر اساس روابط (۹) و (۱۰) می‌باشند.

$$ch_{POP(i,:)} = POP_{i,:} \qquad \forall\, i \in \{1,2,\dots,ch\} \qquad (۹)$$

$$H_{POP(i-ch):} = POP_{i,:} \qquad \forall\, i \in \{ch+1, N\} \qquad (۱۰)$$

این الگوریتم شامل جمعیتی از دسته والد و فرزندان است، که این جمعیت به‌صورت تصادفی در محیط ایجاد می‌شود، بخشی از جمعیت را والدها و بخشی دیگر را فرزندان تشکیل داده‌اند. جمعیت به تعدادی دسته والد و فرزند تقسیم می‌شود، هر دسته شامل یک والد و تعدادی فرزند است که می‌بایست محیط را به‌صورت جداگانه کاوش کنند و منبع غذایی (بهینه محلی یا سراسری) را کشف کنند. روند کار به این‌صورت است که والد به‌صورت تصادفی در محیط حرکت می‌کند و فرزندهایش اطراف آن پخش می‌شوند. دسته والد به‌همراه فرزندهایش محیط اطراف خود را جستجو می‌کنند، تا به‌سمت هدف هم‌گرا شوند. والد و فرزندهایش، میزان برازندگی خود را بررسی می‌کنند و اگر فرزندی در وضعیت بهتر بود، والد به‌سمت فرزند خود حرکت می‌کند. نکته حایز اهمیت در اینجا این است که در روش‌های چنددسته‌ای پیشنهادشده قبلی، دسته‌ها ممکن بود به یک منبع غذایی هم‌گرا شوند و می‌بایست یکی از دسته‌ها در منبع غذایی بماند و بقیه دسته‌ها که دارای برازش کمتری هستند، حذف شوند و دوباره در محیط راه‌اندازی دوباره شوند تا به جستجو خود ادامه دهند و منبع جدیدی را پیدا کنند، که این عمل، کار زمان‌بری است و باعث می‌شود ارزیابی‌های زیادی صورت بگیرد. برای این منظور در اینجا از رفتار دافعه دسته والد-فرزند در طبیعت الهام گرفته شده است. با توجه به این که در طبیعت برخی از

والدها، اجازه نزدیک‌شدن اعضایشان را به دسته‌های دیگر نمی‌دهند و آنها را به دامنه خود برمی‌گرداند، باعث شد که ما در این روش از سازوکار دافعه استفاده کنیم، که دسته‌ها یکدیگر را دفع کنند، که این کار باعث می‌شود، هیچ وقت چندین دسته به یک منبع غذایی هم‌گرا نشوند، تا عمل حذف و راه‌اندازی دوباره صورت بگیرد. همچنین در هر تکرار، والد و فرزندهای هر دسته بررسی می‌شوند و درصورتی‌که برازش فرزندی از والدش بهتر بود، جای آن با والد عوض می‌شود، تا همیشه والد بر روی بهترین مکان قرار بگیرد و بتواند دسته را هدایت و مدیریت کند. در این الگوریتم از حافظه صریح، جهت ذخیره و بازیابی راه‌حل‌هایی که به‌وسیلهٔ دسته والد و فرزندها کشف می‌شوند، استفاده می‌شود. به‌طوری که راه‌حل‌ها به‌صورت دوره‌ای ذخیره می‌شوند و در صورت نیاز دوباره بازیابی می‌شوند. در این الگوریتم از میانگین اندازه‌های بردارهای والد تا فرزندهایش استفاده شده است، این میانگین مشخص می‌کند که فرزندان چقدر اطراف والد پخش شده‌اند، اگر مقدار این میانگین از حد آستانه در نظر گرفته شده، کمتر یا نزدیک به صفر باشد، یعنی فرزندان اطراف والد قرار گرفته‌اند و به آن هم‌گرا شده‌اند و اگر این مقدار بیشتر از حد آستانه باشد، یعنی فرزندان اطراف والد پخش شده‌اند. در نتیجه جهت محاسبه فاصله بین والد و فرزندها، یعنی اندازه برداری آنها از رابطه (۱۱) و جهت محاسبه میانگین اندازه‌های برداری فرزندهای دسته تا والد از رابطه (۱۲) استفاده شده است.

$$\vec{r}_{q,:} = \overrightarrow{POP}_{q,:} - \overrightarrow{POP}_{j+|ch|,:} \qquad (۱۱)$$

که $\overrightarrow{POP}_{j+|ch|,:}$ بردار هر فرزند، $\overrightarrow{POP}_{q,:}$ بردار والد در دسته و $\vec{r}_{q,:}$ اندازه والد تا فرزند در دسته است:

$$v = \frac{\sum_{q=((j-1)\times gs+1)}^{min(j\times gs,|ch|)} |\vec{r}_{q,:}|}{gs} \qquad (۱۲)$$

که v میانگین اندازه‌های بردارهای والد تا فرزندهایش است؛ بنابراین بر اساس میانگین اندازه‌های برداری فرزندها در دسته و این که والد بر روی قله قرار دارد یا خیر، گام حرکتی متفاوت خواهد بود. به‌طوری که:

در صورت قرارگرفتن والد بر روی بهینه و هم‌گراشدن فرزندهایش اطراف او، می‌بایست موقعیت والد در حافظه ذخیره شودد و به فرزندهای دسته اجازه داده شوند تا به‌صورت تصادفی اطراف والد حرکت کنند (اکتشاف بیشتر)، که بتوانند بهینه دیگری را کشف کنند، بنابراین موقعیت جدید فرزندها بر اساس رابطه (۱۳) محاسبه می‌شود.

$$POP_{r,:} = POP_{|ch|+j,:} + \frac{RandV_r}{|RandV_r|} \times VisualCH \qquad (۱۳)$$

که $POP_{|ch|+j,:}$ والد دسته، $RandV$ مقدار تصادفی و $VisualCH$ میدان دید فرزند است.

در صورت قرارنگرفتن والد بر روی بهینه و هم‌گراشدن فرزندهایش اطراف او، فرزندها می‌بایست طوری حرکت کنند که بتوانند بهینه را کشف کنند؛ به‌طوری که فرزندها علاوه‌بر مقداری حرکت تصادفی، می‌بایست از بدترین فرزند در دسته دور و سپس به والد و بهترین فرزند در دسته نزدیک شوند، در نتیجه موقعیت جدید فرزندها بر اساس رابطه (۱۴) محاسبه می‌شود.

$$POP_{p,:} = \frac{(ch_{p,:} - W_{:})r_1 + (POP_{j+|ch|,:} - ch_{p,:})r_2 + (H_{j,:} - ch_{p,:})r_3 + RandV}{|(ch_{p,:} - W_{:})r_1 + (POP_{j+|ch|,:} - ch_{p,:})r_2 + (H_{j,:} - ch_{p,:})r_3 + RandV|} \times VisualCH + ch_{p,:} \qquad (۱۴)$$

که $ch_{p,:}$ فرزند مربوطه، W بدترین فرزند، $POP_{j+|ch|,:}$ والد دسته، $H_{j,:}$ بهترین فرزند در دسته، r و $RandV$ مقدار تصادفی و $VisualCH$ میدان دید فرزند است.

در صورت قرارگرفتن والد بر روی بهینه و هم‌گرانشدن فرزندهایش اطراف او، می‌بایست فرزندها حرکت تصادفی و اکتشاف انجام ندهند، و طوری حرکت کنند که اطراف والد همگرا شوند، تا بتوانند عملیات بهره‌وری بیشتری انجام دهند؛ بنابراین فرزندها به والد و به بهترین فرزند در دسته نزدیک می‌شوند، در نتیجه موقعیت جدید فرزندها بر اساس رابطه (۱۵) محاسبه می‌شود:

$$POP_{p,:} = \frac{(POP_{j+|ch|,:} - ch_{p,:})r_2 + (H_{j,:} - ch_{p,:})r_3}{|(POP_{j+|ch|,:} - ch_{p,:})r_2 + (H_{j,:} - ch_{p,:})r_3|} \times VisualCH + ch_{p,:} \qquad (۱۵)$$

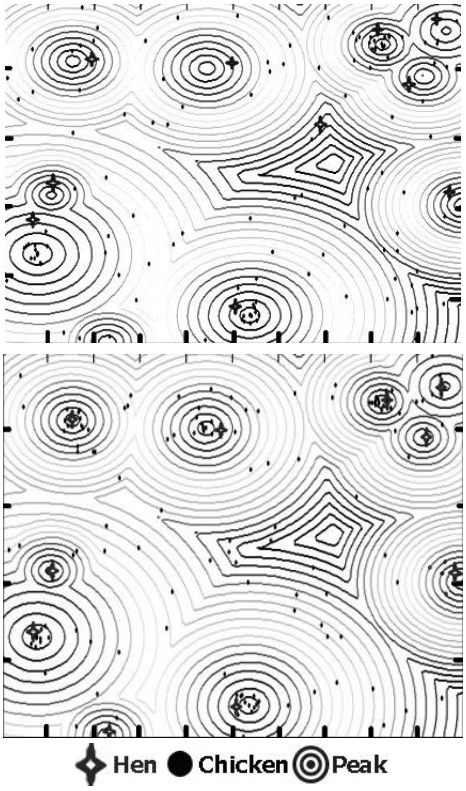
در صورت قرارنگرفتن والد بر روی بهینه و هم‌گراشدن فرزندهایش اطراف او، می‌بایست فرزندها حرکت تصادفی و اکتشاف انجام ندهند و طوری حرکت کنند که اطراف والد همگرا شوند و والد به سمت بهینه حرکت کند؛ بنابراین فرزندها به والد و به بهترین فرزند در دسته نزدیک و از بدترین فرزند دسته دور می‌شوند، درنتیجه موقعیت جدید فرزندها بر اساس رابطه (۱۶) محاسبه می‌گردد.

$$POP_{p,:} = ch_{p,:}: \frac{(ch_{p,:} - W_{:})r_1 + (POP_{j+|ch|,:} - ch_{p,:})r_2 + (H_{j,:} - ch_{p,:})r_3}{|(ch_{p,:} - W_{:})r_1 + (POP_{j+|ch|,:} - ch_{p,:})r_2 + (H_{j,:} - ch_{p,:})r_3|} \times VisualCH + ch_{p,:} \qquad (۱۶)$$

نکته‌ای که می‌بایست به آن توجه کرد این است که، در زمانی که والدها بر روی بهینه قرار دارند، نمی‌توانند تشخیص بدهند که اینجا بهینه قرار دارد و احساس می‌کنند بهبودی اتفاق نیفتاده است و فرزندها دوباره اطراف را جستجو می‌کنند و نمی‌توانند به‌جواب بهتر از آنجا برسند و این باعث می‌شود والد حرکت تصادفی انجام بدهد و از بهینه دور شود. برای رفع این عیب به این‌صورت عمل می‌شود که، اگر میانگین اندازه‌های برداری والد و فرزندهایش از آستانه مورد نظر کمتر شد، یعنی فرزندها اطراف والد جمع شده‌اند و والد بر روی بهینه قرار گرفته است؛ بنابراین می‌بایست متغیری را فعال کند که مشخص کند بر روی بهینه قرار دارد و جابه‌جا نشود و فرزندها را مجبور کند تا جستجوی سراسری خود را زیاد کنند تا در صورت جابه‌جاشدن بهینه، بتوانند سریع نقطه بهتری را پیدا کنند. حال در زمانی که بهینه تغییر می‌کند والد متغیر بولی خود را تغییر می‌دهد تا بتواند دوباره به حالت جستجویش برگردد و اعلام کند که روی بهینه قرار ندارد.

معروف‌ترین توابع محک برای محیط‌های پویا محک قله‌های متحرک است؛ بنابراین الگوریتم باید بتواند تا در حد امکان تمامی این قله‌ها را پوشش دهد تا هنگامی که یکی از آنها تبدیل به بهینه شد، یک دسته والد به همراه فرزندهایش در نزدیکی آن وجود داشته باشند. نحوه هم‌گراشدن دسته والد و فرزندان در شکل (۲) نشان داده شده است، به‌طوری که ابتدا دسته‌های والدها از قله‌ها دور بوده و بعد از مدتی به‌سمت قله‌ها هم‌گرا شده‌اند و همچنین با تغییر محیط، قله‌ها تغییر می‌کنند و دسته والدها سریع محل بهینه را پیدا می‌کنند و به‌سمت آن هم‌گرا می‌شوند.

همچنین یکی از اساسی‌ترین چالش‌ها در محیط‌های پویا، شناسایی لحظه تغییر در محیط می‌باشد. لحظه تغییر در محیط باید برای الگوریتم شناسایی شود، تا الگوریتم بتواند بعد از هر تغییر، بهینه مورد نظر را به‌سرعت ردیابی کند. برای آزمایش‌کردن این که تغییر محیط رخداده یا خیر، می‌توان شایستگی یکی از ذرات را دوباره ارزیابی کرد. درصورتی‌که مقدار شایستگی به‌دست‌آمده برابر مقدار شایستگی ثبت‌شده برای ذره مورد نظر باشد، یعنی محیط تغییر نکرده است، در غیر این صورت محیط تغییرکرده است.



(شکل-۲): نحوه همگرا شدن دسته والد و فرزندان
(Figure-2): Convergence of the CMPCS algorithm

شبه‌کد الگوریتم والد و فرزندها (CMPCS) در شکل (۳) نشان داده شده است.

Basic operations of the CMPCS algorithm

$POP_{:,:} = rand(|POP|, D)$
$gs = \frac{|ch|}{|POP|-|ch|}$ \\Group size, gs, should be a positive integer
$For\ j = 1\ to\ (|POP| - |ch|)$
  $Best\_ind = \max_{r=\{(j-1)\times gs+1,\dots,\min(j\times gs,|ch|)\}\cup\{|ch|+j\}} F(POP_{r,:})$
  $swap(POP_{j+|ch|,:}, POP_{Best\_ind,:})$
$\forall i \in \{1, \dots, |ch|\}: ch_{i,:} = POP_{i,:}$
$\forall i \in \{|ch|+1, \dots, |POP|\}: H_{i-|ch|,:} = POP_{i,:}$
$\forall i \in \{1, \dots, |M|\}: M_i = CreateRecord(POP_{i+|ch|,:}) \&\& EXP_i = 0$
$\forall i \in \{1,2, \dots, |POP| - |ch|\}: \beta_i = 0 \&\& Counter_i = 0 \&\& Trial_i = 0$
$For\ i = 1\ to\ I$
  $For\ j = 1\ to\ (|POP| - |ch|)$
    $if(ChangeFlag)$
      $Worst\_ind = \min_{r=\{|ch|+1,\dots,|POP|\}} F(POP_{r,:})$
      $W_{:} = POP_{Worst\_ind,:}$
      I_{FM} =Index of the most similar memory record to $W_{:}$
      $POP_{Worst\_ind,:} = Retrieve(M_{I_{FM}})$
      $if\left(F(W_{:}) \geq F(POP_{Worst\_ind,:})\right)$
        $POP_{Worst\_ind,:} = W_{:}$
        $EXP_{I_{FM}} = EXP_{I_{FM}} + 1$
        $if(EXP_{I_{FM}} > \theta_E)$
          $Best\_ind = \max_{r=\{|ch|+1,\dots,|POP|\}} F(POP_{r,:})$
          $EXP_{I_{FM}} = 0$
          $M_{I_{FM}} = Save(M_{I_{FM}}, POP_{Best\_ind,:})$

$elseif\left(F(H_{j,:}) < F(POP_{Worst\_ind,:})\right)$
$EXP_{I_{FM}} = 0$
$H_{j,:} = POP_{j+|ch|,:}$
$ChangeFlag = 0$
$Trial_j = 0; \beta_j = 0; Counter_j = 0;$
$Continue;$
$if(Counter_j > \theta_\beta)$ \\ θ_β is a threshold
$Counter_j = 0$
$\beta_j = 1$
$Continue;$
$if(Trial_j > \theta_C)$ \\ θ_C is a threshold
$Trial_j = 0; \beta_j = 0; Counter_j = 0;$
$I_{FM} = POP_{j+|ch|,:}$
$M_{I_{FM}} = Save(M_{I_{FM}}, POP_{j+|ch|,:})$
$Restart\ POP_{j+|ch|,:}$ and $\forall q \in \{(j-1)\times gs + 1, \dots, \min(j \times gs, |ch|)\}: \overrightarrow{POP}_{q,:};$
$Continue;$
$\forall q \in \{(j-1)\times gs + 1, \dots, \min(j \times gs, |ch|)\}: \vec{r}_{q,:} = \overrightarrow{POP}_{q,:} - \overrightarrow{POP}_{j+|ch|,:}$
$v = \frac{\sum_{q=((j-1)\times gs+1)}^{\min(j\times gs,|ch|)} |\vec{r}_{q,:}|}{gs}$
$if(v < \theta_v \&\& \beta_j)$ \\ θ_v is a threshold
\\ Save the best position and wander
$I_{FM} = POP_{j,:}$
$M_{I_{FM}} = Save(M_{I_{FM}}, POP_{j,:})$
$\forall r \in \{(j-1)\times gs + 1, \dots, \min(j \times gs, |ch|)\}:$
$POP_{r,:} = POP_{|ch|+j,:} + \frac{RandV_r}{|RandV_r|} \times VisualCH \&\& RandV_r = rand(1, D)$
$elseif(v < \theta_v \&\& !\beta_j)$
\\child becomes near to the hen & near to the best & far from the worst and also have some freedom
$For\ p = (j-1)\times gs + 1\ to\ \min(j \times gs, |ch|)$
$RandV = rand(1, D)$
$POP_{p,:} = \frac{(ch_{p,:} - W_{:})r_1 + (POP_{j+|ch|,:} - ch_{p,:})r_2 + (H_{j,:} - ch_{p,:})r_3 + RandV}{|(ch_{p,:} - W_{:})r_1 + (POP_{j+|ch|,:} - ch_{p,:})r_2 + (H_{j,:} - ch_{p,:})r_3 + RandV|} \times VisualCH + ch_{p,:}$
$elseif(\beta_j)$
\\ child becomes near to the hen & near to the best
$For\ p = (j-1)\times gs + 1\ to\ \min(j \times gs, |ch|)$
$POP_{p,:} = \frac{(POP_{j+|ch|,:} - ch_{p,:})r_2 + (H_{j,:} - ch_{p,:})r_3}{|(POP_{j+|ch|,:} - ch_{p,:})r_2 + (H_{j,:} - ch_{p,:})r_3|} \times VisualCH + ch_{p,:}$
$else$
$For\ p = (j-1)\times gs + 1\ to\ \min(j \times gs, |ch|)$
$POP_{p,:} = ch_{p,:}:$
$= \frac{(ch_{p,:} - W_{:})r_1 + (POP_{j+|ch|,:} - ch_{p,:})r_2 + (H_{j,:} - ch_{p,:})r_3}{|(ch_{p,:} - W_{:})r_1 + (POP_{j+|ch|,:} - ch_{p,:})r_2 + (H_{j,:} - ch_{p,:})r_3|} \times VisualCH + ch_{p,:}$

$check\ if\ a\ chiken\ is\ outside\ of\ VisualH;\ if\ so, return\ it$
$Best\_ind = \max_{r=\{(j-1)\times gs+1,\dots,\min(j\times gs,|ch|)\}} F(POP_{r,:})$
$x_{:} = POP_{Best\_ind,:}$
$if\left(F(x_{:}) > F(POP_{|ch|+j,:})\right)$
$POP_{|ch|+j,:} = x_{:}$
$Trial_j = 0; \beta_j = 0; Counter_j = 0;$
$if\left(F(H_{j,:}) < F(POP_{|ch|+j,:})\right)$
$H_{j,:} = POP_{j+|ch|,:}$
$else$
$if(\beta_j)$
$Trial_j = Trial_j + 1;$
$else$
$Counter_j = Counter_j + 1;$
$For\ p = (j-1)\times gs + 1\ to\ \min(j \times gs, |ch|)$
$ch_{p,:} = POP_{p,:}$

(شکل-۳): شبه‌کد الگوریتم پیشنهادی CMPCS

(Figure-3): The pseudo-code of the CMPCS algorithm

۴-۳- ذخیره‌سازی راه‌حل‌های حافظه

جهت ذخیره یک راه‌حل در حافظه، در صورتی که حافظه دارای خانه (مدخل) خالی باشد، آن راه‌حل ذخیره می‌شود، اما درصورتی‌که خانه‌های حافظه پر باشند، می‌بایست مدخل جدید با یکی از مدخل‌های موجود، جایگزین حافظه شود. برای این منظور در ابتدا می‌بایست، جمعیت خوشه‌بندی، سپس موقعیت بهترین عضو و خوشه مربوط به آن را مشخص شود.

اگر عناصر حافظه، عضو خوشه مورد نظر بودند، دورترین عنصر حافظه از مرکز خوشه با بهترین عضو جمعیت جابه‌جا می‌شود و درصورتی‌که هیچ عنصری از حافظه، عضو خوشه مورد نظر نبودند، نزدیک‌ترین عضو حافظه به مرکز خوشه با بهترین عضو جمعیت جابه‌جا می‌شود.

۴-۴- بازیابی راه‌حل‌های حافظه

برای بازیابی بهترین مدخل از حافظه، می‌بایست موقعیت بهترین مدخل ذخیره‌شده در حافظه و خوشه مربوط به آن مشخص شود. سپس می‌بایست هر عضو جمعیت به یک خوشه حافظه اختصاص داده و بررسی شود، که کدام عضو در جمعیت عضو خوشه مربوط به بهترین عضو حافظه است. در میان داده‌های جمعیت عضو خوشه، داده‌ای که دورتر از مرکز خوشه است، جهت حذف از جمعیت انتخاب می‌شود. این بدان معنی است که بدترین عنصر انتخاب می‌شود (بدترین عنصر، دورترین عنصر از مرکز خوشه است)؛ بنابراین، بهترین عضو حافظه با عضوی از خوشه مربوطه که بیشترین فاصله را با مرکز خوشه دارد، جایگزین و این باعث می‌شود که بهترین عضو در حافظه در موقعیت بدترین عضو ذخیره‌شده در جمعیت و مجاور خودش قرار بگیرد. همچنین اگر عنصری از جمعیت در خوشه مربوطه وجود نداشت، عنصری که به مرکز خوشه مربوطه نزدیک‌تر است، انتخاب می‌شود.

۵- نتایج و بحث

به‌منظور ارزیابی الگوریتم‌های پیشنهادی و مقایسه آنها با الگوریتم‌های دیگر در محیط‌های پویا، از محک قله‌های متحرک استفاده شده است [17]. محک قله‌های متحرک به‌عنوان یک محیط پویای سراسری برای آزمایش کارایی الگوریتم‌های مختلف، در نظر گرفته شده است. تمام آزمایش‌ها بر روی الگوریتم‌ها در شرایط مشابه صورت

گرفته است و الگوریتم‌های مورد مقایسه همگی بر اساس پیاده‌سازی‌های موجود بر روی محک ارزیابی شده‌اند. در جدول (۲) تنظیمات استاندارد پارامترهای تابع محک قله‌های متحرک نشان داده شده است.

(جدول-۲): تنظیمات استاندارد تابع محک قله‌های متحرک
(Table-2): Standard Setting for moving peaks benchmark

پارامتر	مقدار
number of peaks(P)	10
Change Frequency (f)	5000
Height severity(h_{sev})	7.0
Width severity (w_{sev})	1.0
Peak shape (P_{sh})	Con
Basic function	No
Shift length S_l	1.0
Number of dimensions (D)	5
Correlation coefficient (λ)	0
D_s	[100 ,0]
h_s	[30.0,70.0]
W_s	[1 ,12]
I	50.0

برای سنجش کارآیی الگوریتم‌های تکاملی به‌صورت کمی در محیط‌های پویا از معیاری به‌نام خطای درون‌خطی و برون‌خطی استفاده می‌شود. در کارایی درون‌خطی از میانگین همه مقدار ارزیابی‌هایی که در تمام مدت اجرا به‌دست آمده است، استفاده می‌شود و در کارایی برون‌خطی از میانگین بهترین مقدار ارزیابی پیداشده تاکنون در هر یک از مراحل استفاده می‌شود [33, 35, 36, 51]. تنظیم پیش‌فرض برای الگوریتم پیشنهادشده CMPCS در جدول (۳) نشان داده شده است.

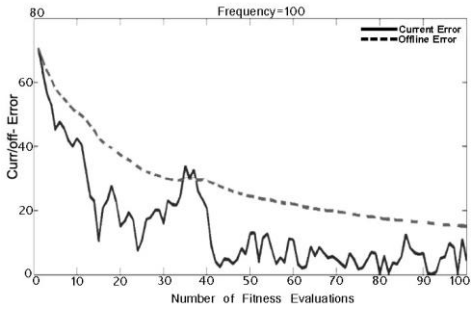
(جدول-۳): تنظیم استاندارد برای الگوریتم پیشنهادی
(Table-3): Standard Setting for the proposed algorithm

پارامتر	مقدار
$VisualH$	5
$VisualCH$	1
$\|M\|$	10
$\|POP\|$	100
G	5

الگوریتم پیشنهادی با الگوریتم‌های mQSO، FMSO، [11] و CellularPSO [9] و Multi-Swarm [12]، AmQSO [10]، FTMPSO [3]، CDEPSO [21] و DPSABC [24] مورد مقایسه قرار گرفته است. در الگوریتم پیشنهادی از حافظه صریح و الگوریتم‌های مورد مقایسه از حافظه ضمنی استفاده شده است. در این بخش به شرح آزمایش‌های انجام‌شده بر روی مدل پیشنهادی در فرکانس‌های پانصد تا ده‌هزار و با تعداد قله‌های یک الی دویست پرداخته می‌شود.

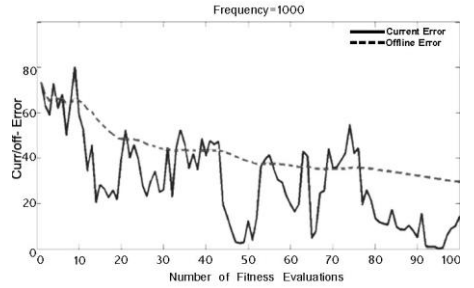
در روش پیشنهادی به‌دلیل وجود تنوع زیاد، به‌طورتقریبی بیش‌تر قله‌ها مورد پوشش ذرات قرار می‌گیرند. شکل‌های (۴) تا (۷) نمودار خطای برون‌خطی و خطای جاری را برای روش پیشنهادی در فرکانس‌های تغییر ۱۰۰، ۱۰۰۰، ۲۰۰۰ و ۳۰۰۰، شدت تغییرات یک و تعداد قله ده را نشان می‌دهند، که می‌توان از این شکل‌ها نتیجه گرفت، خطای روش پیشنهادشده به‌طور قابل قبولی کاهش می‌یابند.

نتایج آزمایش‌ها برای همه الگوریتم‌ها، میانگین خطای برون‌خطی با فاصله اطمینان ۹۵ درصد در ۱۰۰ مرتبه اجرا است. خطای برون‌خطی و خطای استاندارد به‌دست‌آمده از آزمایش‌ها برای محیط‌های با پویایی‌های متفاوت در جدول‌های (۴) تا (۷) نشان داده شده است. نتایج بهتر به‌صورت پر رنگ است. همان‌طور که مشاهده می‌شود، اختلاف خطای برون‌خطی الگوریتم پیشنهادی نسبت به الگوریتم‌های دیگر با افزایش فرکانس محیط و همچنین با افزایش پیچیدگی فضا (افزایش تعداد قله‌ها) افزایش پیدا می‌کند. علت امر این است که الگوریتم پیشنهادی سریع‌تر می‌تواند راه‌حل‌های بهتر را پس از مشاهده تغییر در محیط به‌دست بیاورد.



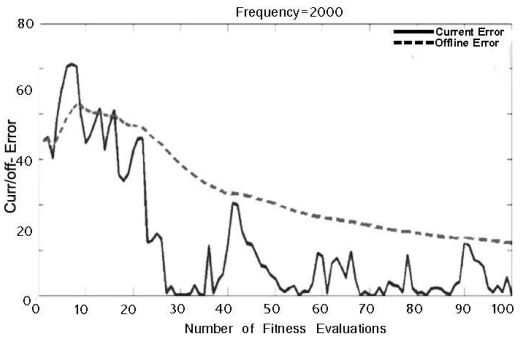
(شکل-۴): نمودار خطای برون‌خطی و جاری الگوریتم پیشنهادی با ۱۰ قله و فرکانس = ۱۰۰

(Figure-4): The diagram of the offline error and the current error of the proposed algorithm in a dynamic environment with 10 peaks and frequency=100



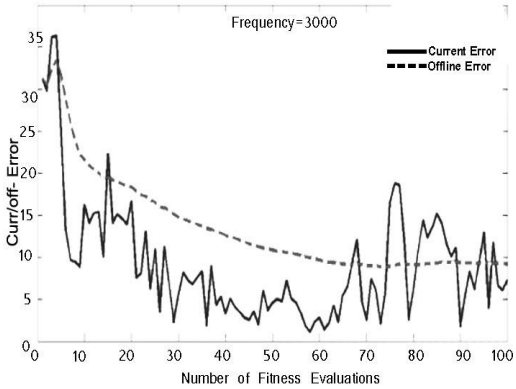
(شکل-۵): نمودار خطای برون‌خطی و جاری الگوریتم پیشنهادی با ۱۰ قله و فرکانس = ۱۰۰۰

(Figure-5): The diagram of the offline error and the current error of the proposed algorithm in a dynamic environment with 10 peaks and frequency= 1000



(شکل-۶): نمودار خطای برون‌خطی و جاری الگوریتم پیشنهادی با ۱۰ قله و فرکانس = ۲۰۰۰

(Figure-6): The diagram of the offline error and the current error of the proposed algorithm in a dynamic environment with 10 peaks and frequency= 2000



(شکل-۷): نمودار خطای برون‌خطی و جاری الگوریتم پیشنهادی با ۱۰ قله و فرکانس= ۳۰۰۰

(Figure-7): The diagram of the offline error and the current error of the proposed algorithm in a dynamic environment with 10 peaks and frequency=3000

(جدول-۴): مقایسه خطای برون‌خطی و استاندارد روش پیشنهادی برای فرکانس ۵۰۰

(Table-4): A comparison between the offline error and the standard error of the proposed method with other methods for f = 500

تعداد قله‌ها	CMPCS	AmQSO	DPSABC	FTMPSO	Multi Swarm PSO	Cellular PSO	FMSO	mQSO10 (5+5q)
1	3.25±0.25	3.02±0.32	2.77±0.00	**1.76±0.09**	5.46±0.30	13.4±0.74	5.58±0.19	33.67±0.34
5	2.54±0.27	5.77±0.56	-	**2.93±0.18**	5.48±0.19	9.63±0.49	9.45±0. 4	11.91±0.7
10	**3.34±0.42**	5.37±0.42	3.42±0.00	3.91±0.19	5.95±0.19	9.42±0.21	8.26±0.3	9.62±0.32
20	**3.11±0.15**	6.82±0.34	3.12±0.00	4.83±0.19	6.45±0.16	8.84±0.28	17.34±0.3	9.07±0.25
30	**3.35±0.18**	7.10±0.39	3.69±0.00	5.05±0.21	6.60±0.14	8.81±0.24	16.39±0.4	8.80±0.21
40	**3.08±0.19**	7.57±0.32	-	-	6.85±0.13	8394±0.24	15.34±0.4	8.55±0.21
50	**3.09±0.24**	7.55±0.32	3.22±0.00	4.98±0.15	7.04±0.10	8.62±0.23	5.54±0.2	8.72±0.20
100	3.10±018	7.34±0.31	**3.01±0.00**	5.31±0.11	7.39±0.13	8.54±0.21	2.87±0.6	8.54±0.16
200	**3.05±0.16**	7.48±0.19	3.16±0.00	5.52±0.21	7.52±0.12	8.28±0.18	11.52±0.6	8.19±0.17

(جدول-۵): مقایسه خطای برون‌خطی و استاندارد روش پیشنهادی برای فرکانس ۱۰۰۰

(Table-5): A comparison between the offline error and the standard error of the proposed method with other methods for f = 1000

تعداد قله‌ها	CMPCS	AmQSO	DPSABC	FTMPSO	Multi Swarm PSO	Cellular PSO	FMSO	mQSO10 (5+5q)
1	2.50±0.24	2.33±0.31	1.68±0.00	**0.89±0.05**	2.90±0.18	6.77±0.38	4.42±0.4	18.6±0.16
5	2.37±0.26	2.90±0.32	-	**1.70±0.10**	3.35±0.18	5.30±0.32	10.59±0.2	6.56±0.38
10	**2.12±0.15**	4.56±0.40	3.23±.0.00	2.36±0.09	3.94±0.08	5.15±0.13	10.40±0.1	5.71±0.22
20	**2.10±0.14**	5.36±0.47	3.40±0.00	3.01±0.12	4.33±0.12	5.23±0.18	10.33±0.1	5.85±0.15
30	**2.05±0.21**	5.20±0.38	3.28±0.00	3.06±0.10	4.41±0.11	5.33±0.16	10.06±0.1	5.81±0.15
40	**1.77±0.26**	-	-	-	4.52±0.09	5.61±0.16	9.85±0.11	5.70±0.14
50	**1.46±0.17**	6.06±0.14	2.67±0.00	3.29±0.10	4.57±0.08	5.55±0.14	9.54±0.11	5.87±0.13
100	**1.17±0.19**	4.77±0.45	3.08±0.00	3.63±0.09	4.77±0.08	5.57±0.12	8.77±0.09	5.83±0.13
200	**1.06±0.14**	5.75±0.26	3.01±0.00	3.74±0.09	4.76±0.07	5.50±0.12	8.06±0.07	5.54±0.11

(جدول-۶): مقایسه خطای برون‌خطی و استاندارد روش پیشنهادی برای فرکانس ۵۰۰۰

(Table-6): A comparison between the offline error and the standard error of the proposed method with other methods for f = 5000

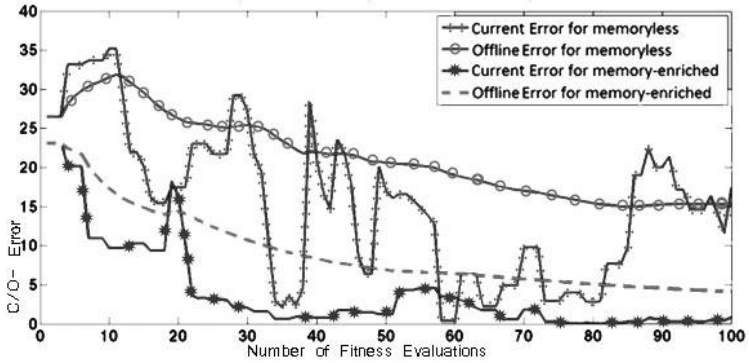
تعداد قله‌ها	CMPCS	AmQSO	DPSABC	FTMPSO	Multi Swarm PSO	Cellular PSO	FMSO	mQSO10 (5+5q)
1	1.08±0.14	2.62±0.10	2.25±0.00	**0.18±0.01**	0.56±0.04	2.55±0.12	3.44±0.11	3.82±0.35
5	0.76±0.19	1.01±0.09	-	**0.47±0.05**	1.06±0.06	1.68±0.11	2.94±0.07	1.90±.0.8
10	**0.65±.10**	1.51±0.1	2.13±0.00	0.67±0.04	1.51±0.04	1.78±0.05	3.11±0.06	1.91±.0.08
20	**0.61±0.23**	2.00±0.15	2.07±0.00	0.93±0.04	1.89±0.04	2.61±0.07	3.36±0.06	2.56±0.10
30	**0.55±0.15**	2.19±0.17	1.88±0.00	1.14±0.04	2.03±0.06	2.93±0.08	3.28±0.05	2.68±0.10

40	**0.51±0.25**	-	-	-	2.04±0.06	3.14±0.08	3.26±0.04	2.65±0.08
50	**0.46±0.08**	2.43±0.13	1.91±0.00	1.32±0.04	2.08±0.02	3.26±0.08	3.22±0.05	2.63±0.08
100	**0.35±0.15**	2.68±0.12	1.89±0.00	1.61±0.03	2.14±0.02	3.41±0.07	3.06±0.04	2.52±0.06
200	**0.31±0.15**	2.62±0.10	1.87±0.00	1.67±0.03	2.11±0.03	3.40±0.06	2.84±0.03	2.36±0.05

(جدول-۷): مقایسه خطای برون‌خطی و استاندارد روش پیشنهادی برای فرکانس ۱۰۰۰۰

(Table-7): A comparison between the offline error and the standard error of the proposed method with other methods for f = 10000

تعداد قله‌ها	CMPCS	AmQSO	DPSABC	FTMPSO	Multi Swarm PSO	Cellular PSO	FMSO	mQSO10 (5+5q)
1	0.17±0.14	0.19±0.02	2.67±0.00	**0.09±0.00**	0.27±0.02	1.53±0.12	1.90±0.06	1.90±0.18
5	**0.12±0.13**	0.45±0.04	-	0.31±0.04	0.70±0.10	0.92±0.10	1.75±0.06	1.03±0.06
10	**0.09±0.12**	0.76±0.06	9.01±0.01	0.43±0.03	0.97±0.04	1.19±0.07	1.91±0.04	1.100.07
20	**0.09±0.01**	1.28±0.12	6.60±0.01	0.56±0.01	1.34±0.08	2.20±0.10	2.16±0.04	1.84±0.08
30	**0.05±0.12**	1.78±0.09	7.70±0.01	0.69±0.09	1.43±0.05	2.60±0.13	2.18±0.04	2.00±0.09
40	**0.04±0.15**	-	-	-	1.47±0.06	2.73±0.11	2.21±0.03	1.99±0.07
50	**0.07±0.24**	1.55±0.08	8.10±0.01	0.86±0.02	1.47±0.04	2.84±0.12	2.60±0.08	1.99±0.07
100	**0.08±0.27**	1.89±0.14	8.34±0.01	1.08±0.01	1.50±0.03	2.93±0.09	2.20±0.03	1.85±0.05



(شکل-۸): نمودار خطای جاری و برون‌خطی الگوریتم پیشنهادی با حافظه و بدون حافظه

(Figure-8): The diagram of the current error of the proposed algorithm enriched with memory andthe memoryless proposed algorithm

استفاده از حافظه یکی از روش‌های است که می‌تواند به‌طور قابل توجهی سرعت هم‌گرایی هر الگوریتم بهینه‌سازی مبتنی بر جمعیت را افزایش دهد. تأثیر حافظه بر عملکرد الگوریتم پیشنهاد شده در شکل (۸) نشان داده شده است.

انتخاب مقدار مناسب برای $VisualH$ نقش مهمی در هم‌گرایی سریع گروه‌ها قبل از هم‌گرایی سراسری الگوریتم پیشنهادی دارد. همچنین بر عملکرد الگوریتم پیشنهادی تأثیر می‌گذارد. همان‌طور که از جدول (۸) می‌توان نتیجه گرفت، بهترین نتیجه برای این پارامتر ۵ است.

(جدول-۸): تأثیر پارامتر $VisualH$ بر عملکرد الگوریتم CMPCS

(Table-8): Effect of different values for $VisualH$ parameter on the algorithm performance

Initial Visual Parent ($VisualH$)	Offline_err ± Std_err
1	2.75 (0.15)
2	1.80 (0.19)
5	1.09 (0.28)
10	1.29 (0.24)
20	1.71 (0.12)
30	2.76 (0.28)
50	2.67 (0.22)

۶- نتیجه‌گیری

مسائل بهینه‌سازی پویا مربوط به یک محور پژوهش مهم هستند، زیرا در برنامه‌های کاربردی متعددی در دنیای واقعی کاربرد دارند. تاکنون روش‌های مختلفی برای بهینه‌سازی ارایه شده است و یکی از معروف‌ترین روش‌های بهینه‌سازی، الگوریتم‌های هوش جمعی هستند. در محیط‌های پویا ما با چالش‌های مختلفی روبه‌رو هستیم که مهمترین آنها تغییر محیط و ازدست‌دادن تنوع است. بنابراین به‌دلیل این پویایی و تغییرات می‌بایست از

روش‌های تکاملی برگرفته از طبیعت استفاده شود. با توجه به چالش‌های مختلف در محیط‌های پویا و ضعف در کارهای انجام‌شده قبلی، ارایه الگوریتم کارا مبتنی بر چند جمعیت و حافظه که دارای قابلیت دافعه باشد، در مسائل بهینه‌سازی پویا حایز اهمیت است؛ بنابراین در این مقاله یک الگوریتم چندجمعیتی مبتنی بر حافظه بر اساس سازوکار جایگزینی خوشه‌بندی و سازوکار دافعه در دسته‌ها پیشنهاد شده است. با توجه به نتایج به‌دست‌آمده از خطای برون‌خطی، الگوریتم CMPCS نسبت به الگوریتم‌های دیگر عملکرد بهتری داشته است و سریع‌تر می‌تواند راه‌حل‌های مناسب را پس از مشاهده تغییر در محیط به‌دست بیاورد. در روش CMPCS به‌دلیل وجود تنوع زیاد، به‌طورتقریبی بیش‌تر قله‌ها مورد پوشش ذرات قرار می‌گیرند. همچنین با توجه به نتایج به‌دست‌آمده در فرکانس‌های مختلف، الگوریتم CMPCS نسبت به الگوریتم‌های دیگر کارکرد بهتری داشته است. حال با توجه به نتایج به‌دست‌آمده می‌توان نتیجه گرفت که الگوریتم CMPCS در محیط‌های بهینه‌سازی دارای عملکرد مناسب است و به بهبود آنها کمک فراوانی کرده است. روش‌های پیشنهادی می‌توانند در آینده برای بسیاری از مسائل پویا از جمله زمان‌بندی کار پویا، بهینه‌سازی مسیریابی در شبکه‌های حسگر تلفن همراه استفاده شوند و همچنین می‌توان از روش‌های حفظ تنوع و پیش‌بینی در کارهای آینده بهره جست.

قدردانی

این مقاله مستخرج از رساله دکترا آقای محسن مرادی با راهنمایی دکتر صمد نجاتیان و دکتر حمید پروین و مشاوره خانم دکتر سیده وحیده رضایی و دکتر کرم الله باقری فرد است.

۷- مراجع 7- References

[1] S. Saremi, S. Mirjalili and A. Lewis, "Biogeography-based optimisation with chaos," *Neural Computing and Applications*, pp. 1077-1097, 2014.

[2] S. Mirjalili, S. Mirjalili and A. Lewis, "Grey Wolf Optimizer," *Advances in Engineering Software*, pp. 69: 46-61, 2014.

[3] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam and M. Meybodi, "a novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization," Applied Soft Computing, 2013.

[4] R. Lung and D. Dumitrescu, "A Collaborative Model for Tracking Optima in Dynamic Environments," in In: *IEEE Congress on Evolutionary Computation*, 2007.

[5] F. Ozsoydan and A. Baykasoglu, "A multi-population firefly algorithm for dynamic optimization problems," in Evolving and Adaptive Intelligent Systems (EAIS), 2015 IEEE International Conference, 2015.

[6] X. Hu and R. Eberhart, "Adaptive particle swarm optimisation: detection and response to dynamic systems," *In Congress on Evolutionary Computation*, pp. 1666–1670, 2002.

[7] S. Sadeghi, H. Parvin and F. Rad, "Particle Swarm Optimization for Dynamic Environments," in Springer International Publishing, 14th Mexican International Conference on Artificial intelligence, MICAI, 2015.

[8] S. Yang and C. Li, "A clustering particle swarm optimizer for dynamic optimization," *in Proc. Congr. Evol. Com*, pp. 439–446, 2009.

[9] A. Hashemi and M. Meybodi, "Cellular PSO: A PSO for Dynamic Environments," Advances in Computation and Intelligence, pp. 422–433, 2009.

[10] T. Blackwell, J. Branke and X. Li, "Particle swarms for dynamic optimization problems," *Swarm Intelligence. Springer Berlin Heidelberg*, pp. 193-217, 2008.

[11] T. Blackwell and J. Branke, "MultiswarmT Exclusion, and Anti-Convergence in Dynamic Environment," 2006.

[12] M. Kamosi, A. Hashemi and M. Meybodi, "A New Particle Swarm Optimization Algorithm for Dynamic Environments," *SEMCCO*, pp. 129-138, 2010.

[13] W. Su, H. Chen, F. Liu, S. Jing and W. Li, "A novel comprehensive learning artificial bee colony optimizer for dynamic optimization biological problem," *Saudi Journal of Biological Sciences*, pp. 695-702, 2017.

[14] S. Nseef, S. Abdullah, A. Turky and G. Kendall, "An adaptive multi-population artificial bee colony algorithm for dynamic optimization problems," *Knowledge-based Systems Center for Artificial Intelligence and Technology (CAIT)*, pp. 14-23, 2015.

[15] D. Yazdani, B. Nasiri and AND..., ""mNAFSA: (2014) A novel approach for optimization in dynamic Environments with global Changes," Swarm and Evolutionary Computation, 2014.

[16] Y. Bravo, G. Luque and E. Alba, "Global memory schemes for dynamic optimization," Springer Science,Business Media Dordrecht, 2015.

[17] S. Biswas, S. Kundu, S. Das and A. Vasilakos, "Information sharing in bee colony for etecting multiple niches in non-stationary environments," 2013.

[18] S. Yang, "Associative memory scheme for genetic algorithms in dynamic environments," *In Applications of Evolutionary Computing: EvoWorkshops*, pp. 788–799, 2006.

[19] S. Yang and C. Li, "A Clustering Particle Swarm Optimizer for Locating and Tracking Multiple Optima in Dynamic Environments," in IEEE Transactions on Evolutionary Computation, 2010.

[20] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam and M. Meybodi, "novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization," Applied Soft Computing, 2013.

[21] J. Kordestani, A. Rezvanian and M. Meybodi, ""CDEPSO: a bi-population hybrid approach for dynamic optimization problems," *Applied intelligence*, pp. vol. 40, pp. 682-694, 2014.

[22] M. Kamosi, A. Hashemi and M. Meybodi, "A Hibernating Multi-Swarm Optimization Algorithm for Dynamic Environments," *in Proceedings of World Congress on Nature and Biologically Inspired Computing, NaBIC, Kitakyush*, pp. 370-376, 2010.

[23] X. Chen, D. Zhang and X. Zeng, "A Stable Matching-Based Selection and Memory Enhanced MOEDA/D for Evolutionary Dynamic Multiobjective Optimization," in Tools with Artificial intelligence (ICTAI),IEEE 27th International Conference, 2015.

[24] N. Baktash and M. Meybodi, "A New Hybrid Model of PSO and ABC Algorithms for Optimization in Dynamic Environment," *Int'l Journal of Computing Theory Engineering*, pp. vol. 4, pp. 362-364, 2012.

[25] M. mojarad, H. parvin, S. nejatiyan and K. A. Bagheri , "Combining a Ensemble Clustering Method and a New Similarity Criterion for Modeling the Hereditary Behavior of Diseases," *JSDP*, vol. 18, no. 2, pp. 97-114, 2021.

[26] F. najafi, H. parvin, K. mirzaei and S. nejatiyan, "A new ensemble clustering method based on fuzzy cmeans clustering while maintaining diversity in ensemble," *JSDP*, vol. 17, no. 4, pp. 103-122, 2021.

[27] A. Prajapati and J. Chhabra, "Harmony search based remodularization for object-oriented software systems," Computer Languages, Systems & Structures, 2017.

[28] X. Peng, K. Liu and Y. Jin, "A dynamic optimization approach to the design of cooperative co-evolutionary algorithms. Knowl," Based Syst, 2016.

[29] D. Wang, F. Liu and Y. Jin, "A multi-objective evolutionary algorithm guided by directed search for dynamic scheduling," Computers & OR, 2017.

[30] W. Luo, J. Sun, C. Bu and H. Liang, "Species-based Particle Swarm Optimizer enhanced by memory for dynamic optimization," Appl. Soft Comput, 2016.

[31] B. Yildiz, "A comparative investigation of eight recent population-based optimisation algorithms for mechanical and structural design problems," *International Journal of Vehicle Design*, pp. 73,1-3,208-218, 2017.

[32] B. Yildiz and H. Lekesiz, "Fatigue-based structural optimisation of vehicle components," *International Journal of Vehicle Design*, pp. 73, 1-3, 54-62, 2017.

[33] D. Simon, EVOLUTIONARY OPTIMIZATION ALGORITHMS, 2013.

[34] J. Branke, "Evolutionary Optimization in Dynamic Environments," Kluwer, 2002.

[35] A. Simoes, IMPROVING MEMORY BASED EVOLUTIONARY ALGORITHM FOR DYNAMIC ENVIRONMENTS, Ph.D. Thesis, Comberia University, March., 2010.

[36] R. SARKER, M. MOHAMMADIAN and X. YAO, EVOLUTIONARY OPTIMIZATION, 2003.

[37] T. Blackwell, "Particle swarms and population diversity II: Experiments," GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, p. 14–18, 2003.

[38] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments—a survey," in IEEE Transactions on Evolutionary Computation, 2005.

[39] H. Cobb, "An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments," Technical Report AIC-90-001, Naval Research Laboratory, Washington, 1990.

[40] F. Vavak, T. Fogarty and K. Jukes, "A genetic algorithm with variable range of local search for tracking changing environments," *In Parallel Problem Solving from Nature*, pp. 376–385, 1996.

[41] f. Vavak and k. Jukes, "Performance of a genetic algorithm with variable local search range relative to frequency for the environmental changes," in In International Conference on Genetic Programming, 1998.

[42] J. Grefenstette, "Genetic algorithms for changing environments," *In Parallel Problem Solving from Nature*, pp. 137–144, 1992.

[43] J. Grefenstette and L. Connie, "An approach to anytime learning," in In International Conference on Machine Learning, 1992.

[44] N. Mori, H. Kita and Y. Nishikawa, "Adaptation to a changing environment by means of the thermodynamical genetic algorithm," *In Parallel Problem Solving from Na*ture, pp. 513–522, 1996.

[45] W. Cedeno and R. Vemuri, "On the use of niching for dynamic landscapes," 1997.

[46] S. Yang, Genetic algorithms with elitism-based immigrants for changing optimization problems, *In Applications of Evolutionary Computing: EvoWorkshops*, 2007, pp. 627–636.

[47] C. Ryan, "Diploidy without dominance," *In Nordic Workshop on Genetic Algorithms*, pp. 45–52, 1997.

[48] S. Yang, "Explicit memory schemes for evolutionary algorithms in dynamic environments," In Evolutionary Computation in Dynamic and Uncertain Environments. Springer, 2007.

[49] S. Yang., "Non-stationary problems optimization using the primal-dual genetic algorithm," *In Congress on Evolutionary Computation*, pp. 2246–2253, 2003.

[50] A. Younes, "Adapting Evolutionary Approaches For Optimization in Dynamic Environments," A thesis presented to the University of Waterloo in fulfillment of the thesis requirement for the degree of Doctor of Philosophy in Systems Design Engineering, Waterloo, Ontario, Canada, 2006.

[51] R. Morrison, Performance Measurement in Dynamic Environments, 2015.

[52] M. Zarei, H. Parvin and M. Dadvar, "A New Method to Optimize Dynamic Environments with Global Changes Using the Chickens-Hen' Algorithm," *Springer International Publishing AG*, pp. 331-340, 2017.

[53] A. Simoes, "Improving Memory Based Evolutionary Algorithm for Dynamic Environments, Ph.D. Thesis, Comberia University, March., 2010.

محسن مرادی مدرک کارشناسی خود را در سال ۱۳۸۵ در رشته مهندسی کامپیوتر گرایش نرم‌افزار از دانشگاه آزاد اسلامی شیراز و مدرک کارشناسی ارشد خود را در سال ۱۳۸۸ در رشته مهندسی کامپیوتر گرایش نرم افزار از دانشگاه آزاد علوم تحقیقات خوزستان و در سال ۱۳۹۷ مدرک دکترای خود را در رشته مهندسی کامپیوتر گرایش سیستم‌های نرم‌افزاری از دانشگاه آزاد اسلامی یاسوج دریافت کرد. حوزه‌های تخصصی ایشان شبکه و الگوریتم‌های تکاملی است. وی تاکنون بیش از چهل مقاله علمی در نشریات و کنفرانس‌های معتبر داخلی و خارجی به چاپ رسانیده‌اند و چندین کتاب چاپ کرده‌اند.

نشانی رایانامه ایشان عبارت است از:

Mohsen2145@yahoo.com

صمد نجاتیان مدرک کارشناسی خود را در سال ۱۳۸۲ در رشته مهندسی برق گرایش الکترونیک از دانشگاه سیستان و بلوچستان و مدرک کارشناسی ارشد خود را در سال ۱۳۸۶ در رشته مهندسی برق گرایش مخابرات از دانشگاه مشهد و در سال ۱۳۹۳ مدرک دکترای خود را در رشته مهندسی برق گرایش مخابرات از دانشگاه یو تی ام مالزی .کووالالامپور، مالزی دریافت کرد. وی دانشیار و عضو هیأت علمی تمام‌وقت گروه برق دانشگاه آزاد اسلامی واحد یاسوج است، از جمله سوابق ایشان معاون پژوهش و فن‌آوری دانشگاه آزاد اسلامی واحد یاسوج و رئیس باشگاه پژوهشگران جوان دانشگاه آزاد اسلامی واحد یاسوج است. حوزه‌های تخصصی ایشان برق، مخابرات و هوش مصنوعی است. وی تاکنون بیش از هفتاد مقاله علمی در نشریات و کنفرانس‌های معتبر داخلی و خارجی به چاپ رسانیده است.

نشانی رایانامه ایشان عبارت است از:

samad.nej.2007@gmail.com

حمید پروین مدرک کارشناسی خود را در سال ۱۳۸۵ در رشته مهندسی کامپیوتر گرایش نرم‌افزار از دانشگاه شهید چمران اهواز و مدرک کارشناسی ارشد و دکترای خود را در سال ۱۳۸۷ و ۱۳۹۲ در رشته مهندسی کامپیوتر گرایش هوش مصنوعی از دانشگاه علم و صنعت ایران دریافت کرد. وی تاکنون بیش از یکصد مقاله علمی در نشریات و کنفرانس-های معتبر داخلی و خارجی به چاپ رسانیده است و چندین کتاب چاپ کرده‌اند.

نشانی رایانامه ایشان عبارت است از:

parvin@iust.ac.ir

کرم الله باقری فرد مدرک کارشناسی خود را در سال ۱۳۸۴ در رشته مهندسی کامپیوتر گرایش نرم‌افزار از دانشگاه اصفهان و مدرک کارشناسی ارشد و دکترای خود را به‌ترتیب در سال‌های ۱۳۸۷ و ۱۳۹۵ از دانشگاه نجف آباد و اراک در رشته مهندسی کامپیوتر گرایش نرم افزار دریافت کرد. وی از سال ۱۳۸۵ تاکنون عضو هیأت علمی بخش مهندسی کامپیوتر دانشگاه آزاد اسلامی واحد یاسوج است. حوزه‌های تخصصی ایشان داده‌کاوی، یادگیری ماشین و سامانه‌های

پیشنهاددهنده است. وی تاکنون بیش از هفتاد مقاله علمی در نشریات و کنفرانس‌های معتبر داخلی و خارجی به چاپ رسانیده است.
نشانی رایانامه ایشان عبارت است از:
k.bagheri@iauyasooj.ac.ir

سیده وحیده رضایی دارای مدرک دکترای ریاضی است. عضو هیأت علمی دانشگاه آزاد اسلامی واحد یاسوج است. وی تاکنون بیش از چهل مقاله علمی در نشریات و کنفرانس‌های معتبر داخلی و خارجی به چاپ رسانیده است.
نشانی رایانامه ایشان عبارت است از:
vahidehrezaie80@gmail.com