Journal of New Technologies in Distributed Systems and Algorithmic Computing
Islamic Azad University of Sabzevar
E-ISSN: 3115-705X
https://sanad.iau.ir/journal/ntds

Reaserch Article

Evaluating and Comparing the Performance of the Alibaba and Forty Thieves Algorithm Compared to Selected Meta-heuristic Algorithms

Fatemeh Delbari [1] | Ali Akbar Neghabi*[2] | Jalal Iziy [3]

[1]Department of Computer Engineering and Information Technology, Islamic Azad University, Sabzevar Branch, Sabzevar, Iran, Fatemedelbari73@gmail.com

[2]Department of Computer Engineering and Information Technology, Islamic Azad University, Sabzevar Branch, Sabzevar, Iran, Aa_neghabi@iaus.ac.ir

[3]Department of Computer Engineering and Information Technology, Islamic Azad University, Sabzevar Branch, Sabzevar, Iran, J.iziy90@gmai.com

Corresponding Author
*Ali Akbar Neghabi, Assistant Professor, Department of Computer Engineering and Information Technology, Islamic Azad University, Sabzevar Branch, Sabzevar, Iran, Aa_neghabi@iaus.ac.ir

Main Subjects: Metaheuristic Algorithms
Received: 29 November 2024
Revised: 31 December 2024
Accepted: 23 March 2025

https://doi.org/10.82195/ntds.2025.1192011

Abstract

Nature has always been a suitable model for humans so that researchers can model the body structure and movement model of animals and biological behavior of creatures, creating algorithms that are inspired by them and can be used to optimize and solve complex problems. The Alibaba and Forty Thieves algorithm is one of the meta-heuristic algorithms that is inspired by the story of Alibaba and the Forty Thieves. In this algorithm, the city where the story takes place is considered as the search space and the thieves act as the search agents. Also, Alibaba is considered as the target and optimal answer to the problem. Given the breadth of meta-heuristic algorithms and their wide application in various fields, it seems necessary to investigate the performance of such algorithms. This paper aims to compare the performance of the Forty Thieves and Alibaba algorithm with seven other algorithms including the Herd of Horses, Harris's Hawk, African Vulture, Ant Lion, Bat, Firefly, and Whale optimization algorithms. The CEC 2017 standard function set and three criteria such as best solution, standard deviation, and average execution time have been used to evaluate the selected algorithms' performance. The simulation results indicate that the African Vulture and Harris's Hawk algorithms performed significantly better than the Forty Thieves and Alibaba algorithm, as well as other algorithms, in most functions. Overall, the performance of the Forty Thieves and Alibaba algorithm is better than that of the Ant Lion, Bat, and Firefly algorithms.

Keywords: Alibaba and Forty Thieves, Meta-Heuristic Algorithms, Harris's Hawk, African Vulture.

فناوری های نوین در سیستم های توزیع شده و محاسبات الگوریتمی

دانشگاه آزاد اسلامی واحد سبزوار

E-ISSN: 3115-705X
https://sanad.iau.ir/journal/ntds

پژوهشی

ارزیابی و مقایسه ی عملکرد الگوریتم علی‌بابا و چهل دزد با برخی الگوریتم‌های فراابتکاری

فاطمه دلبری [۱] | **علی‌اکبر نقابی*** [۲] |**جلال ایزی** [۳]

[۱]گروه مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه آزاد اسلامی واحد سبزوار، سبزوار، ایران،
Fatemedelbari73@gmail.com

[۲]گروه مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه آزاد اسلامی واحد سبزوار، سبزوار، ایران،
Aa_neghabi@iaus.ac.ir

[۳]گروه مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه آزاد اسلامی واحد سبزوار، سبزوار، ایران،
J.iziy90@gmai.com

نویسنده مسئول
***علی اکبر نقابی**، استادیار، گروه مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه آزاد اسلامی واحد سبزوار، سبزوار، ایران،
Aa_neghabi@iaus.ac.ir

موضوع اصلی: الگوریتم‌های فراابتکاری
تاریخ دریافت: ۱۴۰۳/۹/۹
تاریخ بازنگری: ۱۴۰۳/۱۰/۱۱
تاریخ پذیرش: ۱۴۰۴/۱/۳

چکیده:

طبیعت همیشه الگویی مناسب برای انسان‌ها بوده است تا محققان بتوانند با الگوبرداری از ساختار بدن و مدل حرکت حیوانات و رفتارهای زیستی موجودات، الگوریتم‌هایی را به وجود آورند که الهام‌گرفته از آن‌ها بوده و برای بهینه‌سازی و حل مسائل پیچیده به کار برده شوند. الگوریتم علی‌بابا و چهل دزد یکی از الگوریتم‌های فراابتکاری است که از داستان علی‌بابا و چهل دزد الهام گرفته‌شده است. در این الگوریتم، شهری که داستان در آن اتفاق می‌افتد به‌عنوان فضای جستجو در نظر گرفته‌شده و دزدها به‌عنوان عامل‌های جستجو عمل می‌کنند. همچنین، علی‌بابا به‌عنوان هدف و پاسخ بهینه مسئله در نظر گرفته‌شده است. باتوجه‌به گستردگی الگوریتم‌های فراابتکاری و نیز کاربرد وسیع آن‌ها در حوزه‌های مختلف، بررسی عملکرد چنین الگوریتم‌هایی لازم به نظر می‌رسد. پژوهش حاضر قصد دارد عملکرد الگوریتم چهل دزد و علی‌بابا را با هفت الگوریتم دیگر شامل الگوریتم‌های گله اسب، شاهین هریس، کرکس آفریقایی، شیر مورچه، خفاش، کرم شب‌تاب و الگوریتم نهنگ مقایسه نماید. برای ارزیابی عملکرد الگوریتم‌های انتخابی از مجموعه توابع استاندارد CEC2017 استفاده‌شده و همچنین برای مقایسه عملکرد از سه معیار بهترین پاسخ، انحراف معیار و میانگین زمان اجرا استفاده‌شده است. نتایج شبیه‌سازی نشان می‌دهد که الگوریتم کرکس آفریقایی و شاهین هریس به طور قابل‌توجهی در مقایسه با الگوریتم چهل دزد و علی‌بابا و همچنین در قیاس با سایر الگوریتم‌ها در اکثر توابع عملکرد بهتری داشته‌اند. همچنین، به طور معمول این الگوریتم عملکرد بهتری از الگوریتم‌های شیر مورچه، خفاش و کرم شب‌تاب داشته است.

https://doi.org/10.82195/ntds.2025.1192011

کلیدواژه‌ها: الگوریتم علی‌بابا و چهل دزد، الگوریتم‌های فراابتکاری، الگوریتم شاهین هریس، الگوریتم کرکس آفریقایی.

۱-مقدمه

بهینه‌سازی یکی از مفاهیم اساسی در علوم کامپیوتر، مهندسی، اقتصاد و سایر رشته‌ها است. این فرایند به کمک الگوریتم‌ها و روش‌های مختلف، بهترین راه‌حل ممکن برای یک مسئله خاص را پیدا می‌کند. روش‌های مرسوم حل مسائل بهینه‌سازی برای بهینه‌سازی فرایندها، تصمیم‌گیری‌ها، طراحی سیستم‌ها و حتی برنامه‌های کاربردی مختلف استفاده می‌شوند[۱]. این امر به کارایی و بهره‌وری بالاتر در مسائل پیچیده کمک می‌کند و درنهایت به صرفه‌جویی در زمان، هزینه و منابع منجر می‌شود. بسیاری از مسائل بهینه‌سازی پیچیده‌تر و مشکل‌تر از آن هستند که با روش‌های مرسوم بهینه‌سازی نظیر روش‌های دقیق ریاضی و نظایر آن قابل‌حل باشند. ازجمله راه‌حل‌های موجود در برخورد با این‌گونه مسائل، استفاده از الگوریتم‌های تقریبی، ابتکاری و فراابتکاری می باشد. این الگوریتم‌ها یک جواب بهینه را در زمان مناسبی ارائه می دهند [۲].

الگوریتم‌های فراابتکاری، الگوریتم‌هایی هستند که جواب‌ها را باکیفیت بالا و در زمان کوتاهی برای مسائل بهینه‌سازی پیچیده ارائه می‌دهند[۳]. این الگوریتم‌ها جواب‌ها را باکیفیت بالا و در زمان کوتاهی برای مسائل بهینه‌سازی پیچیده ارائه می‌دهند[۳]. هرچند ضمانتی برای دستیابی به جواب دقیق با استفاده از این الگوریتم‌ها وجود ندارد ولی توانایی بالای آن‌ها در دستیابی به جواب‌های نزدیک به بهینه در زمان کوتاه برای این مسائل موجب شهرت فراوان آن‌ها شده است. درواقع الگوریتم‌های فراابتکاری یکی از انواع الگوریتم‌های بهینه‌سازی تقریبی هستند و قابل کاربرد در طیف گسترده‌ای از مسائل می‌باشند. اخیراً استفاده از الگوریتم‌های فراابتکاری برای حل مسائل بهینه‌سازی پیوسته و گسسته به‌طور شگفت‌آوری رشد فزاینده‌ای داشته است. مهم‌ترین عاملی که سبب شده است استفاده از این الگوریتم‌ها موردتوجه جوامع علمی مختلف باشد، سادگی اعمال شدن و کاربرد آن‌ها برای حل مسائل با ماهیت بسیار متفاوت است[۴].

در سال‌های اخیر، الگوریتم‌های الهام گرفته از طبیعت توجه زیادی را به خود جلب کرده‌اند[۵]. الگوریتم بهینه‌سازی چهل دزد و علی‌بابا[۱][۶] یکی از این روش‌ها است که از رفتار جستجوی دزدان در داستان «علی‌بابا و چهل دزد» الهام گرفته است. در این داستان علی‌بابا با استفاده از زیرکی و خلاقیت، موفق به نفوذ به غار دزدان و دستیابی به گنجینه‌های آن‌ها می‌شود. این داستان الگویی برای توسعه الگوریتم چهل دزد و علی‌بابا است که با استفاده از راهکارهای زیرکانه و خلاقانه برای یافتن بهترین راه‌حل در مسائل بهینه‌سازی مورداستفاده قرار می‌گیرد. هدف کلی این الگوریتم ارائه یک روش بهینه‌سازی جدید با تقلید از داستان علی‌بابا و چهل دزد به‌عنوان الگوی هماهنگ رفتار اجتماعی اعمال انسان است.

باتوجه‌به گستردگی کاربرد و اهمیت الگوریتم‌های فراابتکاری، ارزیابی عملکرد آن‌ها امری ضروری و حیاتی است. این ارزیابی‌ها معمولاً از طریق آزمایش بر روی مسائل و مقایسه نتایج حاصل با سایر روش‌ها انجام می‌شود. بررسی کارایی و قابلیت‌های الگوریتم‌های فراابتکاری می‌تواند به شناسایی نقاط قوت و ضعف آن‌ها کمک کرده و امکان بهینه‌سازی و بهبود کارایی آن‌ها را فراهم سازد. ارزیابی دقیق الگوریتم‌های فراابتکاری، همچنین می‌تواند راهنمایی‌های ارزشمندی برای انتخاب مناسب‌ترین الگوریتم‌ها در مسائل مختلف علمی و صنعتی ارائه دهد[7]. الگوریتم بهینه‌سازی چهل دزد و علی‌بابا اولین بار در پژوهش مالک بریک و همکاران در سال ۲۰۲۱ معرفی‌شده است. در آن پژوهش، عملکرد الگوریتم با الگوریتم‌های شعله پروانه[۲][۸]، جستجوی کلاغ[۳][۹]، گرگ خاکستری[۴][۱۰]، منطق فازی[۵][۱۱]، کلونی مورچگان[۶][۱۲]، ازدحام ذرات[۷][۱۳]، جستجوی گرانشی[۸][۱۴]، ژنتیک[۹][۱۵] و الگوریتم تکامل تفاضلی[۱۰][۱۶] مورد مقایسه قرارگرفته است. با توجه به اهمیت کاربرد الگوریتم‌های فراابتکاری در حل مسائل مختلف، بررسی و مقایسه عملکرد این الگوریتم ها ضروری بنظر می‌رسد. در تحقیقات لنجام شده تاکنون، عملکرد

[1] Ali Baba and the Forty Thieves (AFT)
[2] Moth-flame Optimization Algorithm (MFO)
[3] Crow Search Algorithm (CSA)
[4] Grey Wolf Optimizer (GWO)
[5] Fuzzy Logic Algorithm (FLOA)
[6] Ant Colony Optimization (ACO)
[7] Particle Swarm Optimization (PSO)
[8] Gravitational Search Algorithm (GSA)
[9] Genetic Algorithm (GA)
[10] Differential Evolution (DE)

الگوریتم چهل دزد و علی‌بابا با الگوریتم‌های گله اسب[۱][۱۷]، شاهین هریس[۲][۱۸]، کرکس آفریقایی[۳][۱۹]، شیرمورچه[۴][۲۰]، خفاش[۵][۲۱]، کرم شب‌تاب[۶][۲۲] و الگوریتم نهنگ[۷][۲۳] مقایسه نشده است. بنابر این، این پژوهش قصد دارد عملکرد الگوریتم چهل دزد و علی‌بابا را با ۷ الگوریتم اشاره شده مقایسه نماید. برای بررسی عملکرد الگوریتم‌های انتخابی از ۲۳ تابع استاندارد مجموعه CEC2017 استفاده‌شده است. عملکرد این الگوریتم ها با استفاده از سه معیار بهترین پاسخ، انحراف معیار و میانگین زمان اجرا مقایسه شده است. این معیارها کمک می‌کنند تا نقاط قوت و ضعف هر الگوریتم شناسایی‌شده و از این اطلاعات برای بهبود کاربرد آن‌ها در حوزه‌های مختلف بهره برده شود. در ادامه الگوریتم‌های فراابتکاری موردبحث در این پژوهش معرفی می‌شوند.

۲- مرور ادبیات تحقیق

۲-۱-الگوریتم‌های فراابتکاری

الگوریتم‌های فراابتکاری، دسته‌ای از الگوریتم‌های بهینه‌سازی می‌باشند که الهام‌گرفته از فرایندهای طبیعی و تکاملی، مانند تکامل تفاضلی، مهاجرت ابرها و تکامل ذرات هستند[۲۴]. این الگوریتم‌ها برای حل مسائل بهینه‌سازی در مسائلی که فضای جستجوی پیچیده‌ای دارند، بسیار مؤثر هستند. تفاوت مهم الگوریتم‌های فراابتکاری با روش‌های بهینه‌سازی سنتی در این است که آن‌ها به جستجو در فضای سراسری می‌پردازند و بهترین راه‌حل‌ها را به‌طور تصادفی پیدا می‌کنند. در عمل، این الگوریتم‌ها به‌صورت موازی یا متوالی فرایند بهینه‌سازی را ادامه می‌دهند تا به یافتن یک راه‌حل بهینه یا نزدیک به بهینه برسند. این الگوریتم‌ها، به دلیل قابلیت دسترسی به راه‌حل‌های تقریباً بهینه در زمان‌های اجرای مناسب، برای حل مسائل بهینه‌سازی سخت طراحی‌شده و به دلیل قابلیت کاربرد در طیف گسترده‌ای از مسائل، بسیار موردتوجه قرار گرفته اند. الگوریتم‌های ژنتیک، بهینه‌سازی ازدحام ذرات و بهینه‌سازی کلونی مورچگان، نمونه‌های معروف از این دسته الگوریتم‌ها هستند که با موفقیت برای حل مسائل پیچیده استفاده شده‌اند[۲۵].

۲-۲-مروری بر الگوریتم‌های فراابتکاری در این پژوهش

در این بخش، توضیح مختصری در خصوص الگوریتم‌هایی که در این پژوهش استفاده شده است، ارائه خواهد شد.

۲-۲-۱-الگوریتم چهل دزد و علی‌بابا

الگوریتم چهل دزد و علی‌بابا توسط مالک بریک، محمد هاشم ریالت و حسین الذوبی در سال ۲۰۲۱ پیشنهادشده است. این الگوریتم از داستان علی‌بابا و گنج سارقان الهام‌گرفته است. در این داستان، یک تقابل بین علی‌بابا و خدمتکارش و نیز چهل دزد برقرار است. رفتار سارقان در داستان و همچنین روش‌های هوشمندانه‌ای که خدمتکار علی‌بابا (مرجانه) برای نجات او به کار می‌برد، رفتارهایی هستند که این الگوریتم بهینه‌سازی را شبیه‌سازی می‌کند[۲۶]. در این الگوریتم، فضای جستجو معادل با شهری است که داستان در آن اتفاق می‌افتد و دزدها به‌عنوان عامل‌های جستجو در نظر گرفته می‌شوند. علی‌بابا نیز به‌عنوان پاسخ بهینه مسئله در نظر گرفته می‌شود. اصول زیر از داستان علی‌بابا به مفروضات اساسی الگوریتم تبدیل‌شده است:

- چهل دزد به‌صورت گروهی با هم همکاری می‌کنند و از یک نفر از سارقان راهنمایی می‌گیرند تا خانه علی‌بابا را پیدا کنند. این اطلاعات ممکن است درست باشد یا نباشد.
- چهل دزد مسافتی از مسافت اولیه را طی می‌کنند تا بتوانند خانه علی‌بابا را پیدا کنند.
- مرجانه می‌تواند بارها با روش‌های زیرکانه سارقان را فریب دهد تا از علی‌بابا در برابر دزدان محافظت کند.

[5]Horse herd Optimization Algorithm (HOA)
[6]Harris Hawks Optimizer (HHO)
[7]African Vulture Optimization Algorithm (AVOA)
[8]Ant Lion Optimizer (ALO)
[9]Bat Algorithm (BA)
[10]Firefly Algorithm (FA)
[11]Whale Optimization Algorithm (WOA)

در الگوریتم چهل دزد و علی‌بابا، سه وضعیت اساسی وجود دارد که ممکن است هنگام جستجوی سارقان برای پیداکردن علی‌بابا رخ دهد که عبارت‌اند از [۲۷]:

- وضعیت اول: در مرحله اول، سارقان ممکن است بتوانند علی‌بابا را با کمک اطلاعاتی که از شخص دیگری به دست آورده‌اند، ردیابی کنند. در این صورت مکان‌های جدید سارقان را می‌توان از رابطه (۱) به دست آورد:

$$x_{t+1}^{i} = gbest + [Td_t(best_t^i - y_t^i)\, r1 + Td_t\,(y_t^i - m_t^{a(i)})r2]sgn(rand - 0.5) \quad (۱)$$

که در آن x_{t+1}^{i} موقعیت i امین دزد در لحظه جدید است. y_t^i موقعیت علی‌بابا نسبت به دزد در هر تکرار است. $gbest$ نشان‌دهنده بهترین موقعیت سراسری است که تاکنون توسط هر دزدی در هر تکرار به‌دست‌آمده است، $m_t^{a(i)}$ نشان‌دهنده سطح هوش و ذکاوت مرجانه است که برای گمراه کردن سارقان می باشد. Td_t فاصله ردیابی سارقان در زمان در آن لحظه است. $r2$ و $r1$ اعداد تصادفی هستند که با توزیع یکنواخت بین ۰ و ۱ می‌باشند. sgn برای تغییر جهت فرایند جستجو است.

- وضعیت دوم: در فاز دوم، ممکن است سارقان متوجه شوند که فریب‌خورده‌اند، بنابراین به طور تصادفی فضای جستجوی علی‌بابا را کشف می‌کنند. در این صورت مکان‌های جدید سارقان را می‌توان با استفاده از رابطه (۲) به دست آورد:

$$x_{t+1}^{i} = Td_t\,[(u_i - l_j)\, rnad + l_j] \quad (۲)$$

به‌طوری‌که x_{t+1}^{i} مکان جدید سارقان را نشان می‌دهد، Td_t فاصله ردیابی سارقان در آن لحظه است. l_j و u_i کران بالا و پایین جستجو را نشان می‌دهند. $rnad$ یک عدد تصادفی بین ۰ و ۱ می‌باشد.

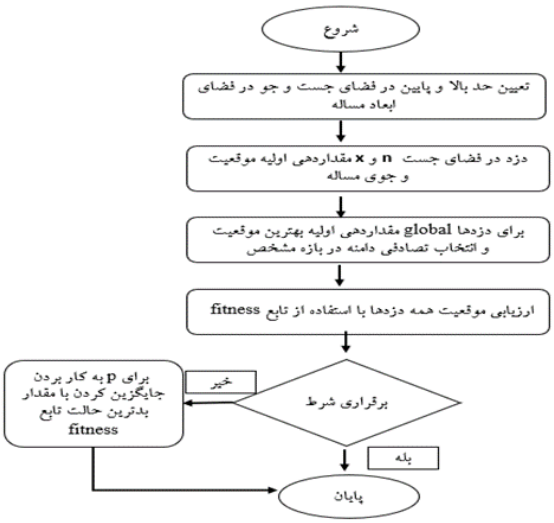
- وضعیت سوم: در فاز سوم، به‌منظور بهبود ویژگی‌های اکتشاف و بهره‌برداری می‌توان از رابطه (۳) استفاده کرد. در این صورت می‌توان مکان‌های جدید سارقان را به شرح زیر به دست آورد:

$$x_{t+1}^{i} = gbest - [Td_t(best_t^i - y_t^i)r1 + Td_t\,(y_t^i - m_t^{a(i)})r2]\; sgn(rand - 0.5) \quad (۳)$$

این رابطه همانند رابطه (۱) می‌باشد با این تفاوت که علامت آن به منفی تغییر پیداکرده است.

دیاگرام الگوریتم علی‌بابا و چهل دزد در شکل ۱ نشان می دهد که این الگوریتم شامل چندین مرحله اصلی زیر است:

- تعیین حد بالا و حد پایین فضای جستجو
- یک جمعیت اولیه از دزدان به طور تصادفی در فضای جستجو پراکنده می‌شوند. هر دزد نمایانگر یک راه‌حل ممکن برای مسئله بهینه‌سازی است.
- هر دزد بر اساس تابع هدف ارزیابی می‌شود. تابع هدف کیفیت هر راه‌حل را اندازه‌گیری می‌کند و به هر دزد یک امتیاز می‌دهد.
- سارقان در محیط‌های محلی خود به جستجو می‌پردازند و بهترین نقاط را برای پنهان‌شدن پیدا می‌کنند.
- بهترین راه‌حل‌های یافته شده به‌عنوان رهبران جدید انتخاب می‌شوند و سایر سارقان به سمت این رهبران حرکت می‌کنند.
- مراحل ارزیابی، جستجوی محلی و به‌روزرسانی جمعیت تا زمانی که معیار توقف برآورده شود، تکرار می‌شوند.



شکل ۱: دیاگرام الگوریتم علی‌بابا و چهل دزد
Figure 1: Algorithm diagram of Alibaba and forty thieves

۲-۲-۲- الگوریتم شاهین هریس

الگوریتم شاهین هریس در سال ۲۰۱۹ توسط علی‌اصغر حیدری ارائه‌شده است. این الگوریتم یک روش بهینه‌سازی مبتنی بر جمعیت است که الهام‌گرفته از رفتار مشارکتی و روش تعقیب و گریز شاهین‌های هریس در غافلگیری طعمه است. در این استراتژی هوشمند، چندین شاهین با همکاری یکدیگر، از جهات مختلف به یک طعمه حمله می‌کنند تا آن را به دام بیندازند. این رفتار شاهین‌های هریس برای حل مسائل بهینه‌سازی مورداستفاده قرار می‌گیرد [۲۸]. شاهین‌های هریس، به‌عنوان یکی از باهوش‌ترین پرندگان در طبیعت شناخته می‌شوند. آن‌ها پرندگان شکاری هستند که در گروه‌های ثابت در نیمه جنوبی آریزونا، ایالات‌متحده زندگی می‌کنند. این پرندگان معمولاً به‌صورت تکی شکار می‌کنند، اما شاهین هریس با فعالیت‌های گروهی و هماهنگ با سایر اعضای خانواده‌اش، از سایر پرندگان شکاری متمایز است. آن‌ها در تعقیب، ردیابی، محاصره و حمله به طعمه نشان داده‌اند که توانایی‌های مبتکرانه‌ای دارند. آن‌ها به‌عنوان شکارچیان هماهنگ و با توانایی‌های برجسته شناخته می‌شوند و مأموریت تیمی خود را با آغاز صبح از محل زندگی‌شان، که بیشتر در درختان بلند و بزرگ است، آغاز می‌کنند. به دنبال حرکات اعضای خانواده‌شان می‌گردند و سعی دارند در حمله آگاهانه‌شان هماهنگ باشند [۲۹]. وقتی شاهین‌های هریس جمع می‌شوند، برخی از آن‌ها به ترتیب تورها یا جستجوهای کوتاهی را انجام می‌دهند و سپس بر روی مکان‌های نسبتاً بلندی فرود می‌آیند. این روش باعث می‌شود که شاهین‌ها گاهی یک حرکت "جهشی" را در سرتاسر منطقه انجام دهند، دوباره به هم متصل شوند، و چندین بار از هم جدا شوند تا به‌طور فعال به دنبال حیوانات تحت پوشش، معمولاً یک خرگوش، بگردند. تاکتیک اصلی شاهین‌های هریس برای شکار، "پرش غافلگیرانه" یا استراتژی "هفت کشته" است که در آن، چندین شاهین سعی می‌کنند به‌طور هم‌زمان از جهات مختلف به طعمه حمله کنند و بر روی یک خرگوش در حال فرار متمرکز شوند. حمله ممکن است به‌سرعت با گرفتن طعمه ی غافلگیر شده در چند ثانیه تکمیل شود، اما گاهی اوقات، به دلیل قابلیت‌های فرار و رفتار طعمه، شامل شیرجه‌های چندگانه و کوتاه و سریع در نزدیکی طعمه به مدت چند دقیقه هم می‌شود. شاهین‌های هریس می‌توانند انواع مختلفی از سبک‌های تعقیب را نشان دهند که به ماهیت پویای شرایط و الگوهای فرار از طعمه بستگی دارد. در الگوریتم شاهین هریس، شاهین‌های هریس راه‌حل‌های کاندید هستند و بهترین راه‌حل کاندید در هر مرحله به‌عنوان طعمه موردنظر یا تقریباً بهینه در نظر گرفته می‌شود. شاهین‌های هریس به‌طور تصادفی در مکان‌هایی نشسته و منتظرند. اگر شانس q را برای هر استراتژی نشستن در نظر بگیریم، بر اساس دو استراتژی، طعمه شناسایی می‌شود، شاهین‌ها بر اساس موقعیت سایر شاهین‌ها و خرگوش نشسته و منتظر هستند یا بر روی درختان بلند به‌صورت تصادفی (مکان تصادفی در محدوده خانه گروه)، نشسته و منتظر هستند که این رفتار در رابطه (۴) نشان داده شده است.

$$X(t+1) = \begin{cases} X_{rand}(t) - r_1|X_{rand}(t) - 2r_2X(t)| & q \geq 0.5 \\ (X_{rabbit}(t) - X_m(t)) - r_3(LB + r_4(UB - LB)) & q < 0.5 \end{cases} \quad (۴)$$

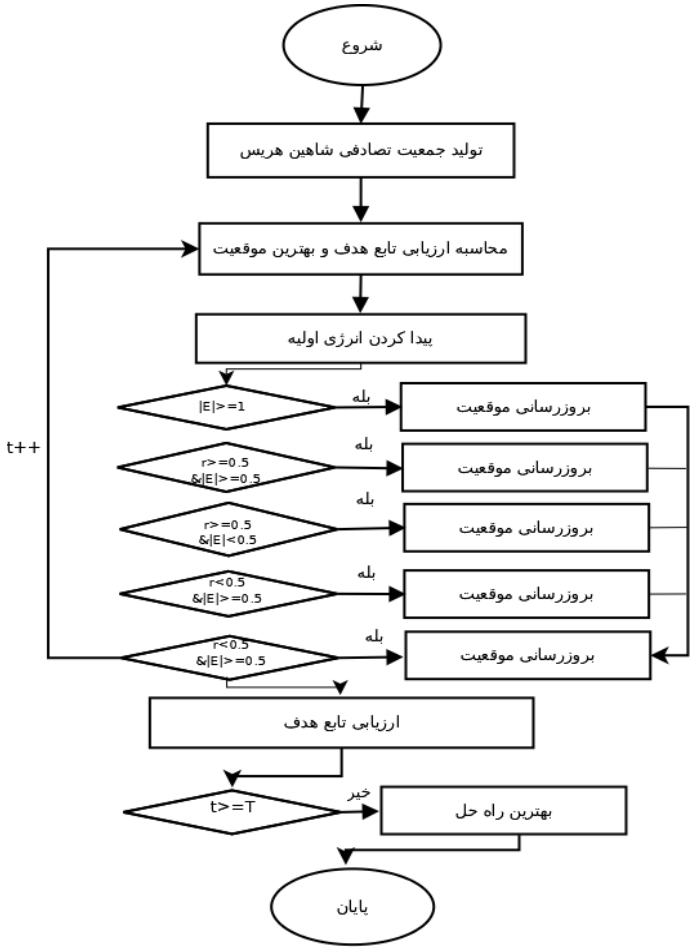
که در آن *X(t+1)* بردار موقعیت شاهین‌ها در تکرار *t* است، *Xrabbit(t)* موقعیت خرگوش، *X(t)* بردار موقعیت فعلی شاهین‌ها، *r1*، *r2*، *r3*، *r4* و *q* اعداد تصادفی در داخل (۰،۱) هستند که در هر تکرار به‌روز می‌شوند، *LB* و *UB* کران‌های بالایی و پایینی متغیرها را نشان می‌دهند، *Xrand(t)* موقعیت یک شاهین تصادفی از جمعیت فعلی و *Xm* میانگین موقعیت از جمعیت فعلی شاهین‌ها است . در قانون اول راه‌حل‌هایی، بر اساس یک مکان تصادفی متأثر از موقعیت قبلی و سایر شاهین‌ها (به‌تصادف) ایجاد می‌شود. اما در قانون دوم معادله ۱ ما یک مدل برای ایجاد مکان‌های تصادفی در محدوده *LB* و *UB* ارائه‌شده است. تفاوت مکان بهترین موقعیت کنونی و میانگین موقعیت گروه را به‌اضافه یک مؤلفه با مقیاس تصادفی بر اساس دامنه متغیرها داریم. متغیرهای *r3* و *r4* ضریب مقیاس برای افزایش ماهیت تصادفی بودن کران بالا و پایین است. در این قانون، یک طول حرکت با مقیاس تصادفی به *LB* اضافه‌شده است سپس، یک ضریب مقیاس‌پذیری تصادفی را برای مؤلفه در نظر گرفته‌شده تا روند متنوع‌سازی جمعیت در مناطق مختلف فضای ویژگی کشف شود. میانگین موقعیت شاهین‌ها با استفاده از معادله (۵) به دست می‌آید.

$$X_m(t) = \frac{1}{N}\sum_{i=1}^{N} X_i\,(t) \quad (۵)$$

که در آن *Xi(t)* مکان هر شاهین را در تکرار *t* و *N* نشان‌دهنده تعداد کل شاهین‌ها است.

بر اساس شکل ۲، الگوریتم شاهین هریس شامل چندین مرحله اصلی است که عبارتند از:

- جمعیت اولیه‌ای از شاهین‌های هریس ایجاد می‌شود.
- محاسبه ارزیابی تابع هدف و پیداکردن بهترین موقعیت
- مقدار انرژی اولیه تعریف و به‌روزرسانی انجام می‌شود.
- در صورت برقراری شرط، موقعیت شاهین‌ها به‌روزرسانی می‌شود.
- اگر شرط برقرار نبود، تابع تناسب برای هر راه‌حل ارزیابی و به بهترین راه‌حل بازمی‌گردیم.



شکل ۲: دیاگرام الگوریتم شاهین هریس

Figure 2: Diagram of Shahin Harris Algorithm

۲-۲-۳- الگوریتم کرکس آفریقایی

کرکس‌های آفریقایی به دو گروه از پرندگان شکاری معروف هستند، کرکس‌های آفریقایی جهان جدید که از آمریکا منشأ می‌گیرند و کرکس‌های آفریقایی جهان قدیمی که در مناطق اروپا، آسیا و آفریقا هستند. در استرالیا و قاره جنوبی قطبی هیچ کرکس آفریقایی وجود ندارد. بیشتر کرکس‌های آفریقایی بال‌دار و بدون پر منظم هستند[۳۰]. درگذشته فکر می‌شد که آن‌ها باید بی‌پر باشند تا از آلودگی در هنگام تغذیه بر روی پوسیدگی‌ها جلوگیری شود، اما تحقیقات اخیر نشان می‌دهد که پوست برهنه نقش مهمی در تنظیم دمای بدن دارد. به همین دلیل، در هوای سرد سر خود را در بدن غوطه می‌زنند و در هوای گرم گردن خود را باز می‌کنند. ویژگی دیگر آن‌ها این است که بر خلاف بیشتر پرندگان دیگر، لانه نمی‌سازند. کرکس‌های آفریقایی به‌ندرت به حیوانات سالم حمله می‌کنند، اما ممکن است حیوان زخمی یا بیمار را بکشند[۳۱]. کرکس های آفریقایی حیوانات مفیدی برای جلوگیری از پوسیدگی اجساد هستند و نقش مهمی در اکوسیستم دارند و نابودی آن‌ها خطرات جدی بهداشتی را برای جوامع انسانی به وجود می‌آورد. الگوریتم بهینه‌سازی کرکس آفریقایی توسط بنیامین عبدالله زاده و همکاران در سال ۲۰۲۱ ارائه‌شده استاین الگوریتم الهام گرفته از رفتار جستجوی کرکس‌های آفریقایی برای یافتن غذا می‌باشد. مراحل این الگوریتم به شرح زیر است:

- تولید جمعیت اولیه: الگوریتم با تولید یک جمعیت اولیه از N کرکس که هر کدام با یک موقعیت تصادفی در فضای جستجوی مسئله مقداردهی می‌شوند، آغاز می‌شود. موقعیت هر کرکس معمولاً به‌عنوان یک بردار از متغیرها تعریف می‌شود.
- ارزیابی اولیه: هر کرکس بر اساس تابع هدف تعیین شده برای مسئله بهینه‌سازی ارزیابی می‌شود. تابع هدف ارزشی را به هر موقعیت نسبت می‌دهد که نشان‌دهنده کیفیت آن راه‌حل است.

- جستجوی غذا: کرکس‌ها از دو استراتژی اصلی در جستجوی غذا استفاده می‌کنند:

- جستجوی محلی: کرکس‌ها در اطراف موقعیت فعلی خود برای یافتن غذا اقدام می‌کنند. این عمل برای جستجوی راه‌حل‌های بهتر در نزدیکی موقعیت کنونی استفاده می‌شود.

- جستجوی سراسری: سپس، کرکس‌ها برای اطمینان از عدم افتادن در دام بهینه‌های محلی، به سمت موقعیت‌های دورتری جابه‌جا می‌شوند. این جستجوی گسترده‌تر به الگوریتم اجازه می‌دهد که نواحی جدید در فضای جستجو را کشف کنند.

- به‌روزرسانی موقعیت‌ها: موقعیت هر کرکس با ترکیب نتایج جستجوی محلی و سراسری به‌روزرسانی می‌شود. این به‌روزرسانی معمولاً به‌صورت تصادفی کنترل‌شده انجام می‌شود؛ یعنی مقدار حرکت هر کرکس بر اساس توزیع احتمالی کنترل می‌شود که می‌تواند ثابت یا متغیر باشد.
- تکرار فرایند: مراحل جستجو و به‌روزرسانی موقعیت‌ها به‌صورت تکراری انجام می‌شوند تا زمانی که شرط خاتمه الگوریتم برقرار شود. در نهایت، وقتی الگوریتم متوقف شد، بهترین موقعیت (راه‌حل) که در طول فرایند به‌دست‌آمده است به‌عنوان خروجی گزارش می‌شود.

۲-۲-۴- الگوریتم گله اسب

الگوریتم فراابتکاری گله اسب یک الگوریتم بهینه‌سازی جدید است که در سال ۲۰۲۱ توسط میر نعیمی و میر جلیلی ارائه‌شده است. این الگوریتم از رفتار اسب‌ها در سنین مختلف در طبیعت الهام گرفته‌شده است. الگوریتم بهینه‌سازی گله اسب نامیده می‌شود، با تقلید از شش ویژگی مهم از عملکردهای اجتماعی اسب‌ها در سنین مختلف ایجادشده است که عبارت‌اند از: چرا رفتن، سلسله‌مراتب، جامعه‌پذیری، تقلید، مکانیسم دفاعی و پرسه زدن است. به دلیل تعداد پارامترهای کنترل بر اساس رفتار اسب‌ها در سنین مختلف، عملکرد بسیار خوبی در حل مسائل پیچیده دارد[۳۲]. اسب‌ها در سنین مختلف رفتار متفاوتی به نمایش می‌گذارند و حداکثر طول عمر یک اسب حدود ۲۵ تا ۳۰ سال است. در الگوریتم گله اسب با توجه به سنشان به ۴ دسته تقسیم‌بندی می‌شوند:.

- δ نشان‌دهنده اسب‌ها در محدوده سنی ۰–۵ سال است،
- γ نشان‌دهنده اسب‌ها در دامنه ۵-۱۰ سال است،
- β نشان می‌دهد اسب‌ها در محدوده سنی ۱۰-۱۵ سال قرار دارند
- و α اسب‌های بزرگ‌تر را نشان می‌دهد.

برای انتخاب سن اسب‌ها باید یک تکرار ماتریس جامع از پاسخ‌ها انجام شود.

در این راستا، می‌توان ماتریس را بر اساس بهترین پاسخ‌ها مرتب کرد و درنتیجه:

- ۱۰٪ درصد اول اسب‌ها از بالای ماتریس طبقه‌بندی‌شده، به‌عنوان اسب α انتخاب می‌شوند.
- ۲۰٪ درصد بعدی در گروه β هستند.
- اسب‌های γ و δ به ترتیب ۳۰٪ و ۴۰٪ اسب‌های باقی مانده را تشکیل می‌دهند.

مراحل الگوریتم گله اسب به این صورت است که در مرحله اول، ایجاد جمعیت اسب‌ها است که در این مرحله، تعدادی اسب به طور تصادفی در فضای جستجو ایجاد می‌شوند. هر اسب یک موقعیت را در فضای جستجو نشان می‌دهد که می‌تواند یک جواب احتمالی برای مسئله بهینه‌سازی باشد. در مرحله دوم، شبیه‌سازی رفتار اسب‌ها در سنین مختلف است، در الگوریتم گله اسب رفتار اسب‌ها را در گروه سنی مختلف شبیه‌سازی می‌کند که به چهار دسته، اسب‌های دلتا که در محدوده سنی ۵ - ۰ سال، اسب‌های گاما در محدوده سنی ۱۰ – ۵ سال، اسب‌های بتا در محدوده سنی ۱۵ - ۱۰ سال و سن اسب‌های آلفا بیشتر از ۱۵ سال است. مرحله سوم، به‌روزرسانی موقعیت اسب‌ها است که موقعیت اسب‌ها بر اساس رفتار اسب‌ها در سنین مختلف و موقعیت بهترین اسب (اسبی که بهترین جواب را پیدا کرده است) به‌روزرسانی می‌شود[۳۳]. الگوریتم گله اسب برای حل مسائل بهینه‌سازی با بُعد بالا پیشنهادشده است. این الگوریتم از شش ویژگی مهم اسب‌ها تقلید می‌کند که عبارت‌اند از: رفتن به چراگاه[1]،

[1] Grazing(G)

زندگی سلسله‌مراتبی[۱]، قابلیت اجتماعی[۲]، تقلید کردن[۳]، مکانیسم دفاعی[۴] و پرسه زدن[۵] می‌باشد. رابطه (۶) حرکت اعمال‌شده به اسب‌ها در هر تکرار را نشان می‌دهد.

$$x^{iter,AGE} = \vec{v}_m^{iter,AGE} + x_m^{(iter-1),AGE} \ , \ AGE = \alpha, \beta, \gamma, \delta \quad (۶)$$

که در آن، $x^{iter,AGE}$ موقعیت اسب m ام، AGE محدوده سنی اسب در نظر گرفته‌شده، $iter$ تکرار فعلی و $\vec{v}_m^{iter,AGE}$ بردار سرعت این اسب را نشان می‌دهد. اسب‌ها رفتارهای مختلفی را در سنین مختلف از خود نشان می‌دهند. حداکثر طول عمر یک اسب در حدود ۲۵ تا ۳۰ سال است. دراین‌رابطه، اسب‌های دلتا[۶] در محدوده سنی ۵ - ۰ سال، اسب‌های گاما[۷] در محدوده سنی ۱۰ − ۵ سال، اسب‌های بتا[۸] در محدوده سنی ۱۵ - ۱۰ سال و سن اسب‌های آلفا[۹] بیشتر از ۱۵ سال است. یک ماتریس جامع از راه‌حل‌ها باید در هر تکرار ایجاد می‌شود تا اسب‌ها انتخاب گردند. در این راستا، ابتدا ماتریس بر اساس بهترین راه‌حل‌ها مرتب می‌شود. سپس از بالای ماتریس مرتب‌شده ۱۰ درصد اولیه از کل اسب‌ها به‌عنوان اسب‌های α انتخاب می‌شوند. ۲۰ درصد بعدی در گروه اسب‌های β قرار می‌گیرند و اسب‌های γ و δ به ترتیب ۳۰ و ۴۰ درصد از مابقی ماتریس راه‌حل‌ها را تشکیل می‌دهند.رابطه (۷) بردار حرکت اسب‌ها در سنین مختلف در طول هر چرخه‌ی الگوریتم را نشان می‌دهد.

$$\begin{aligned}
\vec{v}_m^{iter,\alpha} &= G_m^{iter,\alpha} + D_m^{iter,\alpha} \\
\vec{v}_m^{iter,\beta} &= G_m^{iter,\beta} + H_m^{iter,\beta} + S_m^{iter,\beta} + D_m^{iter,\beta} \\
\vec{v}_m^{iter,\gamma} &= G_m^{iter,\gamma} + H_m^{iter,\gamma} + S_m^{iter,\gamma} + I_m^{iter,\gamma} + D_m^{iter,\gamma} + R_m^{iter,\gamma} \\
\vec{v}_m^{iter,\delta} &= G_m^{iter,\delta} + I_m^{iter,\delta} + R_m^{iter,\delta}
\end{aligned} \quad (۷)$$

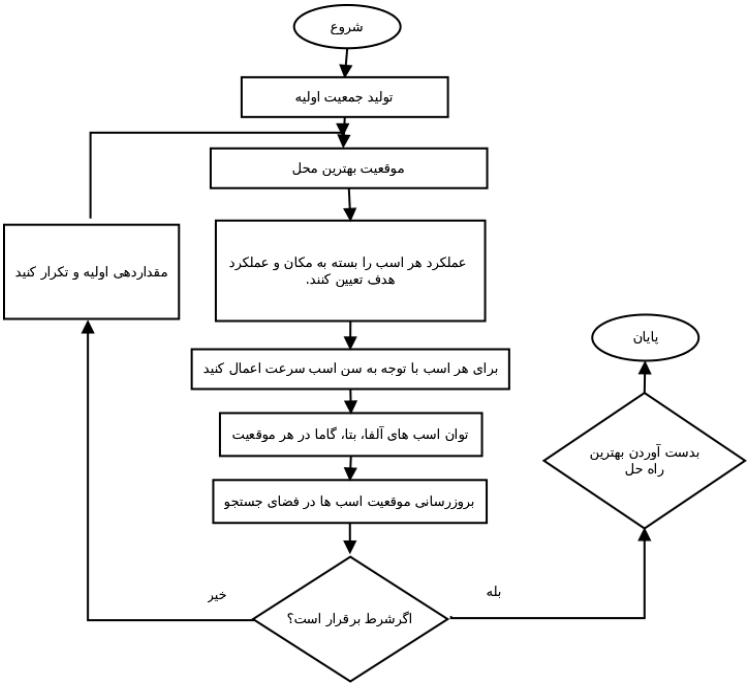
رابطه (۷) نشان می‌دهد که اسب‌ها در چهار گروه سنی تقسیم‌بندی می‌شوند. $\vec{v}_m^{iter,\alpha}$ بردار حرکت اسب‌ها در گروه آلفا را نشان می‌دهد به‌طوری‌که اسب‌های گروه آلفا دارای ویژگی چریدن ($G_m^{iter,\alpha}$) و دفاعی ($D_m^{iter,\alpha}$) می‌باشند. اسب‌های گروه بتا دارای ویژگی‌های چریدن ($G_m^{iter,\beta}$)، زندگیسلسله‌مراتبی ($H_m^{iter,\beta}$)، قابلیت اجتماعی ($S_m^{iter,\beta}$) و ویژگی دفاعی ($D_m^{iter,\beta}$) می‌باشند، $\vec{v}_m^{iter,\beta}$ بردار حرکت اسب‌ها در گروه بتا را نشان می‌دهد. $\vec{v}_m^{iter,\gamma}$ بردار حرکت اسب‌ها در گروه گاما را نشان می‌دهد به‌طوری‌که اسب‌های گروه بتا دارای ویژگی‌های چریدن ($G_m^{iter,\gamma}$)، زندگیسلسله‌مراتبی ($H_m^{iter,\gamma}$)، قابلیت اجتماعی ($S_m^{iter,\gamma}$)، تقلید کردن($I_m^{iter,\gamma}$)، مکانیسم دفاعی ($D_m^{iter,\gamma}$) و ویژگی پرسه زدن ($R_m^{iter,\gamma}$) می‌باشند. اسب‌های گروه دلتا دارای ویژگی‌های چریدن ($G_m^{iter,\delta}$)، تقلید کردن ($I_m^{iter,\delta}$) و پرسه زدن ($R_m^{iter,\delta}$) می‌باشند، $\vec{v}_m^{iter,\delta}$ بردار حرکت اسب‌ها در گروه دلتا را نشان می‌دهد. دیاگرام الگوریتم گله اسب در شکل ۳ نشان داده شده است که شامل مراحل زیر است:

- جمعیت اسب‌ها در فضای ویژگی مقداردهی اولیه می‌شود. این مرحله شامل تعیین تعداد اسب‌ها و موقعیت‌های اولیه آن‌ها است.
- پس از مقداردهی اولیه، تابع تناسب برای ارزیابی عملکرد هر اسب محاسبه می‌شود. این ارزیابی به شناسایی بهترین اسب‌ها کمک می‌کند.
- اولین بهترین اسب شناسایی و به‌روزرسانی می‌شود. این اطلاعات به اسب‌ها اجازه می‌دهد تا بادقت بیشتری در فضای جستجو حرکت کنند.
- مرکز هر اسب از جمعیت تعیین می‌شود. این مرکز به‌عنوان نقطه مرجع برای جستجو و حرکت اسب‌ها عمل می‌کند.
- رده هر اسب ارزیابی می‌شود تا مشخص شود که کدام اسب‌ها در چه موقعیتی قرار دارند.

---

[1] Hierarchy(H)
[2] Sociability(S)
[3] Imitation(I)
[4] Defense mechanism(D)
[5] Roam(R)
[6] δ (Delta)
[7] γ (Gama)
[8] β (Beta)
[9] α (Alpha)

- اگر شرط بهینه‌سازی برقرار باشد و معیار توقف محقق شود، موقعیت اسب‌ها به‌روزرسانی می‌شود. در غیر این صورت، فرایند به مرحله ارزیابی تابع تناسب بازمی‌گردد و مراحل تا زمانی که شرط برقرار باشد، تکرار می‌شوند.



شکل ۳: دیاگرام الگوریتم گله اسب

Figure3: Horse herd algorithm diagram

۲-۲-۵-الگوریتم شیرمورچه

الگوریتم شیر مورچه در سال ۲۰۱۵ توسط سید علی میرجلیلی ارائه‌شده است و از رفتار شکار شیرمورچه‌ها الهام گرفته شده است که در طبیعت با ساختن تله‌های مخروطی شکل، مورچه‌ها را شکار می‌کنند. این تله‌ها به شکل مخروطی هستند و در آن‌ها یک شیرمورچه پنهان شده است. مورچه‌ها زمانی که در این تله‌ها می‌افتند، توسط شیرمورچه به داخل لانه کشیده شده و خورده می‌شوند[۳۴]. شباهت بین الگوریتم شیرمورچه و رفتار شکار شیرمورچه‌ها به این صورت می‌باشد که، مورچه‌ها در الگوریتم شیرمورچه مانند مورچه‌های واقعی در طبیعت هستند که به‌طور تصادفی در فضای جستجو حرکت می‌کنند و تله‌های شیرمورچه در الگوریتم شیرمورچه مانند جواب‌های بهینه مسئله بهینه‌سازی هستند. برای مدل‌سازی چنین فعل و انفعالی مورچه‌ها باید در فضای جستجو حرکت کنند و شیر مورچه‌ها اجازه دارند تا آن‌ها را شکار نمایند. ازآنجایی‌که مورچه‌ها، هنگام جستجوی غذا در طبیعت و به‌صورت کاملاً تصادفی حرکت می‌کنند معادله راه‌رفتن تصادفی برای مدل‌سازی حرکت مورچه‌ها رابطه (۸) می باشد.

$$X(t) = [0, cumsum(2r(t_1) - 1), \dots, cumsum(2r(t_n) - 1] \quad (۸)$$

در رابطه بالا زمانی که *Cumsum* مجموع تجمعی را محاسبه می‌کند؛ n حداکثر تعداد تکرار خواهد بود و همچنین در این حالت t مرحله پیاده‌روی تصادفی را نشان داده و *r(t)* یک تابع تصادفی است و به‌صورت زیر تعریف می‌گردد:

$$r(t) = \begin{cases} 1, & if\ rand > 0.5 \\ 0, & if\ rand \leq 0.5 \end{cases} \quad (۹)$$

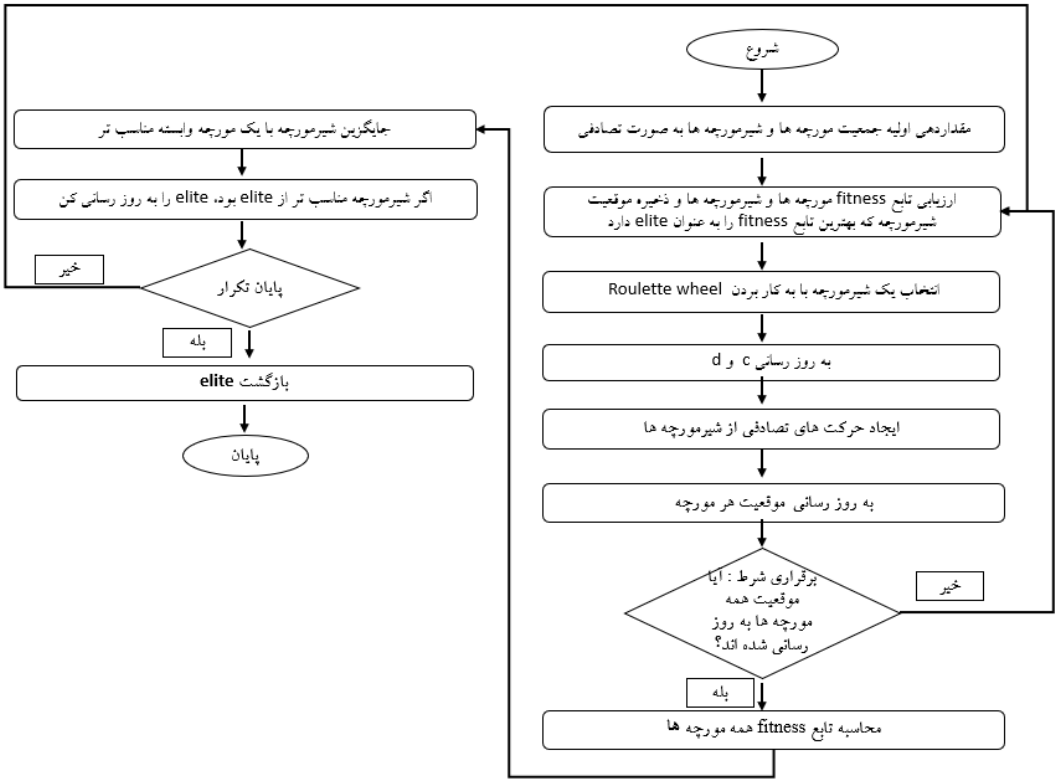
که در آن t مرحله پیاده‌روی تصادفی و rand یک عدد تصادفی است که با توزیع یکنواخت در بازه $[۰,۱]$ تولید می‌شود. در مرحله اول از الگوریتم شیرمورچه جمعیت اولیه ای از مورچه‌ها به‌طور تصادفی ایجاد می شود. هر مورچه یک موقعیت را در فضای جستجو نشان می‌دهد که می‌تواند یک جواب احتمالی برای مسئله بهینه‌سازی باشد. در مرحله دوم، مورچه‌ها بر اساس یک قانون تصادفی در فضای جستجو حرکت می‌کنند. این قانون تصادفی به گونه‌ای است که مورچه‌ها به سمت مورچه‌های دیگر که در موقعیت‌های بهتری قرار دارند، جذب می‌شوند. در مرحله سوم، شیرمورچه‌ها در تله‌هایی که در شن‌ها ایجاد کرده اند، منتظر مورچه‌ها می‌مانند، زمانی که یک مورچه در تله شیرمورچه گرفتار می‌شود، شیرمورچه آن را به سمت لانه خود می‌کشد. در

مرحله چهارم، مورچه‌هایی که در تله شیرمورچه گرفتار شده‌اند موقعیت خود را باتوجه‌به موقعیت شیرمورچه به‌روزرسانی می‌کنند [۳۴]. این کار به مورچه‌ها کمک می‌کند تا به سمت جواب‌های بهینه حرکت کنند. سپس در مرحله پنجم، تکرار مراحل ۲ تا ۴ انجام می شود تا زمانی که معیار توقف الگوریتم برقرار شود. معادلات مربوط به مدل سازی ریاضی تله مورچه ها در رابطه های (۱۰ و ۱۱) نشان داده شده است.

$$C_i^t = Antlion_j^t + C^t \quad (۱۰)$$

$$d_i^t = Antlion_j^t + d^t \quad (۱۱)$$

در این دو رابطه، C^tیک عبارت یا عامل تصحیح اضافی است. C_i^t موقعیت مورچه i ام در تکرار t ام را نشان می‌دهد. موقعیت شیرمورچه j ام انتخاب‌شده در تکرار t ام توسط $Antlion_j^t$ نشان داده می‌شود. d^t بیانگر بیشینه در تمام متغیرها در زمان t است. d_i^t موقعیت جدید مورچه i ام در تکرار t ام است که برابر است با موقعیت شیرمورچه j ام در تکرار t ام به‌علاوه‌ی بردار d^t. شکل ۴، دیاگرام الگوریتم شیرمورچه را نشان می دهد.



شکل ۴: دیاگرام الگوریتم شیرمورچه
Figure 4: diagram of the ant lion algorithm

۲-۲-۶-الگوریتم خفاش

الگوریتم‌های فراابتکاری که معمولاً الهام گرفته‌شده از طبیعت و فرایندهای فیزیکی هستند در حال حاضر به‌عنوان یکی از روش‌های قدرتمند برای حل بسیاری از مسائل بهینه‌سازی پیچیده به کار برده می‌شوند الگوریتم خفاش یکی از الگوریتم‌های فراابتکاری الهام‌گرفته از طبیعت است که در سال ۲۰۱۰ توسط آقای یانگ معرفی گردید. این الگوریتم بر اساس اصول زندگی خفاش‌ها طراحی شده است. خفاش‌ها تنها پستانداران با بال هستند که برای شکار طعمه از انعکاس صدا استفاده می‌کنند. این الگوریتم از خصوصیات ردیابی خفاش‌های کوچک در جستجوی شکار الهام‌گرفته است به‌طوری‌که خفاش‌های کوچک می‌توانند در تاریکی مطلق با انتشار صدا و دریافت آن به شکار طعمه‌های خود بپردازند. برای توسعه این الگوریتم از سه قانون استفاده شده است. اول، همه خفاش‌ها از انعکاس صدا برای تشخیص فاصله استفاده می‌کنند و تفاوت بین مواد غذایی و موانع پیشرو را می‌دانند. دوم، پرواز خفاش‌ها به طور تصادفی با سرعت v_i در مکان x_i با فرکانس ثابت f_{min} و طول موج مختلف λ و بلندی صوت A_0 به‌منظور شکار طعمه صورت می گیرد. همچنین آن‌ها می‌توانند به‌طور خودکار امواج پخش شده و نرخ پالس های

ارسالی خود را ($r\epsilon[0,1]$) باتوجه‌به نزدیکی شکارشان تنظیم کنند. سوم، باتوجه‌به اینکه ممکن است بلندی صدا در بسیاری از جهات مختلف متفاوت باشد لذا فرض می شود که بلندی صدا از بیشترین مقدار تا کمترین مقدار متغیر می‌باشد. مراحل اصلی الگوریتم به شرح زیر است:

۱. مقداردهی اولیه

یک جمعیت اولیه از خفاش‌ها ایجاد می‌شود. هر خفاش با موقعیت و سرعت اولیه خود دارای فرکانس، سرعت و نرخ انتشار پیشنهادی خاصی است.

۲. به‌روزرسانی موقعیت‌ها

برای هر خفاش، فرکانس، سرعت و موقعیت با استفاده از روابط زیر به‌روزرسانی می‌شود:

$$f_i^t = f_{\min} + (f_{\max} - f_{\min})\beta \quad (۱۲)$$

$$v_i^t = v_i^{t-1} + (x_i^t - x^0) \quad (۱۳)$$

$$x_i^t = x_i^{t-1} + v_i^t \quad (۱۴)$$

که در آن $\beta\epsilon[0,1]$ یک بردار تصادفی با توزیع یکنواخت می‌باشد و x^0 بهترین مکان فعلی است که در هر تکرار پس از مقایسه با موقعیت خفاش‌های مجازی انتخاب می‌شود. معمولاً فرکانس f را با $f_{\min} = 0$ و f_{max} =100 در نظر گرفته می شود.

۳. جستجوی محلی

برای تقویت جستجوی محلی، موقعیت هر خفاش با استفاده از پرش‌های کوچک حول بهترین راه‌حل فعلی به‌صورت زیر بروز می‌شود:

$$x_{new} = x_{old} + \epsilon A^t \quad (۱۵)$$

که در آن ε یک عدد تصادفی در بازه [1 -1] است و A^t میانگین بلندی صدای خفاش ها در تکرار t می باشد.

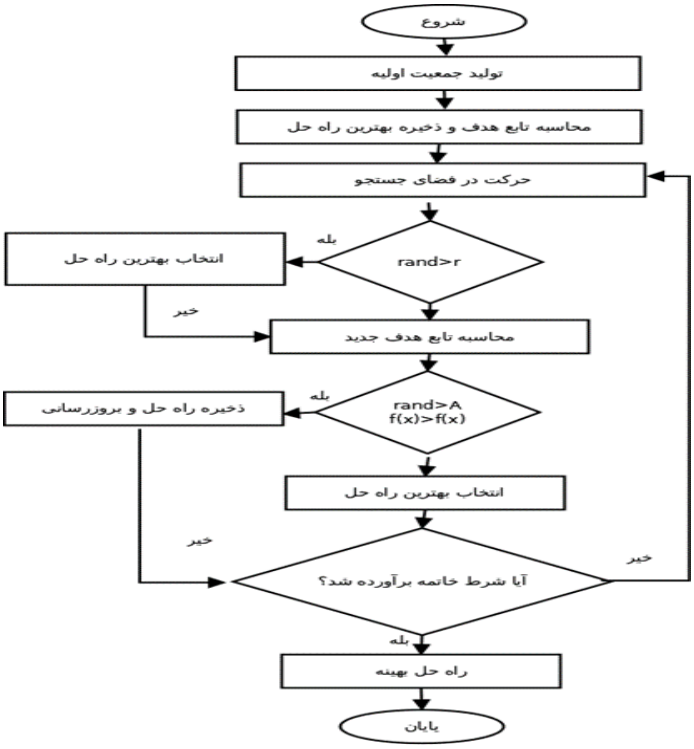
۴. پذیرش راه‌حل‌های جدید

هر خفاش تنها در صورتی به‌روزرسانی موقعیت و سرعت خود را قبول می‌کند که راه‌حل جدید بهبودیافته باشد یا با یک احتمال خاص که توسط شدت انتشار کنترل می‌شود، پذیرفته می‌شود. شدت انتشار از رابطه زیر محاسبه می‌شود:

$$A_i^{t+1} = \alpha A_i^t \quad (۱۶)$$

که α یک پارامتر کمتر از ۱ و بزرگتر از صفر است.

دیاگرام الگوریتم خفاش در شکل (۵) نشان داده شده است.



شکل ۵: دیاگرام الگوریتم خفاش

Figure 5: Bat algorithm diagram

۲-۲-۷- الگوریتم کرم شب‌تاب

تا کنون از الگوریتم کرم شب‌تاب برای حل محدوده وسیعی از مسائل بهینه‌سازی استفاده شده است. این الگوریتم در سال ۲۰۰۷ توسط آقای زین شی یانگ[1] در دانشگاه کمبریج ارائه شد. الگوریتم کرم شتاب، یک مثال بارز از هوش جمعی است که در آن عامل‌هایی که قابلیت چندان بالایی ندارند، در کنار هم و با همکاری یکدیگر می‌توانند نتایج بسیار خوبی به دست آورند. این الگوریتم با الگوبرداری از درخشندگی کرم‌های شب‌تاب و رفتارهای آن‌ها در طبیعت طراحی شده است و از رفتار چشمک‌زن کرم شب‌تاب در طبیعت الهام‌گرفته است که به‌عنوان یک‌شکل از مکانیزم علامت‌دهی برای جذب کرم شب‌تاب دیگر عمل می‌کند. رفتار چشمک‌زن کرم‌های شب‌تاب بر اساس معیارهای جذابیت، روشنایی و فاصله تغییر می‌کند. همه کرم‌های شب‌تاب هم‌جنس هستند و ممکن است در هر دو کرم شب‌تاب یک کرم جذاب باشد. جذابیت آن‌ها با شدت نور آن‌ها متناسب است. کرم شب‌تاب با شدت نور کم، به سمت کرم شب‌تاب با شدت نور بیشتری حرکت می‌کند اگر کرم شب تابی با شدت نور زیاد نباشد، کرم‌های شب‌تاب به طور تصادفی در فضای جستجو حرکت خواهند کرد. در الگوریتم کرم شب‌تاب دو مسئله مهم است: اول، تغییرات شدت نور (I) که از رابطه (۱۷) به دست می‌آید.

$$I = \frac{I_0}{1+\gamma r^2} \qquad (۱۷)$$

به‌طوری که Υ ضریب جذب نور و I_0 شدت نور اولیه، r فاصله و I نور دریافتی است.

میزان جذابیت یک کرم شب‌تاب باید از دید کرم‌های شب‌تاب دیگر مورد قضاوت قرار گیرد؛ لذا مقدار جذابیت باتوجه‌به میزان فاصله (r)، بین دو کرم شب i , j متغیر است. فاکتور جذابیت[2]، مقدار جذابیت کرم شب‌تاب با شدت نوری که توسط کرم‌های شب‌تاب مجاور دیده می‌شود متناسب است، لذا روابط شدت نور برای جذابیت نیز صدق می‌کند که در رابطه (۱۸) نشان داده شده است.

$$\beta = \beta_0 e^{-\gamma r^2} \Rightarrow \beta = \frac{\beta_0}{1+\gamma r^2} \qquad (۱۸)$$

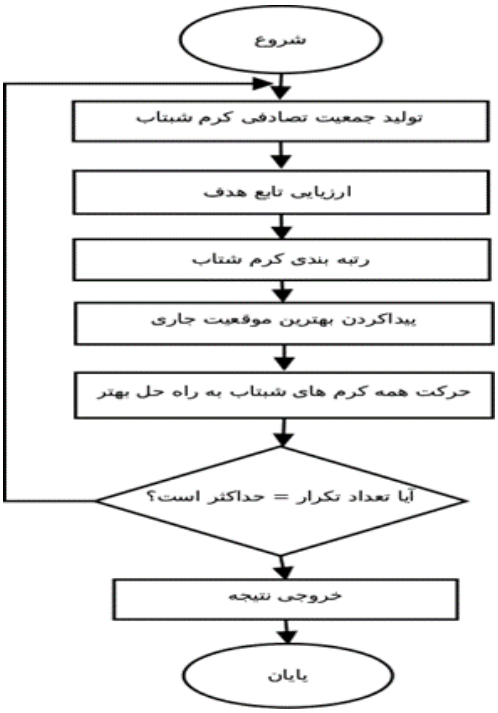
[1] Xin-She Yang

[2] Attractiveness

β فاکتور جذابیت و β_0 جذابیت اولیه است. از فاکتور β برای اندازه گیری میزان کشش و جذابیت میان دو حشره استفاده می‌شود . هر چه میزان فاصله میان دو حشره بیشتر شود، از میزان جذابیت کمتر می‌شود. این قضیه نشان می‌دهد که کرم‌های شب‌تاب به سمت حشراتی می روند که نزدیکتر باشند. دو فاکتور بهتر بودن و نزدیک بودن اولویت دارد. فاصله بین دو کرم شب‌تاب i,j در دوموقعیت xj و xi به‌صورت فاصله دکارتی با کمک رابطه (۱۹) محاسبه می‌شود.

$$r_{ij} = \|x_i - x_j\| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \qquad (۱۹)$$

دیاگرام الگوریتم کرم شب‌تاب در شکل (۶) نمایش داده شده است.



شکل ۶: دیاگرام الگوریتم کرم شب‌تاب

Figure 6: Firefly algorithm diagram

۲-۲-۸- الگوریتم نهنگ

الگوریتم نهنگ بر اساس رفتار شکار نهنگ‌های خاکستری و تکنیک‌های شکار آن‌ها طراحی شده است. نهنگ‌ها به طور اجتماعی زندگی می‌کنند و در گروه‌های مختلفی به شکار می‌پردازند. شکار موردعلاقهٔ آن‌ها، کریل و گروه‌های ماهی کوچک هستند. آنها از تکنیک‌هایی مانند «حملات حلقه‌ای» و «حملات غافلگیرانه» برای شکار استفاده می‌کنند. حملات غافلگیرانه به نهنگ‌ها کمک می‌کند تا با استفاده از حرکات سریع و ناگهانی، به طعمه نزدیک شوند و آن را شکار کنند. این رفتار شبیه به نحوه شکار نهنگ‌ها در طبیعت است. حملات حلقه‌ای به نهنگ‌ها اجازه می‌دهد تا با حرکت دورانی و به شکل حلزونی به سمت بهترین موقعیت (بهترین نهنگ) نزدیک شوند و موقعیت‌های جدیدی را کشف کنند. این تکنیک به نهنگ‌ها کمک می‌کند تا از گیرکردن در نقاط محلی جلوگیری کنند و بتوانند به فضای جستجو گسترش یابند.

مراحل اصلی الگوریتم نهنگ به‌صورت زیر است:

۱. تعیین جمعیت اولیه: تعداد مشخصی از نهنگ‌ها به‌صورت تصادفی در فضای جستجو توزیع می‌شوند.
۲. ارزیابی تابع هدف: هر نهنگ با استفاده از تابع هدف، کیفیت موقعیت خود را ارزیابی می‌کند.
۳. به‌روزرسانی موقعیت: در هر نسل، موقعیت نهنگ‌ها بر اساس دو تکنیک به‌روزرسانی می‌شود:

۱. حملات غافلگیرانه:

این حملات از دو بخش محاسبه موقعیت جدید و حرکات ناگهانی تشکیل می‌شود. در الگوریتم نهنگ فرض می‌کند که بهترین راه‌حل نامزد حال حاضر، شکار هدف بوده و یا نزدیک به حالت مطلوب است. بعد از اینکه بهترین عامل جستجو شناسایی شد، عوامل دیگر جستجو سعی می‌کنند تا مکان خود را نسبت به بهترین عامل جستجو، به‌روزرسانی کنند. محاسبه موقعیت جدید به‌وسیلهٔ معادله زیر بیان می‌شود:

$$Xnew = X + \mathrm{A.D} \quad (۲۰)$$

که X_{new} موقعیت جدید نهنگ و X موقعیت فعلی نهنگ است. A یک پارامتر کنترلی است که شدت حرکت را تعیین می‌کند و D فاصله بین نهنگ فعلی و بهترین نهنگ است که از رابطه زیر محاسبه می‌شود:

$$\mathrm{D} = |\mathrm{C}.X^* - X| \quad (۲۱)$$

که در آن C یک پارامتر تصادفی بین ۰ و ۱ و X^* بهترین موقعیت (بهترین نهنگ) است.

در مرحله حرکت ناگهانی، حرکت نهنگ به‌صورت زیر تعریف می‌شود:

$$Xnew = X + r.(X^* - \mathrm{X}) \quad (۲۲)$$

که در آن، r یک مقدار تصادفی است که می‌تواند بین ۰ و ۱ باشد و شدت حرکت را تنظیم می‌کند.

۲. حملات حلقه‌ای:

حملات حلقه‌ای به نهنگ‌ها اجازه می‌دهد تا با حرکت دورانی و به شکل حلزونی به سمت بهترین نهنگ نزدیک شوند. این حرکت به‌صورت زیر تعریف می‌شود:

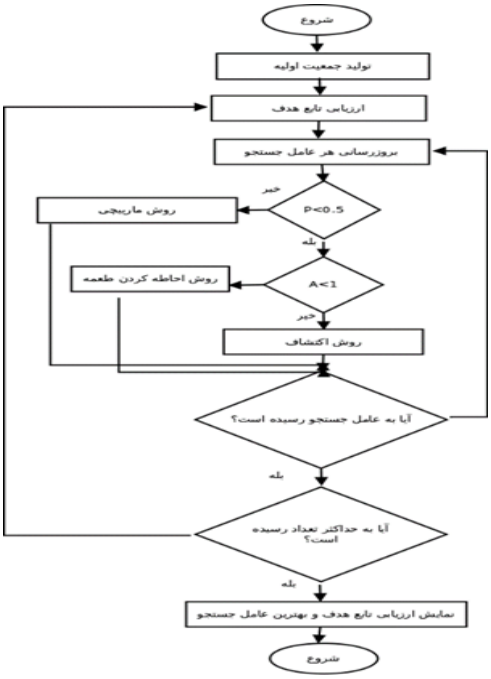
$$Xnew = X_{best} + r.sin(2\pi t) \quad (۲۳)$$

به‌طوری که r یک مقدار تصادفی است که شعاع حرکت را تعیین می‌کند و t زمان یا تعداد تکرارها می‌باشد.

٤. به‌روزرسانی بهترین نهنگ: پس از به‌روزرسانی موقعیت‌ها، بهترین نهنگ (موقعیتی با بهترین کیفیت) شناسایی و ذخیره می‌شود.

٥. مراحل ۲ تا ۴ تا رسیدن به تعداد نسل‌های مشخص یا رسیدن به شرایط توقف تکرار می‌شود.

در شکل (۷)، دیاگرام الگوریتم نهنگ نشان داده شده است.



شکل ۷: دیاگرام الگوریتم نهنگ

Figure 7: Whale optimization algorithm diagram

۳-پیشینه پژوهش

در تحقیقات انجام شده تاکنون عملکرد الگوریتم چهل دزد و علی‌بابا با الگوریتم‌های گله اسب، شاهین هریس، کرکس آفریقایی، شیر مورچه، خفاش، کرم شب‌تاب و نهنگ مورد بررسی قرار نگرفته است. در این بخش برخی از پژوهش هایی که عملکرد الگوریتم‌های فراابتکاری دیگری را مورد بررسی قرار داده اند، آورده شده است.

محمدعلی واقدی و همکاران[۳۵] در سال ۲۰۲۳ پژوهشی با عنوان " ارزیابی عملکرد الگوریتم‌های فراابتکاری بر روی مسائل معیارCEC2021 " ارائه کرده‌اند. در این پژوهش، توابع هدف با افزودن عملگرهایی مانند بایاس، شیفت و چرخش تغییر کرده و عملکرد الگوریتم‌های فراابتکاری توسعه‌یافته ارزیابی شده است. آن‌ها در این تحقیق، الگوریتم‌های فراابتکاری را به دو دسته‌ی توسعه‌یافته و پایه تقسیم کرده‌اند. با استفاده از هر یک از این الگوریتم‌ها، توابع هدف با ابعاد مختلفی را حل کرده و درنهایت، عملکرد این الگوریتم‌ها را به‌طور تجربی بررسی کرده‌اند. نتایج نشان داده است که در بین الگوریتم‌های پایه، الگوریتم بهینه‌سازی مبتنی بر یادگیری آموزش و از میان الگوریتم‌های توسعه‌یافته، الگوریتم تفاضل تکاملی توسعه‌یافته‌ی چند عملگره بهترین عملکرد را نسبت به سایر الگوریتم‌ها ارائه داده‌اند در این تحقیق عملکرد الگوریتم چهل دزد و علی بابا بررسی نشده است.

خدادادی و همکاران[۳۶] در سال ۲۰۲۳، پژوهشی با عنوان "تحلیل مقایسه‌ای عملکرد هشت الگوریتم فراابتکاری برای طراحی بهینه سازه‌های خرپایی با قیود استاتیکی" ارائه کرده‌اند. در این پژوهش، عملکرد الگوریتم‌های بهینه‌سازی کرکس‌های آفریقایی، جهت جریان، حسابی، توزیع عادی تعمیم‌یافته، بهینه‌ساز رنگ تصادفی، بازی هرج و مرج، ساختار کریستال و الگوریتم تولید مواد مقایسه شده است. این الگوریتم‌ها برای بهینه‌سازی اندازه سه ساختار خرپای آلومینیومی استفاده‌شده‌اند که بهینه‌سازی با هدف کاهش وزن اعضای خرپا انجام‌شده است. نتایج شبیه‌سازی نشان داده است که الگوریتم بهینه‌سازی رنگ تصادفی از نظر دقت و نرخ هم‌گرایی بهتر از سایر الگوریتم‌ها عمل کرده است. در این پژوهش نیز عملکرد الگوریتم چهل دزد و علی بابا بررسی نشده است. در این پژوهش از الگوریتم چهل دزد و علی بابا استفاده نشده است.

گورگن و همکاران [37] در سال ۲۰۲۲، پژوهشی با عنوان "تحلیل جامع عملکرد تکنیک‌های بهینه‌سازی فراابتکاری برای طراحی مؤثر چرخه رانکین آلی[1]" ارائه کرده‌اند. در این پژوهش بر روی بهینه‌سازی پارامترهای عملیاتی برای بهبود کارایی سیستم چرخه رانکین آلی تحقیق شده است. از آنجایی که تعیین موثرترین الگوریتم در میان ده‌ها الگوریتم موجود برای حل این مسئله، مانند بسیاری از مسائل بهینه‌سازی دنیای واقعی، کار دشواری است. همچنین، تعریف یک راه‌حل عملی و تعیین روشی که بتواند این راه‌حل را به‌سرعت و قاطعانه پیدا کند، یک چالش بزرگ می باشد، هدف از این تحقیق تعیین مؤثرترین روش‌های بهینه‌سازی فراابتکاری که بتوانند راه‌حل‌های بهینه و قابل اجرا برای حل مسئله طراحی چرخه رانکین آلی را به‌صورت پایدار و سریع بیابند، بوده است. از میان به‌روزترین و قدرتمندترین الگوریتم‌های جستجوی فراابتکاری، ۳۱ الگوریتم انتخاب‌شده و در مطالعات شبیه‌سازی مورداستفاده قرارگرفته‌اند. نتایج نشان داده است که الگوریتم‌های TLABC[2]، DE[3] و PSO[4] در یافتن یک راه‌حل عملی بسیار موفق بوده‌اند. علاوه بر این، الگوریتم TLABC توانسته است یک راه‌حل عملی را در زمان کوتاه‌تری نسبت به جایگزین‌های خود بیابد. در این پژوهش از الگوریتم چهل دزد و علی بابا استفاده نشده است.

پژوهشی با عنوان "تحلیل عملکرد شش الگوریتم فراابتکاری در تولید خودکار مجموعه آزمون برای بهینه‌سازی مبتنی بر پوشش مسیر" در سال ۲۰۲۰ توسط خاری و همکاران[۳۸] انجام‌شده است. آن‌ها بر روی عملکرد شش الگوریتم فراابتکاری به نام‌های: الگوریتم تپه‌نوردی، بهینه‌سازی ازدحام ذرات، الگوریتم کرم شب‌تاب، الگوریتم جستجوی فاخته، الگوریتم خفاش و الگوریتم کلونی زنبور عسل مصنوعی تمرکز کرده‌اند. این الگوریتم‌ها برای بهینه‌سازی پوشش مسیر و پوشش شاخه تولید شده توسط داده‌های آزمایشی، ارزیابی شده اند. هدف از این مطالعه، یافتن بهترین الگوریتم برای محدود کردن تحقیقات آینده در زمینه اتوماسیون آزمایشی برای رویکردهای بهینه‌سازی مبتنی بر پوشش مسیر بوده است. الگوریتم‌ها با استفاده از معیارهای میانگین زمان، بهترین زمان، بدترین زمان و معیارهای محصول مانند پوشش مسیر و مقادیر تابع هدف مجموعه‌های آزمایشی، مقایسه

---

[1] Organic Rankine Cycle

[2] Teaching-Learning-Based Artificial bee Colony

[3] Differential Evolution

[4] Particle Swarm Optimization

شده‌اند. نتایج نشان داده است که الگوریتم کلونی زنبور عسل بهترین عملکرد را داشته و سریع‌ترین الگوریتم بوده است، اما الگوریتم خفاش به‌عنوان کندترین الگوریتم شناخته شده است. در این پژوهش از الگوریتم چهل دزد و علی بابا استفاده نشده است.

۴-روش پژوهش

در این پژوهش برای مقایسه و ارزیابی عملکرد الگوریتم‌های انتخابی، از مجموعه توابع تست استاندارد CEC2017 که دارای ۲۳ تابع می‌باشد؛ استفاده شده است. این توابع مجموعه‌ای از توابع تک‌وجهی، چندوجهی و توابع ترکیبی را شامل می‌شوند. توابع CEC2017 بسیار چالش‌برانگیز هستند و شامل ویژگی‌هایی مانند چندین بهینه محلی، سطح نویز، و پیچیدگی در سطح جهانی هستند. این مجموعه توابع شامل توابع تک‌وجهی *F1* تا *F۷* می‌باشند که ابزاری مناسب برای نشان دادن قدرت الگوریتم در طول مرحله اکتشاف هستند. همچنین، توابع چندوجهی *F۸* تا *F۱۳* معیاری مؤثر برای به چالش کشیدن توانایی بهره‌برداری الگوریتم هستند. درنهایت از توابع ترکیبی *F۱۴* تا *F۲۳* نیز در ارزیابی الگوریتم‌های فراابتکاری می شود. برای ارزیابی عملکرد مقیاس‌پذیری الگوریتم‌های انتخابی، از توابع تست *F1* تا *F۲۳* در ابعاد و تکرارهای مختلف استفاده شده است. ابعاد ۱۰، ۲۰ و ۳۰ برای توابع *F1*تا *F۱۳* و در توابع ترکیبی از *F۱۴* تا *F۲۳* در نظر گرفته می شود. در این پژوهش، شبیه‌سازی با استفاده از نرم‌افزار متلب نسخه B2022 انجام‌شده و سیستم کامپیوتری استفاده‌شده لپ‌تاپ ایسوس نسل ۴ و ۷ هسته‌ای با رم ۸ گیگابایت بوده است. برای پیاده‌سازی هر الگوریتم لازم است پارامترهای اولیه ی آن الگوریتم مقداردهی شوند. مقادیر در نظر گرفته‌شده برای پارامترهای مهم هر الگوریتم در جدول (۱) نشان داده شده است. شرط خاتمه هر یک از الگوریتم‌ها، تعداد ۱۰۰ و ۱۰۰۰ بار اجرا در نظر گرفته‌شده است.

جدول ۱: مقادیر برخی از پارامترهای اولیه الگوریتم‌ها

Table 1: Values of some basic parameters of the algorithms

Algorithms	Parameters	Value
AFT	Marjaneh	**Rand[0,1]**
HOA	Hierarchy factor for β, γ	**0.9 , 0.5**
	Sociability factor for β, γ	**0.2, 0.1**
	Imitation factor for γ, δ	**0.3, 0.3**
	Defense factor for β, γ, α	**0.2, 0.1, 0.5**
	Roam factor for γ, δ	**0.05, 0.1**
	ω	**0.999**
FA	Light Absorption Coefficient,Attraction Coefficient Base Valu, MutationCoefficient, Mutation Coefficient Damping Ratio	**1, 2, 0.2, 0.99**
ALO	antlion by rolette wheel, best antlion so far	**Rand[0,1] , Rand[0,1]**
WOA	Convergence_curve	**zeros(1,Max_iter)**
BA	Loudness, Pulse rate	**0.9 , 0.6**
AVOA	L1, L2, w, P1, P2, P3	**0.8, 0.2, 2.5, 0.6, 0.4, 0.6**
HHO	β	**1.5**

در ارزیابی و مقایسه عملکرد الگوریتم‌های فراابتکاری، معیارهای مختلفی برای سنجش کارایی مورد استفاده قرار می‌گیرند. در این پژوهش از معیارهای زیر برای سنجش کارایی استفاده می‌شود:

۱- بهترین پاسخ[1]: این معیار نشان‌دهنده بهترین نتیجه‌ای است که الگوریتم در طول اجراهای متعدد به دست آورده است. بهترین پاسخ معمولاً کیفیت بالای راه‌حل ارائه‌شده توسط الگوریتم را نشان می‌دهد.

۲- انحراف معیار[2]: انحراف معیار میزان پراکندگی پاسخ‌های به‌دست‌آمده توسط الگوریتم را نسبت به میانگین پاسخ‌ها اندازه‌گیری می‌کند. انحراف معیار پایین نشان‌دهنده پایداری و دقت الگوریتم است. محاسبه آن با استفاده از روابط زیر است:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad (۲۴)$$

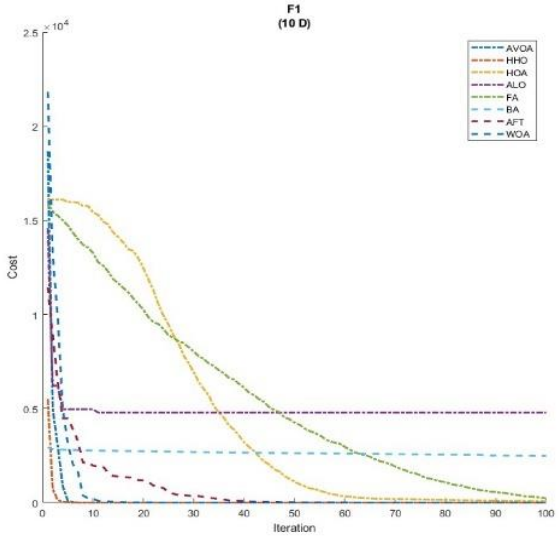
$$SD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \qquad (۲۵)$$

بطوری که μ میانگین پاسخ های بدست آمده در N تکرار الگوریتم و x_i بهترین پاسخ بدست آمده در تکرار iام است.

۳- زمان اجرا[3]: این معیار نشان دهنده مدت زمانی است که الگوریتم برای یافتن یک پاسخ صرف می‌کند و نشان‌دهنده سرعت الگوریتم در حل مسئله است.

۴-۱- نتایج پژوهش

در این پژوهش باتوجه‌به پارامترهای اولیه در نظر گرفته‌شده برای هر الگوریتم در جدول (۱) و با استفاده از مجموعه توابع تست، الگوریتم‌های موردنظر شبیه‌سازی‌شده‌اند. در شکل‌های (۱) تا (۲۳) منحنی همگرایی الگوریتم‌ها برای توابع *F1* تا *F23* در بُعد ۱۰و تکرار۱۰۰ نشان داده شده است. همچنین، نتایج به‌دست‌آمده برای معیارهای بهترین پاسخ، انحراف معیار و زمان اجرا در جداول (۳)، (۴)، (۵) و (۶) آمده است.



شکل (۹): منحنی همگرایی الگوریتم‌ها برای حل تابع *F1* در بعد ۱۰ و با تعداد تکرار ۱۰۰

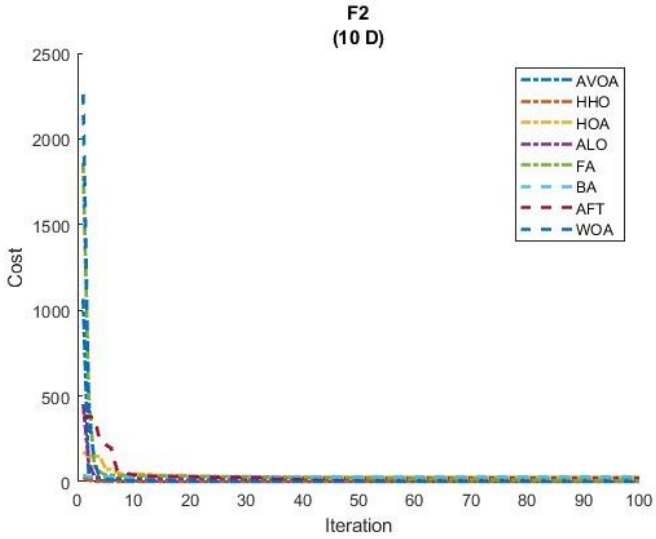
Figure (9): Convergence curve of algorithms in the function F1 and dimension 10

شکل (۹) نمودار همگرایی تابع *F1* را نشان می‌دهد. شکل نشان می دهد که الگوریتم شاهین هریس سریعترین همگرایی را داشته است. علاوه بر این، الگوریتم های شیر مورچه و خفاش به جواب بهینه همگرا نشده اند و الگوریتم علی بابا و چهل دزد با سرعت مناسبی همگرا شده است.

[1] Best Solution

[2] Standard Deviation (SD)

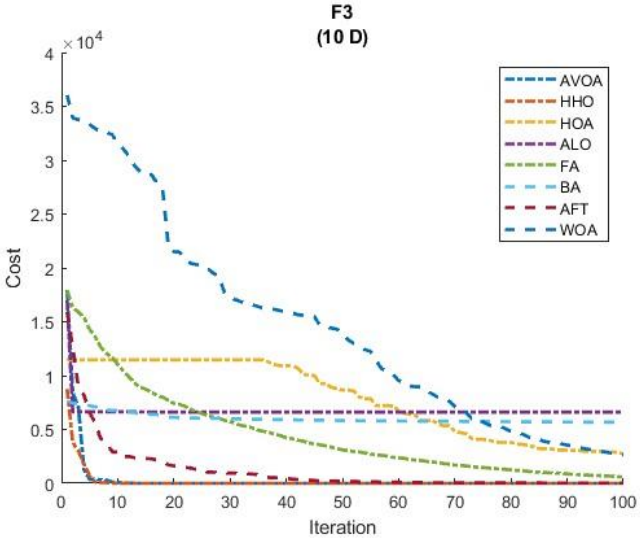
[3] Runtime



شکل (۱۰): منحنی همگرایی الگوریتم‌ها برای حل تابع *F2* در بعد ۱۰ و با تعداد تکرار ۱۰۰

Figure (10): Convergence curve of algorithms in the function F2 and dimension 10

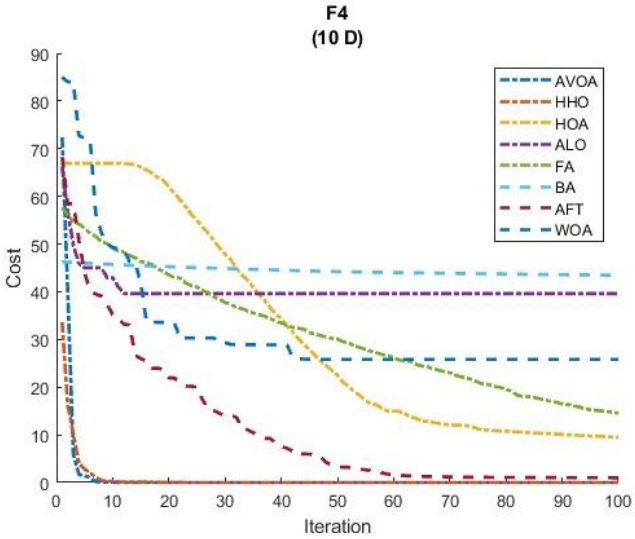
در نمودار همگرایی تابع *F2* (شکل ۱۰)، مشاهده می‌شود که اکثر الگوریتم‌ها سرعت همگرایی مناسبی داشته‌اند که نشان‌دهنده قابلیت بالای این الگوریتم‌ها در این تابع است.



شکل (۱۱): منحنی همگرایی الگوریتم‌ها برای حل تابع *F3* در بعد ۱۰ و با تعداد تکرار ۱۰۰

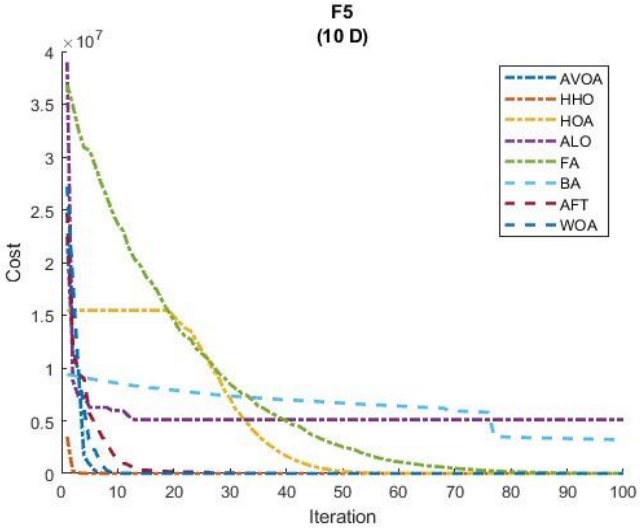
Figure (11): Convergence curve of algorithms in the function F3 and dimension 10

شکل (۱۱) نشان می دهد که الگوریتم های شاهین هریس و کرکس آفریقایی بیشترین سرعت همگرایی را داشته اند. در رتبه بعدی الگوریتم علی بابا و چهل دزد قرار دارد. بدترین عملکرد را الگوریتم شیر مورچه و نهنگ داشته اند.



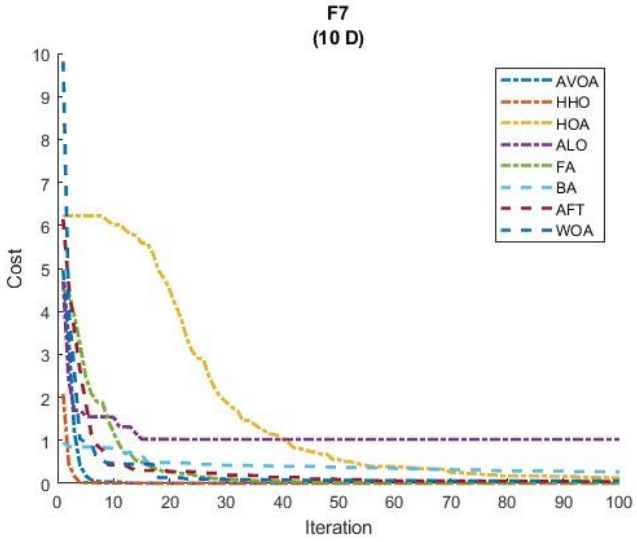
شکل (۱۲): منحنی همگرایی الگوریتم‌ها برای حل تابع *F4* در بعد ۱۰ و با تعداد تکرار ۱۰۰
Figure (12): Convergence curve of algorithms in the function F4 and dimension 10

نمودار فوق نشان می‌دهد که همگرایی الگوریتم‌های کرکس آفریقایی و شاهین هریس از سایر الگوریتم‌ها سریع‌تر بوده است. در اینجا الگوریتم علی‌بابا و چهل دزد عملکرد قابل‌قبولی داشته است. یکی از بدترین عملکردها به الگوریتم نهنگ اختصاص می‌یابد.



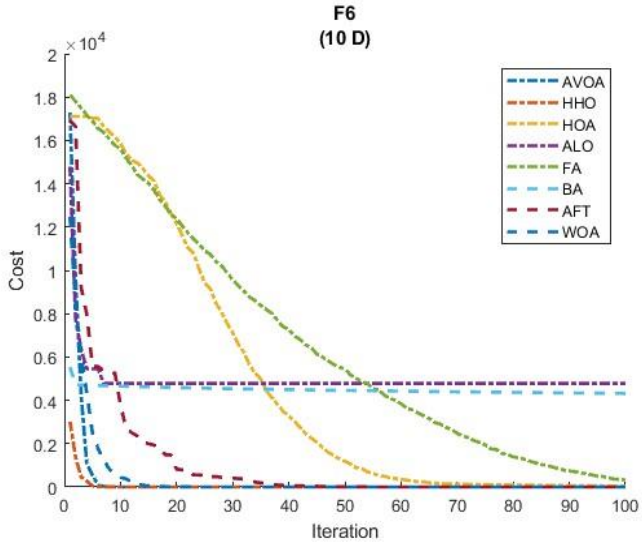
شکل (۱۳): منحنی همگرایی الگوریتم‌ها برای حل تابع *F5* در بعد ۱۰ و با تعداد تکرار ۱۰۰
Figure (13): Convergence curve of algorithms in the function F5 and dimension 10

نمودار همگرایی تابع *F5* نشان می‌دهد که الگوریتم‌های خفاش و شیر مورچه به همگرایی نرسیده‌اند، درحالی‌که کرکس آفریقایی و شاهین هریس بهترین عملکرد را داشته‌اند. در این نمودار عملکرد همگرایی الگوریتم علی‌بابا و چهل دزد در رتبه چهارم قرار می‌گیرد.



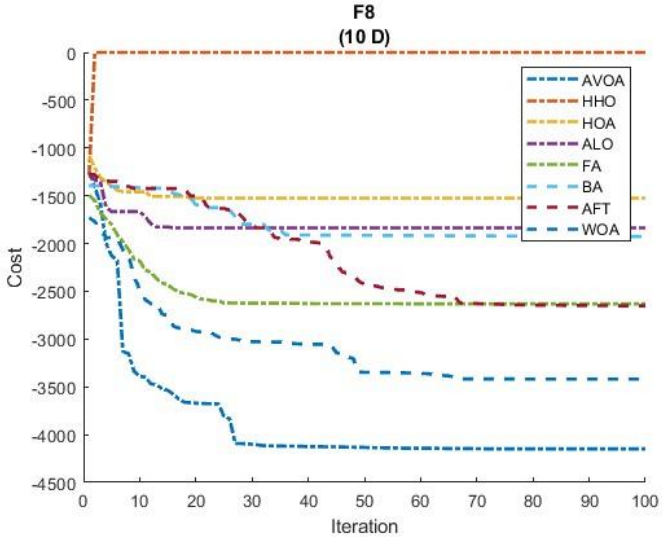
شکل (۱۴): منحنی همگرایی الگوریتم‌ها برای حل تابع *F6* در بعد ۱۰ و با تعداد تکرار ۱۰۰
Figure (14): Convergence curve of algorithms in the function F6 and dimension 10

شکل (۱۴) نشان می دهد که اگر چه الگوریتم های گله اسب و کرم شب تاب در ابتدا سرعت همگرایی بالایی نداشته اند ولی در نهایت به جواب بهینه همگرا شده اند. این شکل نشان می دهد که الگوریتم علی بابا و چهل دزد از سه الگوریتم برتر دیگر عملکرد ضعیفتری داشته است.



شکل (۱۵): منحنی همگرایی الگوریتم‌ها برای حل تابع *F7* در بعد ۱۰ و با تعداد تکرار ۱۰۰
Figure (15): Convergence curve of algorithms in the function F7 and dimension 10

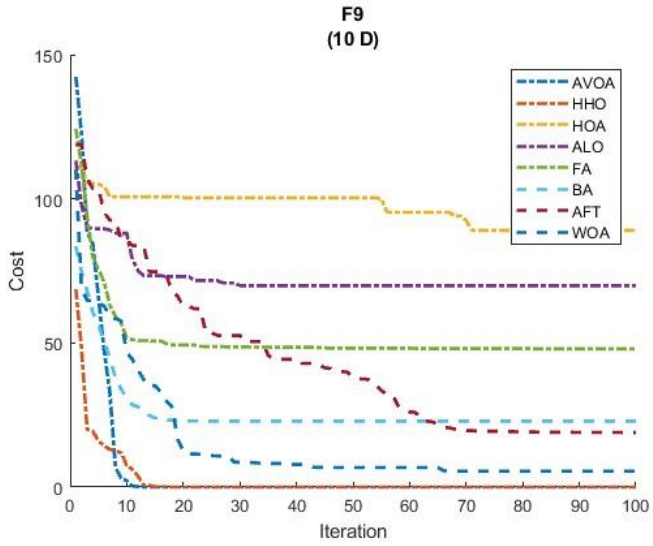
شکل (۱۵) نشان می‌دهد که بجز الگوریتم شیر مورچه سایر الگوریتم ها به پاسخ بهینه همگرا شده و یا در تکرار های بیشتر در نهایت همگرا خواهند شد. در این نمودار مشخص است که سرعت همگرایی الگوریتم های شاهین هریس، کرکس آفریقایی و نهنگ از الگوریتم علی بابا و چهل دزد بیشتر بوده است.



شکل (۱۶): منحنی همگرایی الگوریتم‌ها برای حل تابع *F8* در بعد ۱۰ و با تعداد تکرار ۱۰۰

Figure (16): Convergence curve of algorithms in the function F8 and dimension 10

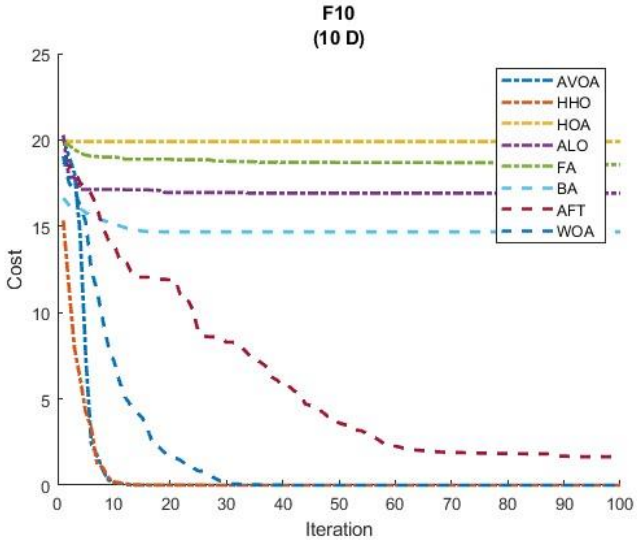
در نمودار شکل (۱۶) مشخص است که بجز الگوریتم شاهین هریس، هیچکدام از الگوریتم ها به پاسخ بهینه همگرا نشده اند. الگوریتم های کرکس آفریقایی نسبت به سایر الگوریتم‌ها بدترین عملکرد را داشته است.



شکل (۱۷): منحنی همگرایی الگوریتم‌ها برای حل تابع *F9* در بعد ۱۰ و با تعداد تکرار ۱۰۰

Figure (17): Convergence curve of algorithms in the function F9 and dimension 10

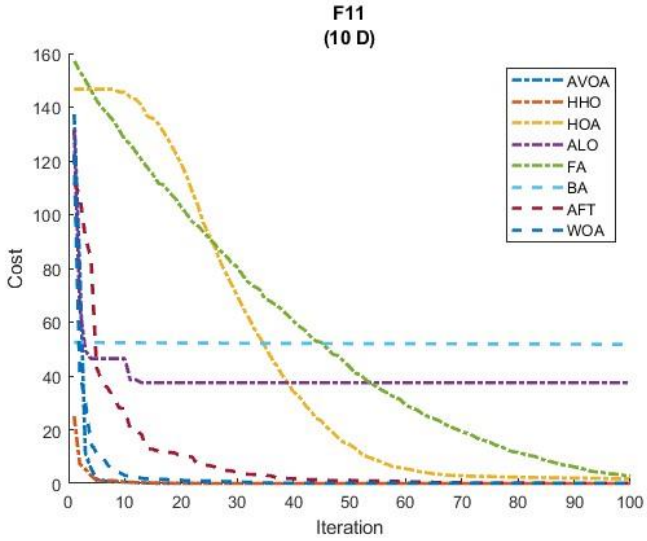
همگرایی الگوریتم‌های کرکس آفریقایی و شاهین هریس برای حل تابع *F9* در شکل (۱۷) کاملاً مشهود است. در اینجا الگوریتم علی بابا و چهل دزد عملکرد مناسبی نداشته و الگوریتم گله اسب بدترین عملکرد را داشته است.



شکل (۱۸): منحنی همگرایی الگوریتم‌ها برای حل تابع *F10* در بعد ۱۰ و با تعداد تکرار ۱۰۰

Figure (18): Convergence curve of algorithms in the function F10 and dimension 10

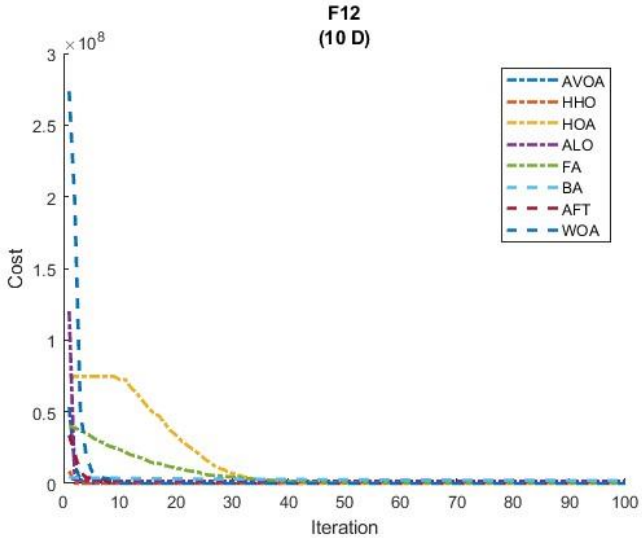
در نمودار همگرایی مربوط به تابع *F10* در *شکل (۱۸) مشاهده می شود که*، الگوریتم علی بابا و چهل دزد در ابتدا سرعت همگرایی مناسبی داشته است ولی در نهایت نتوانسته است به پاسخ بهینه همگرا شود. الگوریتم های خفاش، شیرمورچه، کرم شب تاب و گله است هم نتوانسته اند همگرا شوند.



شکل (۱۹): منحنی همگرایی الگوریتم‌ها برای حل تابع *F11* در بعد ۱۰ و با تعداد تکرار ۱۰۰

Figure (19): Convergence curve of algorithms in the function F11 and dimension 10

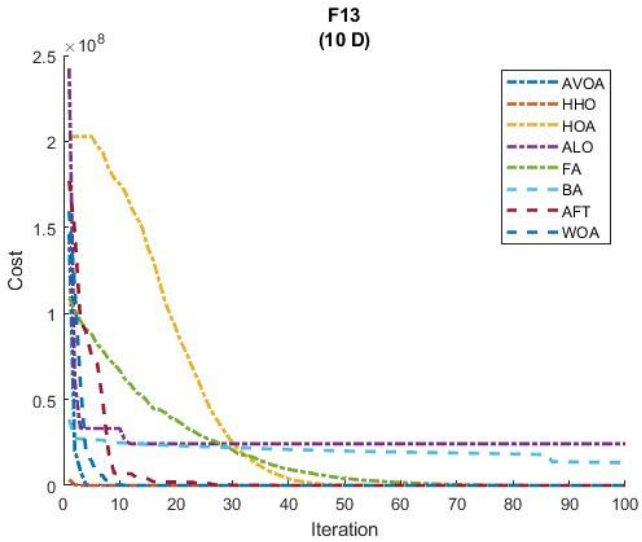
نمودار همگرایی برای تابع *F11*، شکل (۱۹)، نشان می دهد که به ترتیب الگوریتم های شاهین هریس، کرکس آفریقایی و نهنگ عملکرد بهتری نسبت به سایر الگوریتم‌ها داشته اند. همچنین، الگوریتم‌های خفاش و شیر مورچه بدترین عملکرد را داشته اند.



شکل (۲۰): منحنی همگرایی الگوریتم‌ها برای حل تابع *F12* در بعد ۱۰ و با تعداد تکرار ۱۰۰

Figure (20): Convergence curve of algorithms in the function F12 and dimension 10

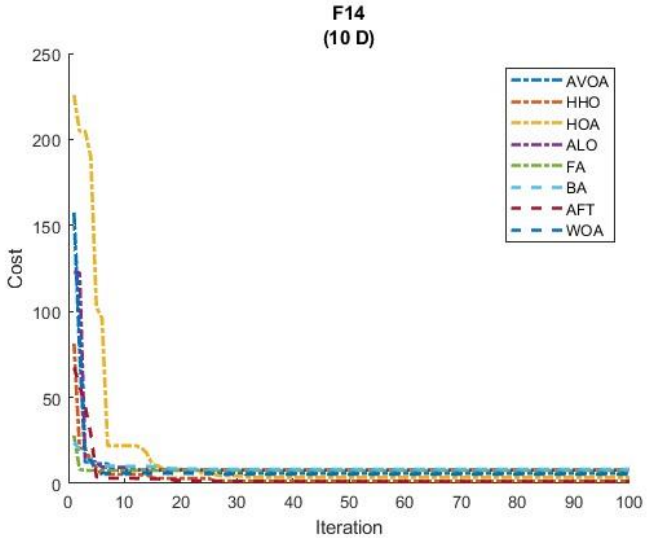
در شکل (۲۰) مشاهده می شود که تمامی الگوریتم ها با سرعت مناسبی به پاسخ بهینه همگرا شده اند.



شکل (۲۱): منحنی همگرایی الگوریتم‌ها برای حل تابع *F13*در بعد ۱۰ و با تعداد تکرار ۱۰۰

Figure (21): Convergence curve of algorithms in the function F13 and dimension 10

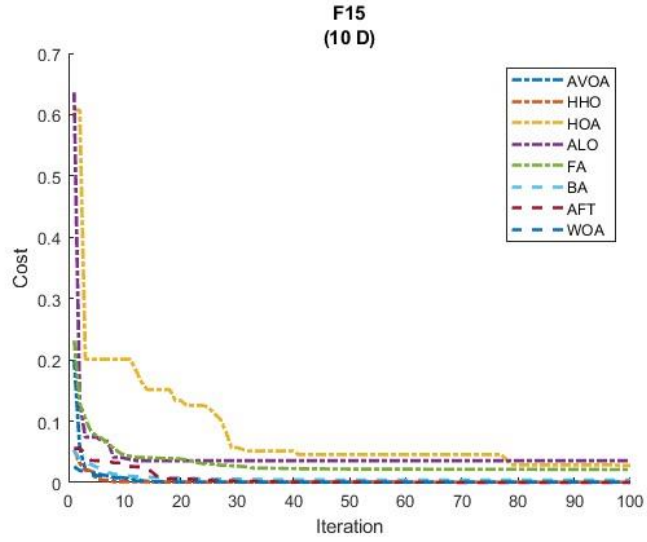
نمودار همگرایی تابع *F13* نشان می‌دهد که به‌جز الگوریتم‌های خفاش و شیر مورچه سایر الگوریتم‌ها به پاسخ بهینه همگرا شده‌اند. الگوریتم علی‌بابا و چهل دزد نسبت به الگوریتم‌های شاهین هریس، کرکس آفریقایی و نهنگ دیرتر همگرا شده است.



شکل (۲۲): منحنی همگرایی الگوریتم‌ها برای حل تابع *F14* در بعد ۱۰ و با تعداد تکرار ۱۰۰

Figure (22): Convergence curve of algorithms in the function F14 and dimension 10

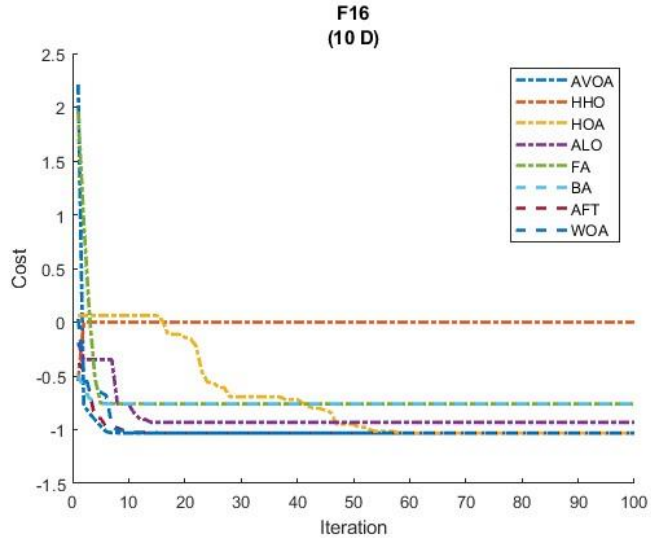
در شکل (۲۲) مشخص است که الگوریتم علی بابا و چهل دزد زودتر از سایر الگوریتم‌ها همگرا شده و عملکرد بهتری داشته است ولی سایر الگوریتم ها نیز عملکرد قابل قبولی داشته اند.



شکل (۲۳): منحنی همگرایی الگوریتم‌ها برای حل تابع *F15*در بعد ۱۰ و با تعداد تکرار ۱۰۰

Figure (23): Convergence curve of algorithms in the function F15 and dimension 10

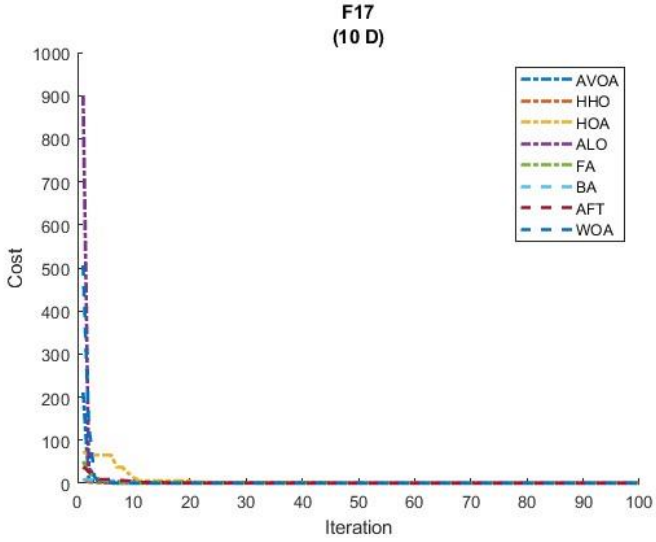
نمودار همگرایی مربوط به حل تابع *F15* نشان می‌دهد که سرعت همگرایی الگوریتم‌های علی‌بابا و چهل دزد، شاهین هریس، کرکس آفریقایی و نهنگ تقریباً مشابه بوده است؛ اما سایر الگوریتم‌ها نتوانسته‌اند به جواب بهینه همگرا شوند.



شکل (۲۴): منحنی همگرایی الگوریتم‌ها برای حل تابع *F16* در بعد ۱۰ و با تعداد تکرار ۱۰۰

Figure (24): Convergence curve of algorithms in the function F16 and dimension 10

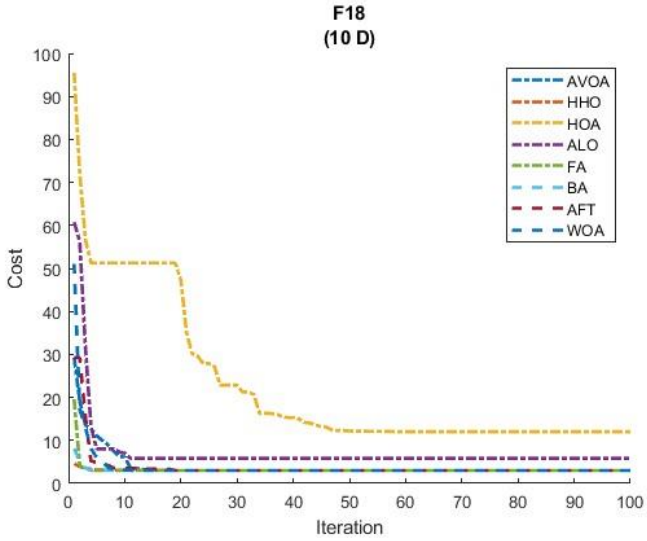
برای حل تابع *F16* به‌جز الگوریتم شاهین هریس هیچ‌کدام از الگوریتم‌های بررسی شده نتوانسته‌اند به پاسخ بهینه همگرا شوند که در شکل (۲۴) مشخص است.



شکل (۲۵): منحنی همگرایی الگوریتم‌ها برای حل تابع *F17* در بعد ۱۰ و با تعداد تکرار ۱۰۰

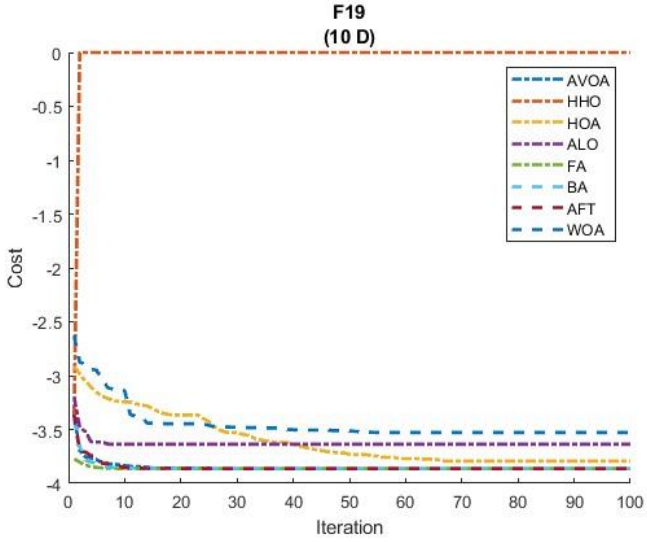
Figure (25): Convergence curve of algorithms in the function F17 and dimension 10

در شکل مربوط به همگرایی تابع *F17*، بیشتر الگوریتم‌ها عملکرد مشابهی در همگرایی داشته‌اند، اما الگوریتم چهل دزد و علی‌بابا به سطح بهتری از همگرایی رسیده است.



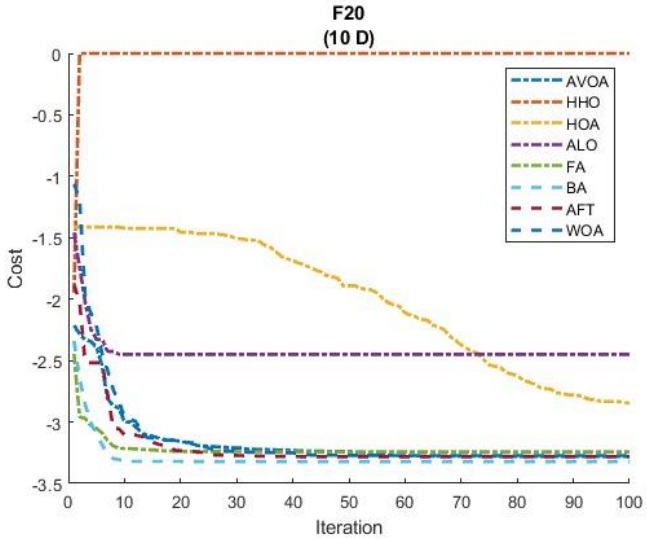
شکل (۲۶): منحنی همگرایی الگوریتم‌ها برای حل تابع *F18* در بعد ۱۰ و با تعداد تکرار ۱۰۰
Figure (26): Convergence curve of algorithms in the function F18 and dimension 10

شکل (۲۶) نشان می دهد که در ابتدا سرعت همگرایی اکثر الگوریتم ها مناسب بوده است ولی در نهایت هیچکدام به جواب بهینه همگرا نشده اند.



شکل (۲۷): منحنی همگرایی الگوریتم‌ها برای حل تابع *F19* در بعد ۱۰ و با تعداد تکرار ۱۰۰
Figure (27): Convergence curve of algorithms in the function F19 and dimension 10

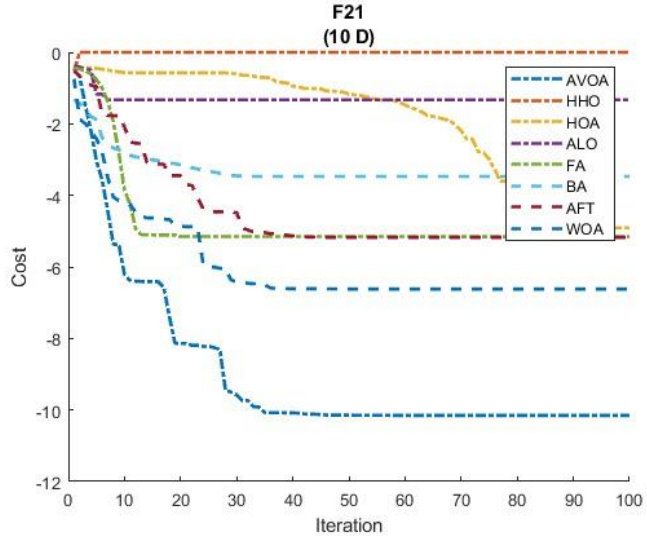
در یافتن پاسخ بهینهٔ مربوط به تابع F19 الگوریتم شاهین هریس بهترین عملکرد را داشته است. الگوریتم علی‌بابا و چهل دزد و سایر الگوریتم‌ها نتوانسته‌اند به پاسخ بهینه دست یابند.



شکل (۲۸): منحنی همگرایی الگوریتم‌ها برای حل تابع *F20*در بعد ۱۰ و با تعداد تکرار ۱۰۰

Figure (28): Convergence curve of algorithms in the function F20 and dimension 10

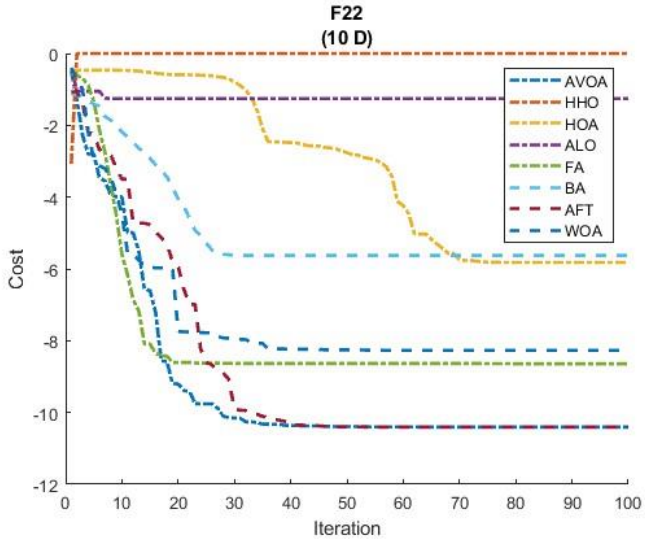
نمودار همگرایی مربوط به تابع *F20* در شکل (۲۸) نشان می‌دهد که الگوریتم شاهین هریس با سرعت مناسبی به پاسخ بهینه رسیده است ولی تمامی الگوریتم‌های دیگر نتوانسته اند به پاسخ بهینه همگرا شوند.



شکل (۲۹): منحنی همگرایی الگوریتم‌ها برای حل تابع *F21* در بعد ۱۰ و با تعداد تکرار ۱۰۰

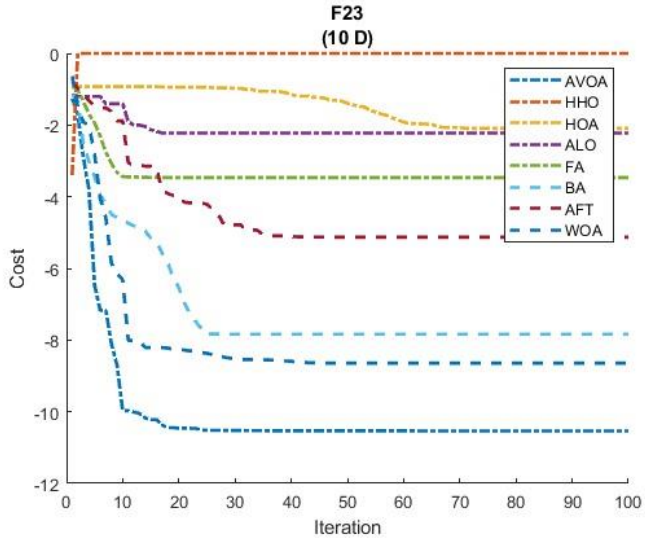
Figure (29): Convergence curve of algorithms in the function F21 and dimension 10

در حل تابع *F21*، به‌جز الگوریتم شاهین هریس سایر الگوریتم‌ها نتوانسته‌اند به پاسخ بهینه همگرا شوند. بدترین عملکرد را الگوریتم‌های نهنگ و کرکس آفریقایی داشته‌اند. الگوریتم علی‌بابا و چهل دزد نیز عملکرد مناسبی نداشته است.



شکل (۳۰): منحنی همگرایی الگوریتم‌ها برای حل تابع *F22* در بعد ۱۰ و با تعداد تکرار ۱۰۰
Figure (30): Convergence curve of algorithms in the function F22 and dimension 10

شکل (۳۰) نشان می دهد که فقط الگوریتم شاهین هریس به پاسخ بهینه همگرا شده است. الگوریتم علی بابا و چهل دزد یکی از الگوریتم هایی بوده است که بدترین عملکرد را داشته است.



شکل (۳۱): منحنی همگرایی الگوریتم‌ها برای حل تابع *F23* در بعد ۱۰ و با تعداد تکرار ۱۰۰
Figure (31): Convergence curve of algorithms in the function F23 and dimension 10

در این نمودار همگرایی مربوط به تابع *F23*، شکل (۳۱)، نتایج متفاوتی مشاهده می‌شود. عملکرد بسیار کارآمد الگوریتم شاهین هریس در مقایسه با سایر الگوریتم‌ها با منحنی همگرایی آن به‌وضوح نشان داده شده است. در اینجا بدترین عملکرد به الگوریتم کرکس آفریقایی و پس از آن به الگوریتم نهنگ تعلق دارد.

نمودارهای همگرایی توابع نشان می‌دهند که در توابع تک‌وجهی F1 تا F7 معمولاً الگوریتم‌های کرکس آفریقایی و شاهین هریس بهترین و الگوریتم‌های شیر مورچه و خفاش بدترین عملکرد را داشته‌اند. در این توابع سرعت همگرایی الگوریتم علی‌بابا و چهل دزد رتبه سوم یا چهارم را دارد. همچنین، در توابع چندوجهی F8 تا F13 نیز عملکرد الگوریتم‌هایی کرکس آفریقایی و شاهین هریس بهترین بوده است. در اینجا، در نیمی از موارد سایر الگوریتم‌ها همگرا نشده‌اند. عملکرد الگوریتم علی‌بابا و چهل دزد نیز مناسب نبوده است. بررسی نمودارهای همگرایی توابع ترکیبی F14 تا F23 نشان می‌دهد که در اکثر موارد عملکرد همگرایی الگوریتم شاهین هریس از سایر الگوریتم‌های بررسی شده بهتر بوده است. با مقایسه نتایج این توابع با توابع تک‌وجهی و چندوجهی

می‌توان نتیجه گرفت که الگوریتم کرکس آفریقایی برای توابع ترکیبی مناسب نبوده و عملکرد قابل‌قبولی نداشته است. الگوریتم علی‌بابا و چهل دزد فقط در ۳۳ درصد از توابع همگرا شده است و این نشان از عملکرد ضعیف این الگوریتم در حل توابع ترکیبی دارد. عملکرد ضعیف الگوریتم چهل دزد و علی‌بابا نسبت به الگوریتم‌های کرکس آفریقایی و شاهین هریس ممکن است به این دلیل باشد که الگوریتم چهل دزد و علی‌بابا در تعادل بین جستجو[1] و بهره‌برداری[2] به اندازه کافی قوی نباشد، و این می‌تواند منجر به گیر افتادن در بهینه‌های محلی یا کندی در همگرایی شود.

مهم‌ترین پارامترهای اولیهٔ الگوریتم‌های بررسی شده در جدول (۱) آمده است. تمامی الگوریتم‌ها در تکرارهای ۱۰۰ و ۱۰۰۰ و با ابعاد ۱۰، ۲۰ و ۳۰ برای توابع F1 تا F23 شبیه‌سازی شده اند. نتایج شبیه‌سازی برای معیارهای بهترین پاسخ، انحراف معیار و زمان اجرای الگوریتم‌ها در جداول (۳)، (۴)، (۵) و (۶) به صورت جداگانه ثبت و تجزیه و تحلیل شده اند.

جدول ۲: عملکرد الگوریتم ها درتکرار ۱۰۰ با بُعد۱۰ (توابع تک وجهی، چند وجهی)

Table 2: The performance of algorithms in repetition 100 and dimension 10 (monohedral, polyhedral functions)

	AFT		HOA		FA		ALO		WOA		BA		HHO		AVOA	
F	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**
F1	0.031929	0.036375	80.453	13.273	229.56	96.895	4789.6	251.93	1.1775e-24	1.238e-23	2479.9	1128.7	5.6455e-09	2.0839e-09	**1.4829e-59**	**2.5685e-59**
F2	0.19325	0.15544	25.104	12.957	4.5511	0.028656	19.473	0.27881	9.8199e-15	9.2454e-15	27.175	20.045	8.2395e-09	2.5417e-09	**2.2326e-27**	**3.8661e-27**
F3	28.285	17.464	2739.9	418.4	594.38	198.07	6608.9	858.24	2650.7	1442.8	5666.9	4240.8	9.3484e-09	7.363e-10	**2.2682e-40**	**3.9286e-40**
F4	1.005	0.73945	9.488	3.9449	14.502	2.8528	38.542	19.057	25.813	10.692	43.461	6.7465	6.0988e-09	2.526e-09	**4.9674e-31**	**8.5713e-31**
F5	394.2	614.89	3076.6	6304	15291	1366.2	2.6288e+06	2.9091e+06	8.1942	0.079619	3.0219e+06	3.9429e+06	0.1791	0.29123	**0.00037869**	**0.00035869**
F6	0.0080304	0.0042959	40.344	65.762	326.16	33.359	4864.6	936.96	0.1166	0.10068	4318.8	469.54	0.0011106	0.00040779	**1.5103e-05**	**1.6269e-05**
F7	0.0434	0.015906	0.13339	0.066198	0.0085447	0.010467	1.0243	0.73293	0.0592	0.0024252	0.26948	0.22908	0.00033848	0.00018062	**0.00022965**	**0.00012599**
F8	-2651.2	125.71	-1524.3	148.07	-2631.5	155.9	-1821.8	130.7	-3595	111.19	-1926.4	185.05	-1286.1	190.61	**-4150**	**68.663**
F9	18.949	4.5183	89.062	9.7035	47.932	27.432	69.885	17.515	5.5219	11.865	22.89	12.765	1.9933e-09	1.7743e-09	**0**	**0**
F10	1.6383	0.60694	19.908	0.20211	17.931	0.55658	16.58	0.80678	3.1162e-12	2.6177e-12	14.678	2.3728	6.3188e-09	1.2243e-09	**4.4409e-16**	**0**
F11	0.1321	0.33442	1.7256	0.017607	2.6658	0.833	34.222	5.8456	0.027448	0.0079645	51.751	17.901	4.6281e-09	3.2977e-09	**0**	**0**
F12	0.97925	0.70265	4.3278	0.2914	12.543	3.9091	4.669e+06	1.5892e+06	0.066894	0.024609	1.5448e+07	1.9904e+07	**0.00056407**	**0.00087688**	0.0067426	0.011676
F13	1.5088	2.4141	7.9603	3.5063	18.557	9.8394	1.3186e+07	1.569e+07	0.16521	0.090832	9.5054e+06	8.3157e+06	0.001435	0.0011482	**1.4513e-06**	**1.1257e-06**
Time	0.3213		0.2134		1.6086		0.5652		0.03876		**0.05217**		0.17824		0.038782	

عملکرد الگوریتم علی‌بابا و چهل دزد در مقایسه با سایر الگوریتم‌ها با استفاده از توابع استاندارد *F1* تا *F13* و با بُعد ۱۰ و تکرار ۱۰۰ مورد بررسی قرار گرفته است. نتایج این ارزیابی در جدول (۲) ارائه شده است که نشان می‌دهد الگوریتم کرکس آفریقایی در تمام توابع تک وجهی و در اکثر توابع چند وجهی توانسته بهترین پاسخ و کمترین انحراف معیار را ارائه دهد. در رتبه های بعدی معمولا الگوریتم های شاهین هریس و یا نهنگ قرار می گیرند. الگوریتم علی‌بابا و چهل دزد در اکثر موارد پس از الگوریتم‌های کرکس آفریقایی، شاهین هریس و نهنگ توانسته بهترین پاسخ را بدست آورد. ارزیابی ها نشان می دهد که برای یافتن بهترین پاسخ، عملکرد الگوریتم علی بابا و چهل دزد از عملکرد سه الگوریتم دیگر بدتر بوده و از سایر الگوریتم های مورد بررسی بهتر بوده است. عملکرد ضعیف‌تر الگوریتم چهل دزد و علی بابا در توابع تک‌وجهی و چندوجهی نسبت به کرکس آفریقایی،

[1] Exploration
[2] Exploitation

شاهین هریس و نهنگ احتمالا به دلیل بهره‌برداری ضعیف‌تر در فاز نهایی جستجو، حساسیت به تنظیمات اولیه، و عدم تنوع کافی در جمعیت است. در مقابل، این الگوریتم‌ها به دلیل طراحی پیچیده‌تر و تطبیق‌پذیری بالاتر، عملکرد بهتری در این نوع توابع دارند. بدتربن عملکرد از این جهت به الگوریتم های شیرمورچه و خفاش اختصاص دارد. کمترین زمان اجرا را به ترتیب الگوریتم های نهنگ، کرکس آفریقایی و خفاش دارا می باشند. از این جهت الگوریتم علی بابا و چهل دزد رتبه ششم را در بین هشت الگوریتم بررسی شده دارد که عملکرد مناسبی نمی باشد. با توجه به موارد بیان شده می نوان نتیجه گرفت که عملکرد الگوریتم علی بابا و چهل دزد از نظر بهترین پاسخ پیدا شده و نیز زمان اجرا در میان الگوریتم های بررسی شده مناسب نمی باشد. بهترین عملکرد را الگوریتم کرکس آفریقایی دارد.

جدول ۳: عملکرد الگوریتم ها درتکرار ۱۰۰۰ با بُعد۱۰ (توابع تک وجهی، چند وجهی)

Table 3: The performance of algorithms in repetition 1000 and dimension 10 (monohedral, polyhedral functions)

	AFT		HOA		FA		ALO		WOA		BA		HHO		AVOA	
F	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**
F1	2.5183e-41	3.3646e-41	2.0212e-30	2.4332e-30	0.78624	0.10796	5122.2	1470.4	1.0846e-268	0	2514	974.78	2.6016e-09	2.1225e-09	**0**	**0**
F2	1.7557e-18	2.4739e-18	27.621	8.0375	0.21926	0.04541	18.236	3.9354	6.5763e-146	5.8723e-146	26.927	29.568	4.3977e-09	4.1608e-09	**1.7139e-320**	**0**
F3	3.6008e-18	2.54e-19	466.18	36.321	0.60065	0.054665	5213.5	827.97	0.0010724	0.0011161	5474.6	1080.1	4.1902e-09	3.6102e-09	**0**	**0**
F4	6.4627e-08	9.1376e-08	6.0606e-12	8.5709e-12	0.46213	0.0054992	41.994	5.4758	1.7471	2.4468	25.481	14.802	5.3274e-09	2.0319e-09	**4.9407e-324**	**0**
F5	2.0011	2.8079	8.9772	4.8091	248.29	329.79	1.3541e+06	1.5457e+06	6.0964	0.095928	3.2327e+05	1.3691e+05	0.00082669	0.001078	**3.0822e-06**	**3.7719e-06**
F6	**0**	**0**	2.2533	0.0027788	0.77823	0.023009	3584.1	531.72	5.9489e-05	9.4825e-06	7584	485.86	3.7306e-06	2.937e-06	2.2746e-14	3.2449e-14
F7	0.0055242	0.00071677	0.057691	0.062594	5.19e-05	1.2255e-05	0.56532	0.20238	0.0028697	0.0039801	0.045428	0.033509	8.1232e-05	8.5386e-05	**8.8398e-05**	**5.6793e-05**
F8	-3341	307.07	-2105.4	780.98	-2577.8	377.89	-1997.3	265.53	-3655.6	230.45	-1911.4	461.85	-1521.2	257.77	**-4051.6**	**239.34**
F9	12.934	5.6283	62.416	7.1029	56.129	4.2733	56.742	7.1583	0	0	29.353	9.1459	6.0462e-09	5.236e-09	**0**	**0**
F10	7.5895e-15	0	19.661	0.10977	19.193	0.0011088	15.561	0.55043	5.7732e-15	2.5121e-15	13.544	0.81076	4.7512e-09	1.6493e-09	**4.4409e-16**	**0**
F11	0.072563	0.053861	1.7573	0.37887	0.52306	0.065161	33.239	1.0068	0	0	38.403	20.706	5.3939e-09	3.1928e-09	**0**	**0**
F12	**4.7116e-32**	**0**	4.7647	3.9736	15.556	2.241	6.7055e+05	3.0648e+05	0.00028729	0.00020188	1.1809e+06	5.908e+05	4.5542e-06	4.7694e-06	1.0342e-14	7.6651e-15
F13	**1.3498e-32**	**0**	8.5953	6.1354	0.045814	0.0011874	7.8154e+06	6.6755e+06	0.00037404	0.00035949	4.1392e+05	5.4784e+05	4.153e-05	6.0309e-05	1.7475e-12	7.4407e-13
Time	0.8793		1.7007		25.3328		14.3370		**0.3666**		0.4913		0.9823		0.5213	

جدول (۳) عملکرد الگوریتم چهل دزد و علی‌بابا با سایر الگوریتم‌ها با استفاده از توابع *F1* تا *F13* و با بُعد ۱۰ و تکرار ۱۰۰۰ در سه معیار بهترین پاسخ، انحراف معیار و میانگین زمان اجرا نشان می دهد. در حل توابع تک وجهی، بهترین عملکرد الگوریتم چهل دزد و علی‌بابا در معیار بهترین پاسخ مربوط به تابع F6 بوده و پس از آن تابع F3 است. همچنین، بجز تابع F6 در سایر توابع الگوریتم کرکس آفریقایی نسبت به الگوریتم چهل دزد و علی‌بابا مقدار بهینه تری را بدست آورده و الگوریتم شاهین هریس نیز در غالب موارد برتر بوده است. در مقایسه بین الگوریتم های نهنگ و شاهین هریس می توان گفت که این دو نسبت به هم برتری معنا داری ندارند. بدترین نتایج در اکثر موارد مربوط به الگوریتم شیرمورچه و خفاش می باشد. در اینجا مشخص است که نسبت به نتایج موجود در جدول (۲)، الگوریتم چهل دزد و علی‌بابا در یافتن بهترین پاسخ، موفق تر عمل کرده و جواب های مورد قبول تری را بدست آورده است. بهترین عملکرد الگوریتم چهل دزد و علی‌بابا برای یافتن بهترین پاسخ در حل توابع چند وجهی متعلق به حل توابع F12 و F13 است بطوری که از تمام الگوریتم ها عملکرد بهتری داشته است. همچنین، بدترین عملکرد را در حل تابع F8 داشته است. برتری الگوریتم کرکس آفریقایی و شاهین هریس در برابر الگوریتم چهل دزد و علی‌بابا و در حل

توابع چند وجهی نسبت به توابع تک وجهی کمتر بوده است و این نشان می دهد که الگوریتم چهل دزد و علی‌بابا عملکرد بهتری در حل مسائل با پیچیدگی بیشتر داشته است. مقدار انحراف معیار برای الگوریتم چهل دزد و علی‌بابا و کرکس آفریقایی نسبتاً پایین است، که نشان‌دهنده پایداری و یکنواختی در عملکرد این دو الگوریتم می‌باشد.

جدول ۴: عملکرد الگوریتم ها درتکرار ۱۰۰ با بُعد۲۰ (توابع تک وجهی، چند وجهی)

Table 4: The performance of algorithms in repetition 100 and dimension 20 (monohedral, polyhedral functions)

	AFT		HOA		FA		ALO		WOA		BA		HHO		AVOA	
F	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**
F1	33.728	3.7598	679.01	410.47	9756.6	92.216	14203	7533.1	2.0828e-22	2.5647e-22	11989	262.02	7.4899e-09	1.9604e-09	**4.883e-64**	**8.4575e-64**
F2	5.238	3.3489	23969	31895	83.672	9.86	3096.6	4065.2	3.0696e-13	1.9675e-13	58.369	5.7998	6.4646e-09	3.2454e-09	**3.7791e-25**	**6.5451e-25**
F3	1859.1	1187	2576.4	1132.43	10469	3029.9	26083	16789	20129	2193	37165	28031	3.7498e-09	4.8856e-09	**5.0685e-39**	**8.7789e-39**
F4	19.59	8.8068	13.012	4.6837	37.578	7.5379	52.381	11.407	69.422	27.487	46.627	9.4885	4.3158e-09	2.8769e-09	**2.9243e-28**	**5.0649e-28**
F5	956.16	705.32	40643	7799.6	1.1023e+06	4.9402e+05	2.2326e+07	7.1048e+06	18.563	0.059716	4.3394e+06	3.5076e+05	**0.22775**	**0.32083**	5.8137	10.065
F6	16.639	17.746	1001.1	221.96	6840.1	600.84	17574	878.68	0.36012	0.15211	10637	847.09	**0.001483**	**0.00095222**	0.083781	0.072762
F7	0.079684	0.03303	0.14578	0.076517	0.086523	0.0041014	9.3582	0.54317	0.012994	0.0036901	0.54378	0.048349	0.0019551	0.001386	**9.5307e-05**	**3.2385e-05**
F8	-4825.9	186	-2436.2	141.68	-5294.2	655.38	-3208.9	1887.15	-7187.9	140.76	-3186.3	1280	-2450.2	123.91	**-7161.1**	**103.1**
F9	73.044	32.189	209.78	13.33	109.74	16.477	194.13	2.9513	0.63533	0.8985	46.288	0.71017	4.8437e-09	1.5191e-09	**0**	**0**
F10	4.2498	0.78657	20.103	0.48313	19.872	0.27959	18.468	0.18611	2.6916e-11	3.5401e-11	14.926	0.84732	5.7634e-09	1.5793e-09	**4.4409e-16**	**0**
F11	1.5086	0.13427	8.3745	1.3215	62.806	14.707	139.28	48.476	7.2164e-16	8.6355e-16	120.31	1.3399	2.7613e-09	2.3536e-09	**0**	**0**
F12	10.626	1.9472	8.8708	3.676	1.7372e+05	1.4034e+05	1.7672e+07	4.0065e+06	0.086189	0.021086	1.255e+07	1.2116e+07	0.00035651	0.00037825	**6.1871e-05**	**7.1163e-05**
F13	39.004	11.323	179.39	221.49	1.6765e+06	1.935e+06	1.2803e+08	5.3276e+07	0.36163	0.029869	7.942e+06	13098	**0.0019454**	**0.0050688**	0.0090698	0.015705
Time	1.1538		0.2153		1.7693		1.2307		0.0764		**0.0622**		1.2391		0.0871	

نتایج موجود در جدول (۴) نشان می دهد که در حل توابع تک وجهی، الگوریتم کرکس آفریقایی در توابع F1 تا F4 و F7، الگوریتم شاهین هریس در توابع F5 و F6 بهترین پاسخ ها را بدست آورده اند. در اینجا نیز الگوریتم چهل دزد و علی‌بابا عملکرد قابل قبولی نداشته و در اکثر موارد برای یافتن پاسخ بهینه در رتبه چهارم قرار می گیرد. همچنین، بدترین پاسخ های بهینه در اکثر توابع توسط الگوریتم های شیرمورچه و خفاش بدست آمده است. در حل توابع چند وجهی، الگوریتم های کرکس آفریقایی، شاهین هریس و نهنگ به ترتیب بهترین پاسخ ها را در اکثر توابع بدست آورده اند. الگوریتم علی بابا و چهل دزد در رتبه بعدی قرار می گیرد. همانطور که از نتایج مشخص است عملکرد الگوریتم علی بابا و چهل دزد در یافتن بهترین پاسخ با افزایش ابعاد مسئله بدتر شده است. در معیار زمان اجرا نیز این الگوریتم یکی از بالاترین زمان ها را دارد.

جدول ۵: عملکرد الگوریتم ها درتکرار ۱۰۰۰ با بُعد۲۰ (توابع تک وجهی، چند وجهی)

Table 6: The performance of algorithms in repetition 1000 and dimension 20 (0monohedral, polyhedral functions)

	AFT		HOA		FA		ALO		WOA		BA		HHO		AVOA	
F	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**
F1	1.2041e-16	1.5934e-16	1.9394e-25	1.7092e-25	5.648	0.24684	15896	2435.6	3.1834e-254	0	9540.1	5326.2	6.3109e-09	4.0984e-09	**0**	**0**
F2	3.9789e-09	4.9577e-09	29352	41361	41.717	57.786	137.18	73.424	4.3325e-147	6.0712e-147	43.964	10.898	7.0264e-09	1.7857e-09	**7.1468e-318**	**0**
F3	0.0079357	0.0057854	4855	802.95	7.4542	1.3327	32801	11815	18.674	9.1394	22145	2795.3	3.9871e-09	1.2306e-09	**0**	**0**

F4	1.212	0.25327	5.6868e-14	7.8101e-14	1.0231	0.023936	61.038	1.9596	56.477	23.088	33.566	8.3755	8.1716e-09	2.4387e-09	**7.4427e-309**	**0**
F5	8.9855	0.012981	18.991	3.012981	114.78	70.731	2.2277e+07	3.9113e+06	16.999	0.02944	5.8151e+06	2.8018e+06	0.0027688	0.003352	**2.3998e-06**	**2.9622e-06**
F6	**4.6234e-18**	**4.6883e-18**	4.7492	0.0013543	5.2365	0.18595	12575	879.08	0.0043188	0.001972	8713.2	4607.8	0.00012704	0.00018692	2.6445e-11	1.0393e-11
F7	0.024464	0.0027115	0.024595	0.01142	0.00023794	3.9433e-05	13.026	6.2971	0.0025634	0.0013429	0.6084	0.48234	3.8436e-05	4.3968e-05	**5.4647e-05**	**1.5004e-05**
F8	-5496.8	29.058	-3245	94.16	-5171.7	151.97	-2807.8	239.92	-5844.7	21.47	-3616.5	1086.2	-2137.1	138.24	**-8379.7**	**1.0689e-09**
F9	71.139	13.367	186	33.504	116.74	24.311	196.09	5.4974	0	0	58.21	30.254	5.2059e-09	2.3833e-09	**0**	**0**
F10	1.2263	1.7342	20.323	0.13118	19.808	0.18589	17.79	0.66146	2.2204e-15	2.5121e-15	15.108	2.2407	4.5747e-09	1.5986e-09	**4.4409e-16**	**0**
F11	0.025786	0.015575	2.0968	0.60166	1.0378	0.006422	157.44	18.831	0.025909	0.035934	84.949	22.711	5.4938e-09	4.6148e-09	**0**	**0**
F12	0.77803	1.1003	6.6928	0.74522	12.221	1.6029	1.5384e+07	7.0539e+06	0.014878	0.0091009	5.9505e+05	6.9568e+05	2.8996e-07	1.2244e-07	**2.8149e-11**	**2.2612e-11**
F13	**5.9508e-17**	**6.407e-17**	19.184	6.4063	0.27232	0.016953	9.9376e+07	1.0062e+08	0.13464	0.019681	5.6348e+06	4.5584e+06	8.7456e-06	9.4594e-06	2.9835e-10	1.7039e-10
Time	1.1428		1.7142		27.2857		34.2857		**0.4857**		0.5362		1.54209		0.5391	

عملکرد الگوریتم چهل دزد و علی‌بابا با سایر الگوریتم‌ها با استفاده از توابع *F1* تا *F13* و با بُعد ۲۰ و تکرار ۱۰۰۰ در سه معیار بهترین پاسخ، انحراف معیار و میانگین زمان اجرا در جدول (۵) نشان داده شده است. الگوریتم علی بابا و چهل دزد در یافتن مقدار بهینه در توابع F1، F2، F6 و F13 عملکرد قابل قبولی داشته است اما برای بقیه توابع نتوانسته است مقدار بهینه مناسبی را پیدا نماید. برخلاف این الگوریتم، الگوریتم کرکس آفریقایی بهترین عملکرد را داشته و توانسته است مقدار بهینه صفر را برای ۴ تابع تک وجهی و ۳ تابع چند وجهی بدست آورد. بهترین عملکرد الگوریتم چهل دزد و علی‌بابا در مقایسه با الگوریتم کرکس آفریقایی در یافتن جواب بهینه ی توابع F6 و F13 بوده است. در اینجا نیز در اکثر موارد الگوریتم شاهین هریس مقدار بهینه تری را از الگوریتم چهل دزد و علی بابا بدست آورده است. در نتایج تابع F8 مشاهده می شود که به طور غیر منتظره ای الگوریتم های خفاش و شیرمورچه عملکرد بهتری از الگوریتم های الگوریتم کرکس آفریقایی و علی بابا و چهل دزد داشته اند در حالی که در سایر توابع از جمله بدترین الگوریتم ها بوده اند. مقایسه نتایج جدول (۴) و (۵) نشان می دهد که با افزایش تعداد تکرار، الگوریتم علی بابا و چهل دزد پاسخ های بهینه تری را بدست آورده است.

جدول ۶: عملکرد الگوریتم ها درتکرار ۱۰۰ با بُعد۳۰ (توابع تک وجهی، چند وجهی)

Table 5: The performance of algorithms in repetition 100 and dimension 30 (monohedral, polyhedral functions)

	AFT		**HOA**		**FA**		**ALO**		**WOA**		**BA**		**HHO**		**AVOA**	
F	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**
F1	431.78	245.16	844.06	51.753	26587	2223.3	27654	4642.2	6.285e-22	8.7708e-22	17471	1205.4	5.5548e-09	3.8624e-09	**7.7257e-59**	**1.3381e-58**
F2	16.673	2.8511	1.4028e+09	1.9498e+09	124.37	43.844	2.802e+06	3.9525e+06	3.5119e-14	6.776e-15	1294.3	1775.1	5.9983e-09	3.235e-09	**3.284e-24**	**5.688e-24**
F3	3874.7	972.24	26234	11230	24604	1121.2	63152	19544	1.4511e+05	13718	76597	5924.8	1.5973e-09	1.3171e-09	**4.6379e-39**	**8.0329e-39**
F4	27.416	9.6117	13.518	2.0722	53.391	5.9528	64.215	4.0686	62.847	38.081	41.291	3.7042	5.6078e-09	3.432e-10	**1.1524e-29**	**1.7938e-29**
F5	90854	25923	82101	60145	2.2226e+07	1.0311e+07	8.5513e+07	1.1187e+06	28.784	0.0051305	2.1693e+07	3.3262e+06	**0.033636**	**0.022209**	9.3713	16.19
F6	616.13	174.71	1599.1	193.68	21270	2840.8	28520	4762.3	1.222	0.0091081	11931	344.12	**0.0026635**	**0.0012735**	0.0083307	0.0028494
F7	0.27079	0.23805	0.28124	0.043154	0.19761	0.067458	22.404	2.7267	0.021494	0.0042587	3.0397	1.367	0.00074961	0.00040023	**0.00042517**	**9.4811e-05**
F8	-6800	857.32	-2985.4	150.73	-6598.7	1012.6	-4348.3	842.89	-8261.2	141.92	-3445.1	249.47	-3103	1129.1	**-11666**	**101.3**
F9	117.28	8.9636	350.66	24.93	208.85	24.626	345.6	12.939	1.1369e-13	0	57.253	23.222	6.1369e-09	3.5386e-10	**0**	**0**

F10	8.5513	2.397	20.488	0.030639	20.04	0.20944	18.94	0.6169	3.2014e-12	2.7282e-12	15.808	1.2852	4.4425e-09	3.5749e-09	**4.4409e-16**	**0**
F11	5.2034	1.6555	9.1101	2.417	164.53	0.70602	300.8	20.855	1.1102e-16	0	165.9	45.662	2.0154e-09	2.7294e-09	**0**	**0**
F12	18.321	10.981	14.284	5.0685	6.2883e+06	4.7396e+06	1.2308e+08	9.9264e+06	0.069364	0.022995	2.3755e+06	1.3887e+06	0.00058703	0.00045554	**0.00056828**	**0.00093293**
F13	124.2	81.408	54.935	2.5664	7.0149e+07	3.8109e+07	2.1207e+08	5.814e+07	1.4268	0.49754	1.1697e+08	9.0965e+07	0.00152	0.00079341	**0.00026218**	**0.00023704**
Time	1.0769		0.2316		1.4094		1.5384		0.5384		0.3846		0.1243		**0.044278**	

عملکرد الگوریتم چهل دزد و علی‌بابا با سایر الگوریتم‌ها با استفاده از توابع F1 تا F13 و با بُعد ۳۰ و تکرار ۱۰۰ در سه معیار بهترین پاسخ، انحراف معیار و میانگین زمان اجرا در جدول (۶) نشان داده شده است. نتایج نشان می دهد که الگوریتم چهل دزد و علی‌بابا در توابع تک‌وجهی در اکثر موارد نتوانسته است به مقدار بهینه (صفر) نزدیک شود. برای مثال، در توابع F3، F5 و F6 با مقدار بهینه فاصله زیادی دارد. سایر الگوریتم‌ها مانند کرکس آفریقایی و شاهین هریس در تمام توابع تک وجهی نسبت به الگوریتم چهل دزد و علی‌بابا برتری داشته اند. این مسئله در اکثر موارد نسبت به الگوریتم نهنگ هم صادق است. بنابر این الگوریتم چهل دزد و علی‌بابا در اکثر موارد در رتبه چهارم بهترین جواب بدست آمده قرار می گیرد. بدتربن عملکرد در این زمینه به الگوریتم های شیر مورچه، خفاش و کرم شب تاب تعلق دارد. در توابع تک‌وجهی الگوریتم های کرکس آفریقایی و شاهین هریس انحراف معیار بسیار پایینی دارند، که نشان‌دهنده پایداری بالای این الگوریتم ها است. در توابع چند وجهی، با افزایش پیچیدگی، عملکرد الگوریتم چهل دزد و علی‌بابا در یافتن بهترین جواب از رقبا بدتر شده است. در اینجا نیز معمولا الگوریتم‌های کرکس آفریقایی، شاهین هریس و نهنگ عملکرد بهتری را نسبت به سایر الگوریتم ها داشته اند. بهترین عملکرد در یافتن جواب بهینه به الگوریتم کرکس آفریقایی و در توابع F9 و F11 تعلق دارد.

جدول ۷: عملکرد الگوریتم ها درتکرار ۱۰۰۰ با بُعد۳۰ (توابع تک وجهی، چند وجهی)

Table 7: The performance of algorithms in repetition 1000 and dimension 30 (monohedral, polyhedral functions)

	AFT		**HOA**		**FA**		**ALO**		**WOA**		**BA**		**HHO**		**AVOA**	
F	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**
F1	9.4834e-10	1.4021e-10	2.2118e-17	3.128e-17	14.527	0.36609	30630	2440.7	1.0953e-250	0	17226	6175.9	1.1397e-09	1.1523e-09	**0**	**0**
F2	0.0074918	0.0003182	80502	36048	77.906	107.31	1.8714e+07	2.3403e+07	2.5026e-140	3.395e-140	91.98	25.13	5.6318e-09	3.3224e-09	**1.4761e-304**	**0**
F3	16.291	13.992	9961.5	1948.4	27.247	6.2741	39966	1355.5	882.76	1246.2	34390	14195	7.7914e-10	4.6796e-10	**0**	**0**
F4	14.963	1.212	3.7972e-13	3.1902e-13	1.4398	0.16319	58.306	0.9614	46.964	9.5844	54.941	9.0864	3.9002e-09	1.6018e-09	**1.0449e-307**	**0**
F5	28.324	1.5338	29.003	0.018953	278.71	163.46	5.709e+07	5.5919e+06	26.771	0.12556	2.9746e+06	2.3561e+05	0.0041129	0.0062492	**7.4748e-06**	**9.9143e-06**
F6	1.3499e-08	1.6015e-08	7.2721	0.3223	13.181	0.86051	30380	2717.9	0.019073	0.0030483	24674	11136	0.00013601	0.00012315	**3.9319e-09**	**2.2007e-09**
F7	0.069513	0.014751	0.058005	0.018878	0.00062787	0.00013624	31.077	11.709	0.0012298	0.0005651	1.1147	0.64565	2.6733e-05	2.2145e-05	**3.5795e-05**	**4.2843e-05**
F8	-8481.7	975.02	-3986.9	281.41	-6371.6	167.183	-3934.8	316.28	-12345	113.17	-3800.4	951.75	-5067.5	606.44	**-12451**	**206.02**
F9	75.617	8.4425	298.78	34.279	178.95	22.469	343.21	25.015	0	0	179.1	94.272	5.0509e-09	4.582e-09	**0**	**0**
F10	3.47	1.6308	20.516	0.065815	20.012	0.040834	19.118	0.15142	5.7732e-15	2.5121e-15	15.738	0.9845	5.49e-09	2.6742e-09	**4.4409e-16**	**0**
F11	0.014761	0.010416	2.1357	1.3012	1.1391	0.0046677	326.75	49.112	0	0	204.47	34.983	1.1529e-09	1.0138e-09	**0**	**0**
F12	3.8012	5.3757	14.711	7.9848	16.046	1.7069	1.0987e+08	7.7235e+07	0.0057582	0.0054699	1.275e+07	1.2367e+07	1.1535e-06	6.3328e-07	**2.5709e-10**	**1.5061e-10**
F13	0.44769	0.61759	9.0951	6.1373	0.62454	0.057198	2.4908e+08	2.6088e+07	0.3128	0.28414	3.7987e+07	2.7736e+07	2.8194e-05	1.4814e-05	**6.2571e-10**	**5.926e-10**
Time	1.2137		2.1863		26.168		50.884		0.6547		0.5926		1.2918		**0.5291**	

جدول (۷) عملکرد الگوریتم ها در حل توابع *F1* تا *F13*، با بُعد ۳۰ و تکرار ۱۰۰۰ را در سه معیار بهترین پاسخ، انحراف معیار و میانگین زمان اجرا نشان می دهد. در مقایسه با سایر الگوریتم ها، بهترین عملکرد الگوریتم علی بابا و چهل دزد در یافتن بهترین جواب مربوط به حل تابع F6 می باشد که پس از الگورتیم کرکس آفریقایی در رتبه دوم قرار می گیرد. همچنین، بدترین عملکرد آن در یافتن پاسخ بهینه ی توابع F7 و F8 می باشد بطوری که سومین بدتربن پاسخ بهینه را بدست آورده است. بجز تابع F8، در سایر توابع الگوریتم کرکس آفریقایی در میان الگوریتم های بررسی شده بهترین پاسخ ها را بدست آورده و در رتبه بعدی معمولا الگوریتم شاهین هریس و یا نهنگ قرار دارند. الگوریتم علی بابا و چهل دزد در قیاس با این الگوریتم ها پاسخ بهینه ای بدست نیاورده است. بدترین پاسخ های بهینه را معمولا الگوریتم های شیرمورچه و خفاش و در رتبه بعدی الگوریتم گله اسب پیدا کرده است. با وجود این، تابع F8 به صورت یک استثنا عمل کرده و بهترین پاسخ این تابع توسط الگوریتم های خفاش و شیرمورچه بدست آمده است. در معیار میانگین زمان اجرا، الگوریتم کرکس آفریقایی و خفاش دارای کمترین میانگین زمان اجرا می‌باشند. مقایسه نتایج جدول (۶) و (۷) نشان می دهد که با افزایش تعداد تکرار، الگوریتم علی بابا و چهل دزد پاسخ های بهینه تری را بدست آورده است.

جدول ۸: عملکرد الگوریتم های مورد بررسی با استفاده از توابع *F14* تا *F23* و تکرار ۱۰۰

Table 8: The performance of algorithms in repetition 100 (F14 - F23 functions)

	AFT		HOA		FA		ALO		WOA		BA		HHO		AVOA	
	Best	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**	**Best**	**STD**
F14	**0.598**	**0**	3.4882	0.020789	7.5301	0.70009	7.814	2.3347	5.2458	7.3586e-07	7.3504	6.1777	1.3293	0.5739	5.8967	**4.8827**
F15	**0.0001023 4**	**0**	0.027716	0.005889	0.021012	0.014171	0.03561	0.0065656	0.0005465	4.9776e-05	0.0038084	0.013142	0.00051654	0.00016839	0.00048364	**4.3605e-05**
F16	**-1.5316**	**1.5701e-16**	-1.0275	0.00012117	-0.75957	3.2191e-08	-0.93245	0.00037752	-1.3316	1.0774e-06	-0.75957	8.0095e-08	-0.553	0.3361	-1.0316	**1.1489e-14**
F17	**0.2872**	**0**	0.92228	0.19588	0.39813	6.6346e-05	0.94757	0.057038	0.4251	1.5905	0.47321	4.7517e-12	0.6341	0.0002203	0.39789	**3.0806e-10**
F18	**3**	**6.2804e-16**	3.0017	0.0024105	3	19.092	5.4009	3.3953	3.0002	0.00017039	3	1.6807e-05	3.0001	0.00011566	3	**2.1279e-05**
F19	**-3.8628**	**9.9301e-16**	-3.8542	0.0027817	-3.8628	7.8985e-07	-3.7044	0.027559	-3.4737	0.5432	-3.8628	2.5571e-06	-3.3665	0.15404	-3.8627	**0.00015341**
F20	**-3.3361**	**0**	-2.6771	0.64145	-3.203	4.8168e-05	-2.9327	0.090862	-3.0468	0.35309	-3.3219	6.8299e-05	-1.9474	0.63049	-3.2811	**0.068004**
F21	-5.173	4.313	-2.2681	0.90114	-5.0983	0.0013679	-1.3268	0.019616	-7.4172	3.3572	-6.3918	5.3193	-0.52706	0.047612	**-10.153**	**7.1687e-07**
F22	**-10.973**	**1.4891e-13**	-5.8191	4.6029	-8.6414	5.3957	-1.373	0.088507	-6.5181	5.4025	-6.5844	5.4001	-3.0753	2.0431	-10.403	**1.1308e-07**
F23	-5.1266	4.685	-2.0933	6.1542	-3.4675	3.7826	-2.2221	0.45655	-8.643	0.14944	-7.8333	3.6075e-06	-3.3962	2.4502	**-10.536**	**1.3179e-06**
	0.3213		0.2134		1.6086		0.5652		**0.03876**		0.05217		0.17824		**0.038782**	

توابع ترکیبی در بهینه‌سازی به دسته‌ای از مسائل گفته می‌شود که از ترکیب چندین تابع ساده‌تر تشکیل شده‌اند. این توابع معمولاً شامل ویژگی‌هایی مانند عدم تحدب، وجود چندین نقطه بهینه محلی و تعامل پیچیده بین متغیرها هستند. هدف از استفاده این توابع در مقایسهٔ الگوریتم‌های بهینه‌سازی، بررسی توانایی آن‌ها در حل مسائل واقعی و پیچیده‌تر است. عملکرد الگوریتم‌ها برای حل توابع ترکیبی *F14* تا *F23* با تکرار ۱۰۰ در سه معیار بهترین پاسخ، انحراف معیار و میانگین زمان اجرا در جدول (۸) نشان داده شده است. نتایج نشان می دهد که الگوریتم علی‌بابا و چهل دزد در توابع *F14*، *F15*، *F17* نسبت به سایر الگوریتم ها توانسته است پاسخ بهینه تری را بدست آورد. اما، در توابع *F16*، *F20*، *F22* بدترین پاسخ های بهینه متعلق به این الگوریتم است. نتایج معیار بهترین پاسخ برای تابع *F18* نشان می دهد که الگوریتم‌های علی‌بابا و چهل دزد، کرم شب تاب، خفاش و کرکس آفریقایی مقدار یکسانی را بدست آورده اند. همچنین، در تابع F19 نتایج بسیار به هم نزدیک هستند. برخلاف نتایج توابع تک وجهی و چند وجهی، در اینجا بهترین عملکرد را معمولا الگوریتم های شاهین هریس و علی‌بابا و چهل دزد داشته اند و عملکرد الگوریتم کرکس آفریقایی علارغم اینکه کمترین زمان اجرا را دارا می باشد، قابل قبول نبوده است. الگوریتم‌های موفق در توابع ترکیبی باید توازن

قوی بین جستجوی سراسری و جستجوی محلی داشته باشند. الگوریتم کرکس آفریقایی بیشتر به بهره‌برداری محلی متکی است و توانایی جستجوی سراسری آن ممکن است در فضاهای پیچیده و ناهمگن ترکیبی کافی نباشد که این می تواند دلیل عدم عملکرد مناسب این الگوریتم در یافتن پاسخ بهینه در توابع ترکیبی باشد. عملکرد الگوریتم شیرمورچه در این توابع به مراتب بهتر از نتایج مرتبط با توابع تک وجهی و چند وجهی بوده است. مقدار انحراف معیار برای الگوریتم چهل دزد و علی‌بابا در چهار تابع برابر صفر بوده و در بقیه توابع نیز نسبتاً پایین است، که نشان‌دهنده پایداری و یکنواختی در عملکرد این الگوریتم می‌باشد.

جدول ۹: عملکرد الگوریتم های مورد بررسی با استفاده از توابع *F14* تا *F23* و تکرار ۱۰۰۰

Table 9: The performance of algorithms in repetition 1000 (F14 - F23 functions)

	AFT		HOA		FA		ALO		WOA		BA		HHO		AVOA	
	Best	STD	Best	STD	Best	STD	Best	STD	Best	STD	Best	STD	Best	STD	Best	STD
F14	**0.698**	**0**	2.0231	0.043007	2.4831	2.1003	2.0346	0.050568	0.788	1.8371e-07	16.872	5.9412	0.918	3.6054e-11	0.998	**1.3597e-16**
F15	**0.00030749**	**5.421e-20**	0.0052264	0.0016914	0.01098	0.01327	0.011039	0.0062676	0.0010402	0.00064066	0.00078498	0.00049747	0.00032969	2.7866e-05	0.00057634	**1.3593e-05**
F16	**-1.0726**	**2.2204e-16**	-1.0313	0.00035224	-1.0116	1.0537e-07	-0.98927	0.0086728	-1.0316	2.8435e-11	-1.0316	2.416e-08	-0.79997	0.21357	-1.0316	**1.5701e-16**
F17	**0.39789**	**0**	0.41362	0.0044792	0.3979	2.2477e-06	1.138	1.0463	0.40137	0.0049218	0.90001	0.7101	0.34789	8.5382e-06	0.49789	**0**
F18	**3**	**9.9301e-16**	3.0631	0.089226	3	1.5323e-07	3.1545	0.083989	3	5.7314e-05	43.5	57.276	3	9.4946e-08	3	**1.3273e-07**
F19	**-3.8628**	**0**	-3.8574	0.0036373	-3.8628	3.2989e-07	-3.7915	0.024945	-3.8618	0.001332	-3.8628	4.1688e-07	-2.9754	0.58296	-3.8628	**1.9743e-14**
F20	**-3.322**	**0**	-2.9354	0.12735	-3.2031	1.9071e-06	-2.8293	0.013911	-3.2489	0.10085	-3.2625	0.084007	-0.94256	0.23864	-3.1924	**0.068643**
F21	-6.418	5.2823	-2.699	3.0532	-5.0774	0.032179	-2.024	0.82211	-10.151	0.0020494	-2.6567	0.037044	-1.8254	1.2583	**-10.153**	**1.7764e-15**
F22	-7.0636	4.7225	-6.084	2.2833	-6.5841	5.3999	-2.7946	0.48072	-7.7451	3.7583	-7.7659	3.7294	-1.1185	0.32486	**-10.403**	**1.7764e-15**
F23	-3.9998	1.5962	-1.554	1.0678	-10.536	0.00062546	-3.1802	0.023872	-10.536	0.00011982	-1.6766	5.8231e-07	-0.99267	0.13178	**-10.536**	**2.8087e-15**
	1.0769		0.2316		1.4094		1.5384		0.5384		0.3846		0.1243		**0.044278**	

جدول (۹) عملکرد الگوریتم ها برای حل توابع ترکیبی *F14* تا *F23* با تکرار ۱۰۰۰ در سه معیار بهترین پاسخ، انحراف معیار و میانگین زمان اجرا را نشان می دهد. نتایج نشان می دهد که در توابع *F16* و *F20* بدترین پاسخ های بهینه متعلق به الگوریتم علی‌بابا و چهل دزد است. این الگوریتم در توابع *F14* و *F15* نسبت به سایر الگوریتم ها توانسته است پاسخ بهینه تری را بدست آورد. نتایج این معیار برای تابع *F18* نشان می دهد که بجز الگوریتم‌های خفاش، شیر مورچه و گله اسب سایر الگوریتم ها مقدار بهینه ی یکسانی را بدست آورده اند. همچنین، در تابع *F19* نتایج بدست آمده برای اکثر الگوریتم ها بسیار به هم نزدیک است. نتایج معیار بهترین پاسخ نشان می دهد که الگوریتم شاهین هریس توانسته است در ۸ تابع (از مجموع ۱۰ تابع) بهترین جواب ها را بدست آورد و این مسئله نشان می دهد که برخلاف نتایج توابع تک وجهی و چند وجهی، در اینجا بهترین عملکرد را معمولا الگوریتم شاهین هریس داشته است و عملکرد الگوریتم کرکس آفریقایی علارغم اینکه یکی از کمترین زمان های اجرا را دارا می باشد، قابل قبول نبوده است. عملکرد الگوریتم شیرمورچه در این توابع به مراتب بهتر از نتایج مرتبط با توابع تک وجهی و چند وجهی بوده است. الگوریتم شیر مورچه از یک ساختار گروهی الهام گرفته از رفتار طبیعی شیر مورچه‌ها استفاده می‌کند: در این الگوریتم گروه‌های شکارچی برای کشف نواحی جدید در فضای جستجو اقدام می‌کنند و سپس شکارچیان محاصره‌کننده روی نقاط جالب در فضای جستجو تمرکز می‌کنند. این ویژگی‌ها باعث می‌شوند که این الگوریتم در فضاهای پیچیده و ناهمگن، مانند توابع ترکیبی، عملکرد بهتری داشته باشد زیرا می‌تواند به سرعت بین نواحی مختلف حرکت کند و پاسخ‌های بهینه را پیدا کند. در مقایسه ی نتایج جداول (۸) و (۹) مشخص می شود که عملکرد الگوریتم ها در تکرار ۱۰۰ و ۱۰۰۰ تغییر محسوسی نداشته است و این می تواند ناشی از پیچیدگی توابع ترکیبی باشد.

۵-نتیجه گیری و پیشنهادهای آینده:

فراابتکاری به دلیل ویژگی‌های منحصربه‌فردی مانند تنوع جستجو و قابلیت بهره‌برداری مؤثر از راه‌حل‌های ممکن، توانایی حل طیف گسترده‌ای از مسائل بهینه‌سازی غیرخطی را دارند. این الگوریتم‌ها از استراتژی‌های جستجوی پویا و سازگاری در فرایند جستجو بهره می‌برند. درعین‌حال، فلسفه‌های متفاوت در طراحی این الگوریتم‌ها منجر به تفاوت در همگرایی، یافتن بهترین پاسخ و زمان محاسبه می‌شود. در این پژوهش، برای ارزیابی و مقایسه عملکرد الگوریتم چهل دزد و علی‌بابا که از داستان معروف " علی‌بابا و چهل دزد" الهام‌گرفته است، با ۷ الگوریتم فراابتکاری دیگر شامل الگوریتم های گله اسب، کرم شب‌تاب، شیرمورچه، نهنگ، خفاش، شاهین هریس و کرکس آفریقایی، از ۲۳ تابع تست استاندارد CEC2017 استفاده شده است. نتایج این پژوهش نشان می دهد که الگوریتم علی‌بابا و چهل دزد در مقایسه با الگوریتم‌های مقایسه شده عملکرد متوسطی دارد. این الگوریتم، در برخی مسائل بهینه‌سازی نتایج مناسبی ارائه می دهد. بااین‌حال، در حل توابع تک وجهی و چند وجهی عملکرد این الگوریتم در مقایسه با الگوریتم‌های کرکس آفریقایی و شاهین هریس و در اکثر مواقع الگوریتم نهنگ قابل قبول نبوده است. دلایل این مسئله می‌تواند این باشد که الگوریتم‌های کرکس آفریقایی و شاهین هریس اغلب دارای تعادل بهتری بین جستجوی سراسری و جستجوی محلی هستند؛ زیرا الگوریتم کرکس آفریقایی از یک استراتژی شکار مبتنی بر رفتار طبیعی کرکس‌ها بهره می‌برد که توانایی بالایی در حرکت بین مناطق مختلف فضای جستجو و یافتن پاسخ‌های خوب را دارد. همچنین، الگوریتم شاهین هریس از مکانیزم‌های حمله ناگهانی و فرار تدریجی بهره می‌برد که باعث می‌شود در فاز نهایی به نقاط بهینه‌تر برسد. اما، احتمالا الگوریتم چهل دزد و علی‌بابا در تعادل بین جستجو و بهره‌برداری به اندازه کافی قوی نبوده است. الگوریتم علی‌بابا و چهل دزد معمولا توانسته است عملکرد بهتری از الگوریتم‌های شیر مورچه، خفاش، کرم شب تاب و گله اسب داشته باشد. دلیل این عملکرد می‌تواند این مسئله باشد که الگوریتم چهل دزد و علی بابا از یک داستان یا سناریوی خاص بهره می‌برد که به ترکیب اصول جستجوی سراسری و موضعی کمک می‌کند. این الگوریتم ممکن است از رویکردهای خلاقانه‌تر یا سازگارتر با فضای پیچیده جستجوی توابع CEC2017 استفاده کرده باشد. الگوریتم‌هایی مانند شیر مورچه یا خفاش گاهی اوقات تعادل کافی بین جستجوی سراسری و موضعی را ندارند زیرا الگوریتم شیر مورچه ممکن است در فاز جستجوی سراسری بیش از حد تهاجمی عمل کرده و بهینه‌های محلی را از دست دهد. همچنین، الگوریتم خفاش به شدت به پارامترهای تنظیمات اولیه (مانند فرکانس و دامنه) وابسته است و ممکن است در مسائلی، کارایی لازم را نداشته باشد. الگوریتم‌هایی مانند گله اسب و کرم شب تاب بیشتر برای مسائل ساده‌تر یا مسائل با تعداد محدود بهینه‌های محلی طراحی شده‌اند و ممکن است با چالش‌های توابع CEC2017 مانند نویز و چند بهینگی به‌خوبی سازگار نشوند. در مسائل ساده‌تر (مانند توابع تک‌وجهی)، برخی الگوریتم‌ها ممکن است عملکرد خوبی نشان دهند، اما در مواجهه با توابع ترکیبی که ویژگی‌های پیچیده دارند، عملکرد واقعی آن‌ها مشخص می‌شود. بنابراین، توابع ترکیبی به‌عنوان یک معیار جامع‌تر برای ارزیابی الگوریتم‌ها استفاده شده است. در حل توابع ترکیبی، الگوریتم شاهین هریس توانسته پاسخ های بهینه تری را نسبت به کلیه الگوریتم های مورد بررسی بدست آورد. در اینجا، عملکرد الگوریتم کرکس آفریقایی ضعیف بوده است ولی الگوریتم علی‌بابا و چهل دزد و نیز الگوریتم شیر مورچه عملکرد بهتری داشته اند. الگوریتم شاهین هریس دارای مولفه‌های دینامیکی است که با پیشرفت زمان اجرا، رفتار خود را تغییر داده و بهتر با توابع پیچیده سازگار می‌شود. ولی الگوریتم چهل دزد و علی‌بابا شاید فاقد چنین تطبیق‌پذیری بوده یا به دلیل ساختار کمتر پیچیده خود نتواند در شرایط چالش‌برانگیز رقابت کند که این می‌تواند دلیل عملکرد بهتر الگوریتم شاهین هریس در توابع ترکیبی باشد. بااین‌حال، به‌طورکلی الگوریتم های شاهین هریس و کرکس آفریقایی در حل توابع بهینه‌سازی بررسی شده به دلیل کارایی بالا و زمان اجرای بهینه، بهترین عملکرد را از میان الگوریتم‌های مقایسه شده به دست آورده اند. این بررسی نشان می دهد که هر الگوریتم نقاط قوت و ضعف خاص خود را دارد و این تفاوت‌ها بسته به نوع مسئله و ویژگی‌های آن قابل‌توجه است. برخی الگوریتم‌ها در مسائل خاص با ویژگی‌های پیچیده و چندبعدی عملکرد بهتری دارند و برخی دیگر در مسائل ساده‌تر و بعد کم بهینه‌تر عمل می‌کنند. انتخاب بهترین الگوریتم بهینه‌سازی نیازمند توجه به خصوصیات مسئله، معیارهای عملکردی و محدودیت‌های زمانی و محاسباتی است. یافته‌های این تحقیق نشان دهنده اهمیت و ضرورت انجام مطالعات بیشتر و دقیق‌تر در زمینه مقایسه ی الگوریتم‌های فرا ابتکاری، به‌ویژه در مسائل واقعی و پیچیده می باشد. مطالعات مقایسه‌ای بیشتر می‌توانند به شناسایی نقاط ضعف و قوت الگوریتم‌های مختلف در حوزه‌های متنوع علمی و صنعتی کمک کنند. نتایج این پژوهش می‌تواند به‌عنوان الگوی مناسبی برای پژوهش‌های آینده در زمینه بهبود عملکرد الگوریتم‌های فراابتکاری مورداستفاده قرار گیرد. یکی از پیشنهادهای

آتی این پژوهش، توسعه الگوریتم چهل دزد و علی‌بابا برای حل مسائل بهینه‌سازی چندهدفه است. در مسائل چندهدفه، نیاز است که چندین هدف به‌صورت هم‌زمان بهینه شوند که این چالش نیازمند بهبود روش‌های جستجو و بهره‌برداری است. این امر می‌تواند قابلیت الگوریتم چهل دزد و علی‌بابا را در حل مسائل پیچیده‌تر افزایش دهد. علاوه بر این، ترکیب الگوریتم‌های فراابتکاری با روش‌های یادگیری ماشین نیز می‌تواند به بهبود عملکرد این الگوریتم‌ها کمک کند. با ترکیب این دو رویکرد، الگوریتم‌ها می‌توانند با استفاده از تجربیات گذشته و یادگیری از داده‌های پیشین، بهینه‌سازی‌های بهتری انجام دهند و نتایج کارآمدتری حاصل شود.

منابع

1. Zhang, Y., S. Wang, and G. Ji, *A comprehensive survey on particle swarm optimization algorithm and its applications.* Mathematical problems in engineering, 2015. **2015**(1): p. 931256.
2. Gamarra, C. and J.M. Guerrero, *Computational optimization techniques applied to microgrids planning: A review.* Renewable and Sustainable Energy Reviews, 2015. **48**: p. 413-424.
3. SS, V.C. and A. HS, *Nature inspired meta heuristic algorithms for optimization problems.* Computing, 2022. **104**(2): p. 251-269.
4. Odili, J.B., *The dawn of metaheuristic algorithms.* International Journal of Software Engineering and Computer Systems, 2018. **4**(2): p. 49-61.
5. Slowik, A. and H. Kwasnicka, *Nature inspired methods and their industry applications—Swarm intelligence algorithms.* IEEE Transactions on Industrial Informatics, 2017. **14**(3): p. 1004-1015.
6. Braik, M., M.H. Ryalat, and H. Al-Zoubi, *A novel meta-heuristic algorithm for solving numerical optimization problems: Ali Baba and the forty thieves.* Neural Computing and Applications, 2022. **34**(1): p. 409-455.
7. Abualigah, L., et al., *Meta-heuristic optimization algorithms for solving real-world mechanical engineering design problems: a comprehensive survey, applications, comparative analysis, and results.* Neural Computing and Applications, 2022: p. 1-30.
8. Mirjalili, S., *Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm.* Knowledge-based systems, 2015. **89**: p. 228-249.
9. Zolghadr-Asli, B., O. Bozorg-Haddad, and X. Chu, *Crow search algorithm (CSA).* Advanced optimization by nature-inspired algorithms, 2018: p. 143-149.
10. Mirjalili, S., S.M. Mirjalili, and A. Lewis, *Grey wolf optimizer.* Advances in engineering software, 2014. **69**: p. 46-61.
11. Thrift, P.R. *Fuzzy Logic Synthesis with Genetic Algorithms*. in *ICGA*. 1991.
12. Dorigo, M., M. Birattari, and T. Stutzle, *Ant colony optimization.* IEEE computational intelligence magazine, 2006. **1**(4): p. 28-39.
13. Kennedy, J. and R. Eberhart. *Particle swarm optimization*. in *Proceedings of ICNN'95-international conference on neural networks*. 1995. ieee.
14. Rashedi, E., H. Nezamabadi-Pour, and S. Saryazdi, *GSA: a gravitational search algorithm.* Information sciences, 2009. **179**(13): p. 2232-2248.
15. Mirjalili, S. and S. Mirjalili, *Genetic algorithm.* Evolutionary algorithms and neural networks: theory and applications, 2019: p. 43-55.
16. Price, K., R.M. Storn, and J.A. Lampinen, *Differential evolution: a practical approach to global optimization*. 2006: Springer Science & Business Media.
17. MiarNaeimi, F., G. Azizyan, and M. Rashki, *Horse herd optimization algorithm: A nature-inspired algorithm for high-dimensional optimization problems.* Knowledge-Based Systems, 2021. **213**: p. 106711.
18. Heidari, A.A., et al., *Harris hawks optimization: Algorithm and applications.* Future generation computer systems, 2019. **97**: p. 849-872.
19. Abdollahzadeh, B., F.S. Gharehchopogh, and S. Mirjalili, *African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems.* Computers & Industrial Engineering, 2021. **158**: p. 107408.
20. Mirjalili, S., *The ant lion optimizer.* Advances in engineering software, 2015. **83**: p. 80-98.
21. Bangyal, W.H., J. Ahmad, and H.T. Rauf, *Optimization of neural network using improved bat algorithm for data classification.* Journal of Medical Imaging and Health Informatics, 2019. **9**(4): p. 670-681.
22. Yang, X.-S. and X. He, *Firefly algorithm: recent advances and applications.* International journal of swarm intelligence, 2013. **1**(1): p. 36-50.

23. Mirjalili, S. and A. Lewis, *The whale optimization algorithm.* Advances in engineering software, 2016. **95**: p. 51-67.
24. Singh, A. and A. Kumar, *Applications of nature-inspired meta-heuristic algorithms: A survey.* International Journal of Advanced Intelligence Paradigms, 2021. **20**(3-4): p. 388-417.
25. Luan, J., et al., *A novel method to solve supplier selection problem: Hybrid algorithm of genetic algorithm and ant colony optimization.* Mathematics and Computers in Simulation, 2019. **156**: p. 294-309.
26. Rodríguez, N., et al., *Optimization algorithms combining (meta) heuristics and mathematical programming and its application in engineering.* 2018.
27. Mirjalili, S., et al., *Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems.* Advances in engineering software, 2017. **114**: p. 163-191.
28. Alabool, H.M., et al., *Harris hawks optimization: a comprehensive review of recent variants and applications.* Neural computing and applications, 2021. **33**: p. 8939-8980.
29. Kamboj, V.K., et al., *An intensify Harris Hawks optimizer for numerical and engineering optimization problems.* Applied Soft Computing, 2020. **89**: p. 106018.
30. Fan, J., Y. Li, and T. Wang, *An improved African vultures optimization algorithm based on tent chaotic mapping and time-varying mechanism.* Plos one, 2021. **16**(11): p. e0260725.
31. Liu, R., et al., *Improved African vulture optimization algorithm based on quasi-oppositional differential evolution operator.* IEEE Access, 2022. **10**: p. 95197-95218.
32. Awadallah, M.A., et al., *Binary Horse herd optimization algorithm with crossover operators for feature selection.* Computers in biology and medicine, 2022. **141**: p. 105152.
33. Hosseinalipour, A. and R. Ghanbarzadeh, *A novel approach for spam detection using horse herd optimization algorithm.* Neural Computing and Applications, 2022. **34**(15): p. 13091-13105.
34. Abualigah, L., et al., *Ant lion optimizer: a comprehensive survey of its variants and applications.* Archives of Computational Methods in Engineering, 2021. **28**: p. 1397-1416.
35. Mohamed, A.W., et al., *Evaluating the performance of meta-heuristic algorithms on CEC 2021 benchmark problems.* Neural Computing and Applications, 2023. **35**(2): p. 1493-1517.
36. Khodadadi, N., et al., *A comparison performance analysis of eight meta-heuristic algorithms for optimal design of truss structures with static constraints.* Decision Analytics Journal, 2023. **8**: p. 100266.
37. Gürgen, S., et al., *A comprehensive performance analysis of meta-heuristic optimization techniques for effective organic rankine cycle design.* Applied Thermal Engineering, 2022. **213**: p. 118687.
38. Khari, M., et al., *Performance analysis of six meta-heuristic algorithms over automated test suite generation for path coverage-based optimization.* Soft Computing, 2020. **24**(12): p. 9143-9160.