توسعه الگوریتم تکامل شوراهای شهر برای حل مسائل بهینه‌سازی چندهدفه

مهدیه غفار علیشاهی[۱]، دانشجوی کارشناسی ارشد، عین‌الله پیرا[۲*]، استادیار، علیرضا روحی[۳]، استادیار

[۱، ۲ و ۳] دانشکده فناوری اطلاعات و مهندسی کامپیوتر- دانشگاه شهید مدنی آذربایجان - تبریز-ایران-

{pira,rouhi}@azaruniv.ac.ir

چکیده: پیشرفت فناوری و ظهور مسائل بهینه‌سازی چندهدفه در شاخه‌های علوم مختلف باعث تحقیق و ارائه الگوریتم‌های فراابتکاری جدید برای حل چنین مسائلی شده‌اند. اگرچه این الگوریتم‌ها تا حدودی توانسته‌اند تقریب نسبتاً خوبی از جبهه بهینه پرِتو را پیدا کنند ولی هنوز بهینه‌سازی به‌طور کامل انجام نشده است. در این مقاله، برای افزایش میزان بهینگی جبهه پرِتو تولید شده، نسخه چندهدفه‌ای از الگوریتم تکامل شوراهای شهر (CCE) با نام الگوریتم تکامل شوراهای شهر چندهدفه (MOCCE) ارائه می‌شود. در الگوریتم ارائه شده، یک آرشیو با اندازه ثابت برای ذخیره و بازیابی راه‌حل‌های بهینه پرِتو در نظر گرفته می‌شود. از این آرشیو برای تعریف ساختار هرم‌گونه شوراهای شهرها و شبیه‌سازی تکامل آن در فضاهای جستجوی چندهدفه استفاده می‌شود. کارایی الگوریتم MOCCE روی ۱۸ تابع تست چندهدفه شناخته شده موسوم به UF و IMOP مورد ارزیابی قرار گرفته و با نتایج الگوریتم‌های بهینه‌سازی شیر مورچه چندهدفه (MOALO)، کپک مخاطی چندهدفه (MOSMA) و مرغ مگس‌خوار مصنوعی چندهدفه (MOAHA) مقایسه شده‌اند. مطابق با نتایج آزمون میانگین رتبه‌ی فریدمن، در همه توابع تست UF، الگوریتم MOCCE اولین رتبه را در بین الگوریتم‌های مقایسه شده از لحاظ معیارهای فاصله نسلی (GD)، فاصله نسلی معکوس (IGD) و بیشینه گستردگی (MS) کسب می‌کند. همچنین، این الگوریتم اولین رتبه را در همه توابع تست IMOP از لحاظ معیار GD و دومین رتبه را از لحاظ معیارهای IGD و MS به خود اختصاص می‌دهد.

واژه‌های کلیدی: الگوریتم‌های فراابتکاری، بهینه‌سازی، چندهدفه، تکامل شورای شهر، جبهه پرِتو

* نویسنده مسئول، pira@azaruniv.ac.ir

Development of City Councils Evolution Algorithm for Multi-objective Optimization Problems

Mahdiyeh Ghafar-Alishahi[1], M.Sc. Student, Einollah Pira[2*], Assistant Professor, Alireza Rouhi[3], Assistant Professor

[1,2,3] Faculty of Information Technology and Computer Engineering, Azarbaijan Shahid Madani University, Tabriz, Iran, {pira, rouhi}@azaruniv.ac.ir

Abstract: The advancement of technology and the emergence of multi-objective optimization problems in various scientific domains have led to the research and presentation of new meta-heuristic algorithms to solve such problems. Although these algorithms have been able to find a relatively good approximation of the optimal Pareto front, but a complete optimization has not been carried out yet. In this paper, to increase the optimality of the generated Pareto front, we present a multi-objective version of the city council evolution algorithm (CCE) called the multi-objective city council evolution algorithm (MOCCE). In the presented algorithm, an archive with a fixed size is considered for storing and retrieving optimal Pareto solutions. This archive is used to define the hierarchical structure of city councils and to simulate its evolution in multi-objective search spaces. The efficiency of MOCCE algorithm has been evaluated on 18 well-known multi-objective test functions known as UF and IMOP and with the results of multi-objective ant lion optimization (MOALO), multi-objective orthogonal mould algorithm (MOSMA) and multi-objective artificial hummingbird optimization algorithms (MOAHA) have been compared. According to the results of the Friedman's mean rank test, in all UF test functions, MOCCE ranks first among all compared algorithms in terms of generation distance (GD), inverse generation distance (IGD) and maximum spread (MS) criteria. Also, this algorithm takes the first rank in all IMOP test functions in terms of GD criterion and the second rank in terms of IGD and MS criteria.

Keywords: *Meta-heuristic algorithms, Optimization, Multi-objective, Evolution of city council, Pareto front.*

*** Corresponding author, pira@azaruniv.ac.ir**

۱. مقدمه

بهینه‌سازی، فرایندی است که در آن بهترین جواب (با توجه به مجموعه‌ای از معیارها) از میان مجموعه‌ای از جواب‌های ممکن برای یک مسأله‌ی خاص انتخاب می‌شود [۱]. هدف از حل مسائل بهینه‌سازی تک‌هدفه، یافتن بهترین جوابی است که شرایط مسأله از جمله بیشینه‌سازی یا کمینه‌سازی تابع هدف و حفظ محدودیت‌های از پیش تعریف شده را برآورده کند [۲]. در حالی که در مسائل بهینه‌سازی چندهدفه، هدف، کمینه‌سازی یا بیشینه‌سازی چندین تابع هدف با لحاظ محدودیت‌های متعدد است [۳]. در این گونه مسائل، باید مجموعه‌ای از جواب‌ها که اهداف مد نظر را در سطح قابل قبولی برآورده می‌سازند، به‌عنوان مجموعه جواب بهینه معرفی شوند. این مجموعه جواب بهینه که مجموعه‌ی پرِتو[1] نامیده می‌شود، شامل جواب‌هایی است که با همدیگر مصالحه[2] دارند [۴]. به‌عبارت دیگر، مجموعه‌ی پرِتو شامل همه‌ی جواب‌های مناسبی است که نسبت به همدیگر هیچ‌گونه برتری ندارند. دو روش پیشینی[3] و پسینی[4] برای حل این قبیل مسائل وجود دارند [۵]. در روش‌های پیشینی، مسأله‌ی بهینه‌سازی چندهدفه با چندین تابع هدف به یک مسأله‌ی بهینه‌سازی تک‌هدفه با یک تابع هدف تبدیل می‌شود طوری که، با توجه به اهمیت هر کدام از اهداف، وزنی (ضریبی) توسط طراح مشخص شده و با توجه به این ضرایب، توابع هدف با هم ترکیب می‌شوند [۶]. مجموع وزن‌دار[5] [۷]، بیشینه-کمینه وزن‌دار[6] [۷]، برنامه‌نویسی هدف[7] [۸] و لکسیکوگراف[8] [۹] از جمله‌ی روش‌های پیشینی به حساب می‌آیند. عدم وجود اطلاعات در مورد میزان اهمیت هر کدام از اهداف در بعضی از مسائل بهینه‌سازی از جمله‌ی محدودیت‌های روش‌های پیشینی محسوب می‌شود. علاوه بر این، با توجه به این که تنها یک راه‌حل به‌عنوان جواب تولید می‌شود طراح هیچ بینشی نسبت به مصالحه‌ی بین اهداف ندارد. لذا، می‌توان این موضوع را به‌عنوان محدودیت دیگری برای این روش‌ها ذکر کرد. برخی از روش‌ها از جمله تجمیع وزنی پویا[9] (DWA) [۱۰] سعی در بهبود اِشکالات روش‌های پیشینی دارند. این الگوریتم تدریجاً وزن‌ها را تغییر داده و برنامه را برای هر مجموعه اجرا می‌کند تا نیاز قبلی به اطلاعات ترجیحی مسأله را رفع کند. اگرچه پیاده‌سازی این روش ساده است ولی نمی‌تواند دغدغه‌ی مهم دیگر مربوط به عدم توانایی دستیابی به جبهه بهینه پرِتو را برطرف کند. برعکس، در روش‌های پسینی، توابع هدف باهم ترکیب نشده و ماهیت چندهدفه بودن مسأله حفظ می‌شود تا امکان بررسی رفتار مسأله در طیفی از پارامترهای طراحی و شرایط عملیاتی فراهم شود [۱۱]. واضح است که در روش‌های پسینی، برخلاف پیشینی، با چندین جواب بهینه مواجه خواهیم شد که یکی از آن‌ها توسط طراح انتخاب می‌شود. الگوریتم‌های بهینه-سازی چندهدفه ژنتیک با مرتب‌سازی نامغلوب (NSGA) [۱۲] و ازدحام ذرات (MOPSO) [۱۳] نمونه‌های آشنایی از روش‌های پسینی محسوب می‌شوند.

حفظ تنوع جمعیت و نخبه‌گرایی دو هدف اصلی الگوریتم‌های فراابتکاری چندهدفه مبتنی بر روش‌های پسینی هستند. حفظ تنوع از همگرایی زودرس جلوگیری کرده و تضمین می‌کند که مجموعه پرِتو حاصل شده دارای توزیع و گستردگی بالایی است. نخبه-گرایی نیز تضمین می‌کند که جواب‌های نامغلوب در طی فرایند تکامل حفظ خواهند شد [۱۴]. اکثر الگوریتم‌های بهینه‌سازی فرا-ابتکاری چندهدفه از طبیعت الهام گرفته شده‌اند که با توجه به نوع منبع الهام در چهار گروه مبتنی بر تکامل، مبتنی بر ازدحام، مبتنی بر فیزیک و مبتنی بر انسان دسته‌بندی می‌شوند.

الگوریتم‌های مبتنی بر تکامل از فرایند تکامل بیولوژیکی نشأت می‌گیرند و مکرراً از عملگرهای بیولوژیکی مثل انتخاب، تولیدمثل و جهش استفاده می‌کنند تا جمعیت ابتدایی را در جهت رسیدن به

1. Pareto set
2. Trade-off
3. Priori
4. Posteriori
5. Weighted sum
6. Weighted min–max
7. Goal programming
8. Lexicographic
9. Dynamic weighted aggregation

مجموعه بهینه پرتو تکامل ببخشند [۱۵]. الگوریتم ژنتیک چندهدفه (MOGA) [۱۶] یکی از مشهورترین الگوریتم‌ها در این گروه است که از اصل تکاملی داروین (اصل بقای شایسته‌ترین‌ها) الهام می‌گیرد. این الگوریتم در واقع بیانگر تئوری انتخاب طبیعی است که در آن، مناسب‌ترین افراد برای ادامه نسل و تولید فرزندان انتخاب می‌شوند. این الگوریتم با استفاده از روش مجموع وزن‌دار، توابع هدف را با هم ترکیب کرده و یک مقدار برازندگی به هر جواب نسبت می‌دهد. سپس تعدادی از آن‌ها را برای عمل تولیدمثل انتخاب می‌کند. الگوریتم NSGA2 [۱۷]، نسخه اصلاح شده NSGA، جبهه‌های نامغلوب جمعیت فعلی را رتبه‌بندی کرده و با توجه به این رتبه‌ها، برازندگی هر جواب را مشخص می‌کند. همچنین از فاصله ازدحامی راه‌حل‌ها جهت ایجاد تنوع در انتخاب راه‌حل‌ها برای عمل ترکیب و تولید نسل بعدی بهره می‌برد. اگرچه این الگوریتم از آرشیو خارجی جهت نگهداری راه‌حل‌های بهینه پرِتو استفاده نمی‌کند ولی با انتخاب بهترین‌ها از میان نسل والد و فرزندان تولید شده، نخبه‌گرایی را تقویت می‌کند. الگوریتم‌های SPEA2 [۱۷]، NPGA [۱۸]، PDE [۱۹] و DOMOEA [۱۸] نمونه‌های مشهوری از الگوریتم‌های مبتنی بر تکامل هستند.

الگوریتم‌های مبتنی بر ازدحام، رفتار برخی از حیوانات از جمله پرندگان، مورچه‌ها و خفاش‌ها را شبیه‌سازی می‌کنند و بر مبنای اشتراک اطلاعات کسب شده توسط همه افراد در طی فرایند تکامل می‌باشند. بهینه‌سازی چندهدفه ازدحام ذرات (MOPSO) [۱۴]، یکی از مشهورترین الگوریتم‌ها در این گروه است که از رفتار دسته‌جمعی پرندگان برای جستجوی غذا الهام گرفته شده است. ایده اصلی الگوریتم پرندگان، دنبال‌کردن پرنده‌ای است که کمترین فاصله را تا غذا دارد. در این الگوریتم، هر پرنده معرف یک جواب است که یک مقدار برازندگی دارد که نشانگر میزان نزدیکی به غذاست. هر پرنده دارای یک موقعیت در فضای جستجو و یک مؤلفه‌ی سرعت است که موجب هدایت حرکت و تغییر موقعیت آن پرنده می‌شود. MOPSO میزان برازندگی راه‌حل‌ها را با توجه به تراکم آن‌ها در اطراف راه‌حل‌های نامغلوب مشخص می‌کند. به‌علاوه، این الگوریتم از آرشیو خارجی برای نخبه‌گرایی استفاده می‌کند. بهینه‌ساز چندهدفه شاهین هریس (MOHHO) [۳] الگوریتم دیگری در این گروه است که از رفتار غافلگیرانه‌ی شاهین‌های هریس در تعقیب‌وگریز طعمه الهام گرفته شده است. در این الگوریتم، چندین شاهین با همکاری یکدیگر طعمه را از جهات مختلف تعقیب می‌کنند تا آن را غافلگیر کنند. MOHHO علاوه بر حفظ ساختارهای الگوریتم شاهین هریس (HHO) [۲۰] از یک آرشیو خارجی جهت ذخیره و بازیابی راه‌حل‌های بهینه پرِتو استفاده می‌کند. این آرشیو برای شبیه‌سازی شاهین‌ها و طعمه استفاده می‌شود، طوری که یک راه‌حل از کم‌جمعیت‌ترین منطقه آرشیو با استفاده از فرآیند چرخ رولت انتخاب شده و به عنوان طعمه استفاده می‌شود. از الگوریتم‌های دیگر در این گروه می‌توان به مواردی همچون الگوریتم بهینه‌ساز چندهدفه شیر مورچه (MOALO) [۲۱]، الگوریتم بهینه‌ساز چندهدفه مرغ مگس‌خوار (MOAHA) [۲۲]، الگوریتم بهینه‌ساز چندهدفه شامپانزه (MOCHOA) [۲۰]، الگوریتم بهینه‌ساز چندهدفه - گرگ خاکستری (MOGWO) [۱۲]، الگوریتم بهینه‌ساز چندهدفه شکارچیان دریایی (MOMPA) [۲۳]، بهینه‌سازی ازدحام ذرات چند راهنما (MGPSO) [۲۴]، بهینه‌ساز شامپانزه کوتوله چندهدفه (MOBO) [۲۵] اشاره کرد.

الگوریتم‌های مبتنی بر علوم فیزیکی مبتنی بر قوانین فیزیک در طبیعت هستند. الگوریتم بهینه‌ساز چندهدفه چرخه آب (MOWCA) یکی از مشهورترین الگوریتم‌ها در این گروه است که از فرایند چرخه آب در طبیعت الگو گرفته است [۲۶]. این الگوریتم در فرار از بهینگی محلی و سرعت زیاد در رسیدن به جبهه پرِتو توانمندی بالایی دارد. الگوریتم جستجوی گرانشی چندهدفه (MOGSA) [۲۷] الگوریتم شناخته شده دیگری است که مبتنی بر الگوریتم جستجوی گرانشی (GSA) است. در الگوریتم GSA که از برهم‌کنش گرانشی بین اجسام الهام گرفته است [۲۸]، هر راه‌حل به صورت یک شئ در نظر گرفته می‌شود که جرم آن، میزان برازندگی آن را نشان می‌دهد. در طی تکرارهای الگوریتم،

نیروی گرانشی مابین اشیاء باعث جذب اشیاء سبک‌تر به سمت اشیاء سنگین‌تر می‌شوند که این موضوع باعث می‌شود نهایتاً شئ با بیشترین جِرم یا برازنده‌ترین راه‌حل باقی بماند. الگوریتم بهینه‌ساز چندهدفه مبتنی بر گرادیان (MOGBO) [۲۹]، الگوریتم بهینه‌ساز تعادل چندهدفه (MOEO) [۳۰]، الگوریتم ترکیبی چندهدفه مبتنی بر گرادیان به کمک جانشین (GS-MOHA) [۳۱] و الگوریتم بهینه‌سازی ساختار کریستال چندهدفه (MOCryStAl) [۳۲] و الگوریتم قالب متعامد چندهدفه (MOOSMA) [۳۳] نمونه‌های دیگری از این گروه هستند.

الگوریتم‌های مبتنی بر انسان از رفتار اجتماعی و تعامل بین انسان‌ها در محیط اجتماعی الگو می‌گیرند. الگوریتم بهینه‌سازی مبتنی بر آموزش و یادگیری چندهدفه[1] (MOTLBO) [۳۴] یکی از الگوریتم‌های مشهور در این گروه است که فرایند دومرحله‌ای یادگیری و آموزش در یک کلاس درسی را شبیه‌سازی می‌کند [۳۵]. در مرحله‌ی آموزش، بهترین فرد کلاس به عنوان معلم انتخاب شده و شایستگی بقیه افراد جمعیت را ارتقاء می‌دهد در حالی که در مرحله‌ی یادگیری، افراد کلاس دانش کسب شده در مرحله آموزش را به بقیه منتقل می‌کنند. الگوریتم MOTLBO از مفاهیم مرتب‌سازی جبهه‌های نامغلوب و فاصله ازدحامی استفاده کرده و یک راه‌حل با بیشترین فاصله ازدحامی را به‌عنوان معلم انتخاب می‌کند.

پیوست الف-۱ خلاصه‌ای از جزئیات الگوریتم‌ها از جمله تکنیک مورد استفاده، سال ارائه، معیارهای کارایی، مزایا، و معایب را نشان می‌دهد.

اگرچه الگوریتم‌های فراابتکاری چندهدفه‌ی زیادی در سال‌های اخیر ارائه شده‌اند، اما مطابق با قضیه NFL[2] [۳۶]، هیچ الگوریتم فراابتکاری وجود ندارد که بتواند تمام مسائل بهینه‌سازی موجود و پیش‌رو را حل کند. به عبارت ساده‌تر، این قضیه بیان می‌کند که یک الگوریتم خاص ممکن است گروهی از مسائل بهینه‌سازی چندهدفه را حل کند، با این حال در حل مسائل دیگر ناتوان خواهد بود. با توجه به این قضیه و با توجه به موفقیت الگوریتم تکامل شوراهای شهر (CCE) در حل مسائل بهینه‌سازی تک‌هدفه، نسخه چندهدفه‌ای از این الگوریتم را ارائه می‌کنیم.

در این مقاله، نسخه چندهدفه‌ای از الگوریتم تکامل شوراهای شهر (CCE) با نام الگوریتم تکامل شوراهای شهر چندهدفه (MOCCE) ارائه می‌شود. در الگوریتم ارائه شده، از آرشیو راه-حل‌های بهینه برای تعریف ساختار هرم‌گونه شوراهای شهرها و شبیه‌سازی تکامل آن در فضاهای جستجوی چندهدفه استفاده می‌شود. برای ارزیابی و مقایسه‌ی کارایی الگوریتم MOCCE با کارایی الگوریتم‌های بهینه‌سازی شیر مورچه چندهدفه (MOALO)، کپک مخاطی چندهدفه (MOSMA) و مرغ مگس-خوار مصنوعی چندهدفه (MOAHA)، آن‌ها را روی ۱۸ تابع تست چندهدفه شناخته شده موسوم به UF و IMOP اجرا می‌کنیم. به‌علاوه، کارایی هر الگوریتم از نظر معیارهای فاصله نسلی معکوس (IGD)، فاصله نسلی (GD) و بیشنیه گستردگی (MS) مورد بررسی قرار می‌گیرد. برخی از دستاوردهای اصلی الگوریتم ارائه شده عبارتند از:

۱- استفاده از الگوریتم تکامل شوراهای شهر برای حل مسائل بهینه‌سازی چندهدفه

۲- ارائه یک هیوریستیک جدید برای بدست آوردن مقدار هدف واحد به‌ازای هر راه‌حل جهت ایجاد یک درخت موسوم به درخت شوراها که به صورت یک هرم بیشینه است.

در ادامه و در بخش ۲، برخی مفاهیم پایه درباره بهینه‌سازی چندهدفه و الگوریتم تکامل شوراهای شهر، مطرح و مرور می‌شوند. بخش ۳ به جزئیات الگوریتم MOCCE می‌پردازد. در بخش ۴، جزئیات پیاده‌سازی و نتایج تجربی ارائه می‌شوند. بخش ۵ نیز به نتیجه‌گیری مقاله و پیشنهادهایی برای کارهای بعدی اختصاص دارد.

1. Teaching-learning-based optimization
2. No Free Lunch theorem

۲. پیش‌زمینه

در این بخش، برخی مفاهیم پایه درباره بهینه‌سازی چندهدفه و الگوریتم تکامل شوراهای شهر ارائه و توصیف می‌شوند.

۱،۲ بهینه‌سازی چندهدفه

بهینه‌سازی چندهدفه به بهینه‌سازی یک مسأله با بیش از یک تابع هدف می‌پردازد. یک مسأله بهینه‌سازی چندهدفه (کمینه‌سازی) با n متغیر تصمیم‌گیری، o به‌عنوان تابع هدف و m قید به صورت زیر تعریف می‌شود:

$$\begin{aligned} &\text{Minimize: } y = f(x) = \big(f_1(x), f_2(x), \cdots, f_n(x)\big) \\ &\text{Subject to: } e(x) = (e_1(x), e_2(x), \cdots, e_m(x)) \le 0 \\ &\text{Where: } x = (x_1, x_2, \cdots, x_n) \in X, L_i \le x_i \le U_i, i = 1,2,\cdots,n, \\ &y = (y_1, y_2, \cdots, y_o) \in Y \end{aligned} \quad (۱)$$

در این رابطه، x بردار تصمیم‌گیری، y بردار هدف، X فضای تصمیم و Y فضای هدف است.

در مسائل بهینه‌سازی چندهدفه، مقایسه دو راه‌حل با توجه به وجود چندین تابع هدف با استفاده از عملگرهای رابطه‌ای (یعنی $\succ$، $\prec$ و $=$) امکان‌پذیر نیست. در این حالت، راه‌حل x بر راه‌حل x' برتری دارد (غلبه می‌کند) اگر و تنها اگر، مقادیر تمام توابع هدف برای راه‌حل x نسبت به راه‌حل x' کمتر یا مساوی بوده و برای حداقل یک تابع، مقدار آن برای راه‌حل x نسبت به راه‌حل x' کمتر باشد [۱۲].

تعریف ۱ (غلبگی پرِتو[1]): بردار تصمیم $x = (x_1, x_2, \cdots, x_n)$ با بردار هدف $y = (y_1, y_2, \cdots, y_o)$ بر بردار تصمیم $x' = (x'_1, x'_2, \cdots, x'_n)$ با بردار هدف $y' = (y'_1, y'_2, \cdots, y'_o)$ غلبه دارد ($x \succ x'$) اگر و تنها اگر رابطه $y_i \le y'_i, 1 \le i \le o$ برقرار باشد. علاوه بر این، برای حداقل یک تابع هدف j، داشته باشیم: $y_j \le y'_j$ به عبارت دیگر، x' در هیچ تابع هدفی بهتر از x نیست.

در این صورت، گفته می‌شود راه‌حل x' مغلوب راه‌حل x است که با تابع dominate(x, x') نشان داده می‌شود.

با توجه به این تعریف می‌توان نتیجه گرفت که راه‌حل x بر راه‌حل x' غلبه ندارد (نامغلوب) اگر برای حداقل یک تابع هدف j، داشته باشیم: $y_j \le y'_j$.

تعریف ۲ (بهینگی پرِتو[2]): گفته می‌شوذ که x یک راه‌حل نامغلوب است اگر و تنها اگر، هیچکدام از راه‌حل‌های موجود در جمعیت فعلی بر آن غلبه نکند. به عبارت دیگر، راه‌حلی مثل x' وجود نداشته باشد که مقدار برگشتی تابع dominate(x, x') برابر true باشد.

تعریف ۳ (مجموعه و جبهه بهینه پرِتو[3]): به مجموعه همه جواب‌های (بردارهای) نامغلوب، مجموعه بهینه پرِتو گفته می‌شود و بردارهای هدف این قبیل راه‌حل‌های نامغلوب، جبهه بهینه پرِتو را تشکیل می‌دهند.

۲،۲ الگوریتم تکامل شوراهای شهر (CCE)

در کشورهای دموکراتیک، شورای عالی هر شهر، شهردار و مسئولان آن شهر را انتخاب می‌کنند. شورای عالی نیز با تشکیل شوراها از کوچکترین محله‌ها به بزرگترین محله‌ها، مناطق و در نهایت، کل شهر شکل می‌گیرد. اعضای شورای یک منطقه سعی می‌کنند عملکرد خود را بهبود ببخشند تا در انتخابات آینده به عنوان رئیس شورا انتخاب شده و به عنوان عضو شورای منطقه بزرگتر نیز انتخاب شوند. محبوبیت، میزان تحصیلات، تجارب اجرایی و از این دست می‌توانند جزو معیارهای کارایی در نظر گرفته شوند.

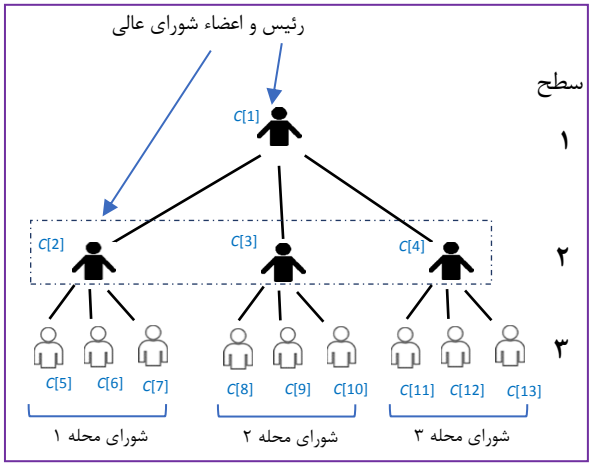
الگوریتم CCE یک الگوریتم فراابتکاری الهام‌گرفته از فرایند تشکیل شورای عالی یک شهر است [۳۷]. در این الگوریتم، همه اعضاء و رؤسای شوراها در یک ساختار سلسله‌مراتبی، همچون

1. Pareto dominance

2. Pareto optimality

3. Pareto optimal set and front

درخت (در اصطلاح، به نام درخت شوراها) نگهداری می‌شوند. فرض کنیم یک شهر دارای سه شورای محلی و یک شورای عالی می‌باشد. مطابق با درخت شورای مرتبط با این شهر (شکل ۱)، ریشه درخت ($C[1]$) در سطح یک قرار گرفته و رئیس شورای عالی را نگهداری می‌کند. بقیه اعضای شورای عالی در سطح دو و در خانه‌های $C[2]$، $C[3]$ و $C[4]$ قرار خواهند گرفت. با توجه به مفهوم کارایی اعضاء و رؤسای شوراها و این که رؤسا در درخت شوراها در سطوح پایین‌تری نسبت به اعضاء قرار می‌گیرند می‌توان چنین استنباط کرد که درخت شوراها به‌صورت یک هرم بیشینه است. در یک هرم بیشینه که معمولاً در آرایه نگهداری می‌شود، مقدار هر گره والد از مقادیر همه گره‌های فرزندان بیشتر یا مساوی است. با فرض این که، تعداد اعضاء هر شورا برابر d و ارتفاع درخت شوراها برابر h باشد تعداد کل اعضاء (اندازه آرایه C) برابر $N = 1 + d^1 + ... + d^h = (d^h - 1) / (d - 1)$ خواهد بود. اگر رئیس یک شورا در خانه j–ام آرایه باشد اعضاء این شورا در خانه‌های $d*j-d+2$, $d*j+1$, ... نگهداری خواهند شد. برعکس، برای یک عضو شورا که در خانه i–ام است رئیس این شورا در خانه $\left\lfloor \frac{i-2}{d} \right\rfloor + 1$ نگهداری خواهد شد. نکته قابل ذکر این است که هر $C[i]$ به‌صورت یک بردار با طول تعداد ملاک‌های کارایی است که مقادیر آن نشان‌دهنده نمرات عضو i–ام شورا در این ملاک‌هاست.



شکل ۱: نمونه‌ای از درخت شوراهای یک شهر فرضی [۳۷]

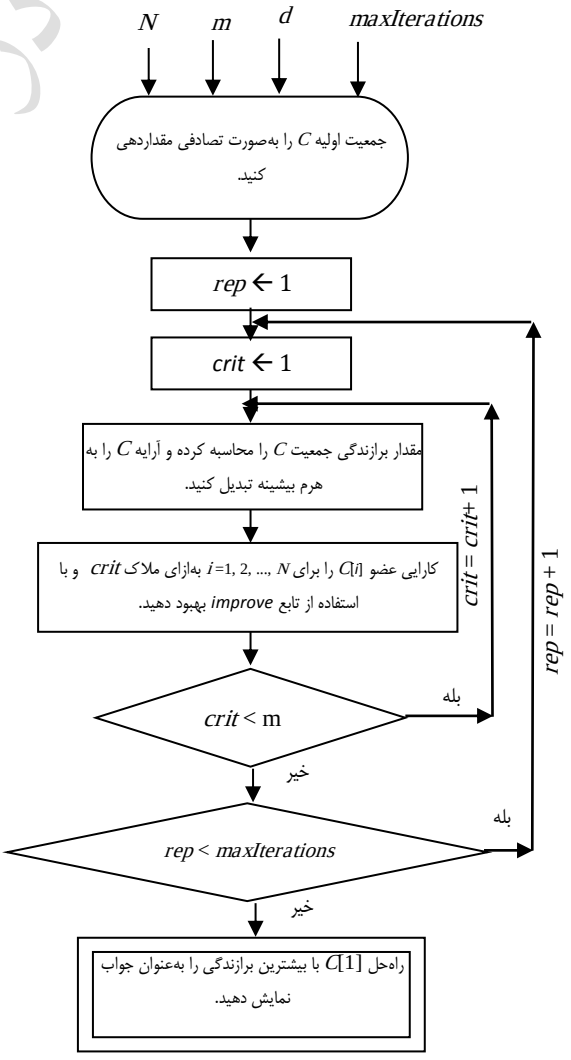
فرض کنید اندازه‌ی جمعیت (تعداد اعضاء و رؤسای شورها) و تعداد متغیرها (تعداد ملاک‌های کارایی) با متغیرهای N و m نشان داده شوند جمعیت C در یک ماتریس $N \times m$ نگهداری می‌شود که سطرهای این ماتریس (N سطر) راه‌حل‌های کاندید به طول m را مشخص می‌کنند. واضح است که مجموع کارایی (نمرات) یک عضو (راه‌حل) در ملاک‌های ارزیابی، میزان برازندگی آن راه‌حل را نشان می‌دهد. الگوریتم CCE با یک جمعیت اولیه از راه‌حل‌های کاندید که به‌صورت تصادفی تولید شده‌اند، فرایند تکامل را آغاز می‌کند. در ادامه، دو مرحله تبدیل و تکامل را به‌ازای هر کدام از m متغیر (ملاک) و به تعداد مشخص شده توسط کاربر (*maxIterations*) تکرار می‌کند.

- **مرحله تبدیل**: آرایه C به یک هرم بیشینه تبدیل می‌شود.
- **مرحله تکامل**: برازندگی (کارایی) همه اعضای شوراها از سطح یک (ریشه درخت) تا آخرین سطح درخت با استفاده از تابع *improve* بهبود پیدا می‌کند. تابع *improve* برای بهبود کارایی هر عضو جمعیت (به‌عنوان عضو شورا) با توجه به عضو شایسته‌تر (به‌عنوان رئیس شورا) استفاده می‌کند. این تابع که از روش ترکیب ریاضی استفاده می‌کند دو عضو جمعیت با نام‌های X (به‌عنوان عضو شورا) و Y (به‌عنوان رئیس شورا) به همراه شماره ملاک و کران‌های پایین و بالای ملاک مورد نظر را گرفته و کارایی X را در ملاک cr–ام با توجه به Y افزایش می‌دهد.

با توجه به تعریف درخت شوراها که به صورت یک هرم بیشینه است. در پایان الگوریتم، $C[1]$ دارای بیشترین برازندگی خواهد بود و به عنوان بهترین راه‌حل به کاربر اعلام می‌شود. شکل ۲، فلوچارت الگوریتم CCE را نشان می‌دهد.

۳. الگوریتم تکامل شوراهای شهر چندهدفه (MOCCE)

همانطورکه در بخش قبلی عنوان شد، الگوریتم CCE در طی فرآیند اجرا، راه‌حل‌های موجود در جمعیت جاری را در یک درخت شوراها که به صورت یک هرم بیشینه است، نگهداری می‌کند. در مسائل بهینه‌سازی تک‌هدفه که فقط با یک تابع هدف مواجه هستیم امکان ایجاد چنین درختی وجود دارد در حالی که در مسائل بهینه‌سازی چندهدفه با چندین تابع هدف نمی‌توان ما بین دو راه‌حل مشخص کرد که کدام یکی والد دیگری شود. بنابراین، مهمترین مسأله در الگوریتم MOCCE نحوه ایجاد درخت شوراها هست. برای بدست آوردن مقدار هدف واحد به‌ازای هر راه‌حل، مقدار هر تابع هدف را در یک وزنی (ضریبی) ضرب کرده و حاصل‌جمع آن‌ها را بدست می‌آوریم. آرایه *weightsCF* این وزن‌ها را نگهداری می‌کند که در ابتدا مقادیر اولیه آن‌ها یکسان و برابر $\frac{1}{o}$ است (o تعداد توابع هدف را نشان می‌دهد).



شکل ۲: فلوچارت الگوریتم CCE [۳۷]

بعد از مشخص کردن مجموعه بهینه پرتو با استفاده از تعریف ۳ و ذخیره آن‌ها در آرایه ***Archive***، مقادیر وزن توابع هدف (آرایه *weightsCF*) به این صورت محاسبه می‌شود: با این فرض که *nArch* تعداد راه‌حل‌های موجود در آرایه *Archive* می‌باشد، ابتدا با استفاده از رابطه (۲)، مقادیر آرایه *temp* به طول o محاسبه می‌شوند.

$$temp[j] = \sum_{i=1}^{nArch} \frac{f_j(Archive[i])}{\sum_{k=1}^{nArch} f_k(Archive[i])} \quad , 1 \le j \le o \qquad (۲)$$

بعد از محاسبه عناصر آرایه *temp*، عناصر آرایه *weightsCF* از رابطه (۳) محاسبه می‌شوند.

$$weightsCF[j] = \frac{temp[j]}{\sum_{k=1}^{o} temp[k]} \quad , 1 \le j \le o \qquad (۳)$$

به‌عنوان مثال، فرض کنید که یک مسأله بهینه‌سازی ۳ تابع هدف دارد (یعنی $o = 3$) و مقادیر توابع هدف برای ۴ راه‌حل موجود در مجموعه بهینه پرتو (*Archive*) به صورت {(0.12, 0.38, 0.21), (0.22, 0.15, 0.43), (0.49, 0.17, 0.11), (0.35, 0.31, 0.20)} می‌باشد (یعنی *nArch* = 4). ابتدا مقادیر آرایه *temp* را با استفاده از رابطه (۲) محاسبه می‌کنیم که برابر (1.48, 1.21, 1.20) خواهد بود. برای مثال، مقدار *temp*[1] به صورت زیر محاسبه می‌شود:

$$temp[1] = \frac{0.12}{0.12+0.38+0.21} + \frac{0.22}{0.22+0.15+0.43} + \frac{0.49}{0.49+0.17+0.11} + \frac{0.35}{0.35+0.31+0.20} = 1.48$$

بعد از محاسبه عناصر آرایه *temp*، عناصر آرایه *weightsCF* را با استفاده از رابطه (۳) محاسبه می‌کنیم که برابر (0.38, 0.31, 0.31) خواهد بود. برای مثال مقدار *weightsCF*[1] به صورت زیر محاسبه می‌شود:

$$weightsCF[1] = \frac{1.48}{1.48+1.21+1.20} = 0.38$$

بعد از محاسبه مقادیر وزن توابع هدف (آرایه *weightsCF*)، باید جمعیت فعلی M را با استفاده از تابع *sortPopulation* به درخت شوراها که به صورت یک هرم بیشینه می‌باشد تبدیل کنیم. در واقع، به‌ازای هر کدام از راه‌حل‌ها، مقدار واحدی از توابع هدف را براساس ضرایب بدست آمده در آرایه وزن‌ها (*weightsCF*) محاسبه کرده و با توجه به این مقادیر، راه‌حل‌ها را به صورت نزولی مرتب می‌کنیم که این باعث می‌شود مقدار هر گره از مقادیر همه فرزندان آن بیشتر شود (یعنی آن گره، از گره‌های فرزند خود برازنده‌تر است). با توجه به این که راه‌حل‌های موجود در گره‌های سطح بالای درخت شوراها (نزدیک به برگ‌های درخت) نسبت به راه‌حل‌های موجود در سطوح پایین (نزدیک به ریشه درخت) بهبود داده می‌شوند، بنابراین ابتدا جمعیت فعلی M را به دو زیرجمعیت M' (شامل تمام راه‌حل‌های نامغلوب) و M'' (شامل تمام راه‌حل‌های مغلوب) تقسیم می‌کنیم. سپس هر کدام از راه‌حل‌های موجود در زیرجمعیت‌های M' و M'' را به صورت جداگانه به صورت نزولی مرتب کرده و سپس آن‌ها را دوباره در جمعیت M کپی می‌کنیم. این کار باعث می‌شود که راه‌حل‌های نامغلوب در سطوح پایین درخت و راه‌حل‌های مغلوب در سطوح بالای درخت قرار بگیرند.

مشابه با الگوریتم فراابتکاری چندهدفه، الگوریتم MOCCE نیز با جمعیت اولیه‌ای از راه‌حل‌ها که در ابتدا به صورت تصادفی تولید شده‌اند، آغاز می‌شود. بعد از محاسبه برازندگی جمعیت فعلی و ایجاد آرشیوی از راه‌حل‌های نامغلوب جمعیت فعلی، مراحل زیر تا وقتی تکرار می‌شوند که تعداد فراخوانی‌های تابع برازندگی از مقدار *FESmax* (حداکثر مقدار پیش‌بینی‌شده) کمتر است:

مرحله ۱: شبکه‌بندی فضای هدف کشف شده (آرشیو) با توجه به اندازه شبکه و پارامتر تورم شبکه[1] (α)

مرحله ۲: محاسبه موقعیت هر کدام از راه‌حل‌های نامغلوب در شبکه ایجاد شده

مرحله ۳: محاسبه مقادیر وزن توابع هدف (آرایه *weightsCF*)

مرحله ۴: انجام دو مرحله زیر به‌ازای هر کدام از m متغیر (ملاک) و برای تک‌تک راه‌حل‌ها:

مرحله ۴-۱: تبدیل جمعیت فعلی به یک هرم بیشینه با استفاده از تابع sortPopulation

مرحله ۴-۲: بهبود راه‌حل فعلی (*child*) با استفاده از تابع Improve و با توجه به میانگینِ دو راه‌حل زیر: ۱- والدش در درخت ۲- رهبرش که توسط تابع selectLeader انتخاب می‌شود. این تابع پارامترهای آرشیو (*Archive*) و فشار انتخاب[2] (β) را به عنوان ورودی گرفته و یک راه‌حل را به صورت زیر انتخاب کرده و به‌عنوان رهبر بر می‌گرداند. این تابع در ابتدا خانه‌های شبکه که شامل حداقل یک راه‌حل نامغلوب است را به‌همراه تعداد راه‌حل‌های نامغلوب موجود در آن‌ها را پیدا می‌کند (k). سپس با استفاده از رابطه (۴) احتمال انتخاب هر خانه را محاسبه می‌کند. بعد از محاسبه احتمالات، از روش چرخ‌رولت استفاده کرده و یکی از خانه‌ها را انتخاب می‌کند و در نهایت، یکی از راه‌حل‌هایی که در این خانه قرار دارند را به صورت تصادفی انتخاب می‌کند.

$$P = k^{-\beta} \qquad (۴)$$

اگر راه‌حل بهبودیافته (Y)، راه‌حل *child* را مغلوب کند جایگزین آن خواهد شد. ولی اگر راه‌حل *child*، راه‌حل بهبودیافته (Y) را مغلوب کند هیچ تغییری صورت نمی‌گیرد. در غیر این صورت، یعنی هیچکدام همدیگر را مغلوب نکنند با احتمال ۵۰٪، راه‌حل Y جایگزین *child* می‌شود.

مرحله ۵: محاسبه اعضای نامغلوب جمعیت فعلی و به‌روزرسانی آرشیو موجود

1. Grid Inflation Parameter

2. Selection Pressure

مرحله ۶: نگهداری تعداد *nArch* از راه‌حل‌های نامغلوب در آرشیو و حذف اضافه‌ها

شکل ۳، شبه‌کد الگوریتم MOCCE را نشان می‌دهد. در خطوط ۲-۶ این الگوریتم، جمعیت اولیه به صورت تصادفی تولید شده و برازندگی آن‌ها نیز محاسبه می‌شود. در خط ۷، آرشیوی از راه‌حل‌های نامغلوب جمعیت فعلی با استفاده از تابع getNonDom ایجاد می‌شود. کارِ اصلی الگوریتم با تکرار حلقه‌ی while تا موقعی که تعداد فراخوانی‌های تابع برازندگی از مقدار *FESmax* (حداکثر مقدار پیش‌بینی‌شده) کمتر باشد شروع می‌شود (خطوط ۸-۲۹). در خط ۹ فضای هدف کشف شده (آرشیو) با استفاده از تابع createHypercubes و با توجه به اندازه شبکه و پارامتر تورم شبکه شبکه‌بندی شده و سپس با استفاده از تابع setGridIndex اندیس محل قرارگیری هرکدام از راه‌حل‌های نامغلوب مشخص می‌شوند. در خط ۱۱، مقادیر وزن توابع هدف (آرایه *weightsCF*) با استفاده از تابع computeWCF محاسبه می‌شوند. در خطوط ۱۲-۲۶، به‌ازای هرکدام از *m* متغیر (ملاک)، ابتدا جمعیت فعلی به یک هرم بیشینه با استفاده از تابع sortPopulation تبدیل شده سپس تک‌تک راه‌حل‌ها، با استفاده از تابع Improve و با توجه به میانگینِ راه‌حل والد و راه‌حل رهبر که توسط تابع selectLeader انتخاب شده است، بهبود داده می‌شوند. در خط ۲۷، اعضای نامغلوب جمعیت فعلی محاسبه شده و آرشیو موجود به‌روزرسانی می‌شود و در خط ۲۸ نیز تعداد *nArch* از راه‌حل‌های نامغلوب در آرشیو نگهداری شده اضافی‌ها حذف می‌شوند. در پایان الگوریتم، مجموعه آرشیو به‌عنوان راه‌حل‌های نامغلوب به کاربر اعلام می‌شوند.

۱،۳ پیچیدگی الگوریتم ارائه شده

برای محاسبه پیچیدگی الگوریتم MOCCE، ابتدا توابع کلیدی که در الگوریتم فراخوانی شده‌اند را بررسی کرده، سپس به کل الگوریتم می‌پردازیم. با توجه به شبه‌کد الگوریتم MOCCE (الگوریتم ۱)، توابع کلیدی به صورت زیر در این الگوریتم فراخوانی می‌شوند: (۱) تابع getNonDom: چون این تابع، هر راه‌حل را با تمام راه‌حل‌های دیگر مقایسه می‌کند، لذا پیچیدگی این تابع از مرتبه $O(N^2)$ است؛ (۲) پیچیدگی توابع createHypercubes و setGridInde از مرتبه $O(nFunc)$ است؛ (۳) پیچیدگی تابع computeWCF با توجه به فرمول‌های ۲ و۳، از مرتبه $O(nFunc*nArch)$ خواهد بود؛ (۴) تابع sortPopulation در ابتدا جمعیت موجود را به صورت یک هرم بیشینه درآورده ($O(N)$)، سپس جمعیت غیرمغلوب و مغلوب را جداگانه مرتب می‌کند (به روش مرتب‌سازی حبابی)، لذا پیچیدگی آن از مرتبه $O(N+dom*dom+(N-dom)*(N-dom)) = O(N^2)$ خواهد بود، با این فرض که تعداد راه‌حل‌های مغلوب برابر *dom* باشد (حداکثر N)، و (۵) پیچیدگی توابع selectLeader و deleteExtra از مرتبه $O(nArch)$ می‌باشد. بعد از محاسبه پیچیدگی توابع مورد استفاده، به کل الگوریتم می‌پردازیم. پیچیدگی خطوط ۲ تا ۶ از مرتبه $O(N)$ است. پیچیدگی خط ۷ که در آن تابع getNonDom فراخوانی شده برابر $O(N^2)$ است. خطوط ۹ تا ۲۸ در یک حلقه به تعداد *FESmax* تکرار می‌شود که پیچیدگی این قسمت در مقدار *FESmax* ضرب می‌شود. به طور خلاصه، پیچیدگی الگوریتم MOCCE به صورت رابطه (۵) بدست می‌آید:

$$O(N+N^2+FESMax*(nFunc+nFunc+nFunc*nArch +dim*(N^2+N*Arch+N^2+nArch))) = O(N^2+FESMax*(nFunc*nArch+dim*N^2)) \quad (۵)$$

۴. نتایج تجربی

در این بخش، کارایی الگوریتم MOCCE را از نظر معیارهای فاصله نسلی (GD)، فاصله نسلی معکوس (IGD) و بیشینه گستردگی (MS) روی ۱۸ تابع تست چندهدفه شناخته شده موسوم به UF [۱۲] و IMOP [۳۸] مورد بررسی قرار داده و نتایج بدست آمده را با نتایج الگوریتم‌های MOALO، MOSMA و MOAHA مقایسه می‌کنیم. GD یک معیار ارزیابی برای تخمین فاصله اقلیدسی میان راه‌حل‌های بهینه پرِتو محاسبه شده و راه‌حل-

های بهینه پرِتو درست می‌باشد [۳۹] و با استفاده از رابطه (۶) محاسبه می‌شود:

$$GD = \frac{\sum_{i=1}^{l} d_i^2}{2} \qquad (۶)$$

در اینجا، l تعداد راه‌حل‌های بهینه پرِتو محاسبه شده و d_i فاصله اقلیدسی مابین i–امین راه‌حل بهینه پرِتو محاسبه شده و نزدیکترین راه‌حل بهینه پرِتو درست می‌باشد. در حالی که IGD یک معیار ارزیابی برای محاسبه فاصله اقلیدسی میان راه‌حل‌های بهینه پرِتو درست و راه‌حلهای بهینه پرِتو محاسبه شده می‌باشد [۴۰] و با استفاده از همان رابطه (۶) محاسبه می‌شود با این تفاوت که که l به تعداد راه‌حل‌های بهینه پرِتو درست و d_i به فاصله اقلیدسی مابین i–امین راه‌حل بهینه پرِتو درست و نزدیکترین راه‌حل بهینه پرِتو محاسبه شده اشاره دارند.

Algorithm 1 The MOCCE algorithm

Input: *FESmax, N, lb, ub, dim, d, height, nArch, nGrid, α, β, γ, f*;
Output: The best non-dominated solutions;
1: *nfe* = 0;
2: **for** *i* = 1 to *N* **do**
3: *M*[*i*].*pos* = *lb* + (*ub*-*lb*)*rand;
4: *M*[*i*].*cost* = *f*(*M*[*i*].*pos*);
5: *nfe* = *nfe* + 1;
6: **end**
7: *Archive* = getNonDom (*M*);
8: **while** *nfe* <= *FESmax* **do**
9: *Grid* = createHypercubes (*Archive*, *nGrid*, *α*);
10: setGridIndex (*Archive, Grid*);
11: *weightsCF* = computeWCF (*Archive*);
12: **for** *g* = 1 to *dim* **do**
13: *M* = sortPopulation (*M*, *weightsCF*);
14: **for** *i* = 1 to *N* **do**
15: *child* = *M*[*i*];
16: **if** *i* == 1 **then** *parent* = *child*;
17: **else** *parent* = *M*[$floor(\frac{i-2}{d}) + 1$]; **end**
18: *leader* = selectLeader (*Archive*, *β*);
19: *X* = (*parent.pos* + *leader.pos*)/2;
20: *Y* = improve (*child, X, g, lb, ub*); *nfe* = *nfe* + 4;
21: **if** dominates (*Y, child*) **then** *M*[*i*] = *Y*;
22: **elseif** dominates (*child, Y*) **then** nothing;
23: **elseif** rand <0.5 **then** *M*[*i*] = *Y*;
24: **end**
25: **end**
26: **end**
27: *Archive* = *Archive* ∪ getNonDom (*M*);
28: deleteExtra (*Archive*, *nArch*, *γ*);
29: **end**
30: **return** *Archive*;

شکل ۳- شبه‌کد الگوریتم MOCCE

MS به گستردگی جبهه بهینه پرِتو پیدا شده اشاره دارد که باید بیشینه شود به این معنی که به‌ازای هر هدف، باید دامنه‌ی گسترده‌ای از مقادیر راه‌حل‌های بهینه پرِتو پیدا شده تحت پوشش قرار بگیرد. این معیار از رابطه (۷) محاسبه می‌شود:

$$MS = \sqrt{\frac{1}{o}\sum_{i=1}^{o}\left\{\frac{\min(f_i^{\max}, F_i^{\max}) - \max(f_i^{\min}, F_i^{\min})}{F_i^{\max} - F_i^{\min}}\right\}^2} \qquad (۷)$$

که o تعداد توابع هدف، f_i^{max} و f_i^{min} به‌ترتیب به بزرگترین و کمترین مقادیر i–امین هدف در جبهه بهینه پرِتو پیدا شده اشاره دارند. همچنین F_i^{max} و F_i^{min} نشان‌دهنده بزرگترین و کمترین مقادیر i–امین هدف در جبهه بهینه پرِتو درست هستند.

IMOP یک مجموعه تست با جبهه بهینه پرتو نامنظم است که شامل ۸ تابع IMOP1، IMOP2، … ، IMOP8 است که IMOP1، …، IMOP3 دارای دو تابع هدف و IMOP4، …، IMOP8 دارای سه تابع هدف هستند. IMOP1 و IMOP2 دارای جبهه بهینه پرتو محدب و مقعر هستند. جبهه بهینه پرتو تابع IMOP3 یک منحنی ناپیوسته چندبخشی است. IMOP4 دارای جبهه بهینه پرتو به صورت یک خط موجی است. جبهه بهینه پرتو تابع IMOP5 از ۸ وجه دایره‌ای شکل تشکیل شده است. تابع IMPOP6 نیز دارای جبهه پرتو با چندین شبکه می‌باشد. جبهه بهینه پرتو تابع IMOP7 بخشی از یک هشتم یک کره هست در حالی که برای تابع IMOP8، جبهه بهینه پرتو مانند یک صفحه پیوسته به نظر می‌رسد اما در واقع از ۱۰۰ قطعه کوچک ناپیوسته تشکیل شده است.

مشابه با IMOP، تابع تست UF نیز فضاهای جستجوی چندهدفه متفاوتی را با جبهه‌های بهینه پرِتو محدب، غیرمحدب، ناپیوسته و چندوجهی ارائه می‌دهد. UF شامل ۱۰ تابع تست با نام‌های UF1، UF2، …، UF10 است که تعداد توابع هدف برای UF1، …، UF7 برابر ۲ و برای UF8، …، UF10 برابر ۳ می‌باشد. در جدول ۱، لینک دسترسی به این توابع تست موجود است.

الگوریتم MOCCE دارای یک پارامتر اختصاصی با نام d است که نشان‌دهنده تعداد فرزندان هر گره در درخت شوراهاست و جزو پارامترهای اختصاصی الگوریتم CCE می‌باشد. مشابه با الگوریتم-های در نظر گرفته، این الگوریتم دارای ۸ پارامتر عمومی است. جدول ۱، اسامی، توضیحات و مقادیر مناسب این پارامترها را نشان می‌دهد. چون بقیه الگوریتم‌ها به جز MOCCE دارای پارامتر اختصاصی نیستند، لذا فقط پارامترهای عمومی در جدول ۱ آورده شده‌اند. لازم به ذکر است که مقدار پارامتر d برای الگوریتم MOCCE برابر ۲ در نظر گرفته شده است.

جدول ۱: اسامی، توصیف و مقادیر مناسب برای پارامترهای الگوریتم‌ها

اسم پارامتر	توصیف	مقدار مناسب
N	اندازه جمعیت	۴۰
$nArch$	اندازه آرشیو	۱۰۰
$FESmax$	حداکثر تعداد فراخوانی‌های تابع برازندگی	۱۰۰۰۰۰
$nGrid$	اندازه شبکه	۱۰
dim	تعداد متغیرها (تعداد ملاک‌های کارایی)	۱۰
α	اندازه تورم شبکه	۰٫۱
β	فشار انتخاب برای انتخاب رهبر	۴
γ	فشار انتخاب برای حذف از آرشیو	۲
لینک دسترسی به توابع تست	https://www.mathworks.com/matlabcentral/fileexchange/135984-uf-and-imop-test-functions	

نتایج گزارش شده در جداول نتایج حاصل اجرای همه الگوریتم‌ها روی هر کدام از توابع تست به تعداد ۲۰ بار می‌باشند. نتایج، شامل معیارهای میانگین (Ave) و انحراف معیار (Std) است. ضمناً، جهت مقایسه عادلانه، پیکربندی پیاده‌سازی و اجرای همه الگوریتم‌ها با نرم‌افزار مطلب نسخه ۲۰۱۷ و CPU Intel® Core™ i5 و RAM 6GB به‌صورت یکسان در نظر گرفته شده‌اند. در این مقاله، از آزمون فریدمن جهت محاسبه میانگین رتبه هر الگوریتم استفاده می‌شود که یک آزمون فرضیه آماری ناپارامتریک بوده و برای مقایسه چندگانه نمونه‌های مرتبط استفاده می‌شود [۴۱].

جداول ۲-۷ نتایج حاصل از اجرای همه الگوریتم‌ها را روی توابع تست UF و IMOP از نظر معیارهای فاصله نسلی (GD)، فاصله نسلی معکوس (IGD) و بیشنیه گستردگی (MS) نشان می‌دهد. مطابق با این جداول، در همه توابع تست UF، الگوریتم MOCCE اولین رتبه را در بین الگوریتم‌ها از لحاظ همه معیارهای GD، IGD و MS کسب کرده است. همچنین، این الگوریتم اولین رتبه را در همه توابع تست IMOP از لحاظ معیار GD و دومین رتبه را از لحاظ معیارهای IGD و MS به خود اختصاص داده است.

در این مقاله، علاوه بر آزمون فریدمن برای مقایسه چندگانه رتبه الگوریتم‌ها، آزمون رتبه علامت‌دار ویلکاکسون نیز جهت مقایسه الگوریتم MOCCE با تک‌تک الگوریتم‌های دیگر استفاده می‌شود. آزمون ویلکاکسون یک آزمون فرضیه آماری ناپارامتریک است که می‌تواند برای مقایسه دوگانه نمونه‌های مرتبط به کار رود [۴۲]. این آزمون دارای خروجی به صورت $R^-/R^+/R^-$ است که تعداد توابعی را نشان می‌دهد که الگوریتم MOCCE دارای عملکرد بهتر/بدتر/یکسان نسبت به الگوریتم‌های دیگر است. به‌علاوه، این آزمون دارای خروجی معیار تصمیم (یا همان Asymp. Sig.) است که به مقدار p-value معروف است که کمتر بودن مقدار آن از ۰٫۰۵ نشان دهنده این است که تفاوت معناداری بین عملکرد الگوریتم MOCCE و سایر الگوریتم‌ها وجود دارد. جدول ۸، نتایج حاصل از اجرای این آزمون را نشان می‌دهد. مطابق با این جدول، مقادیر R^-، بسیار بیشتر از مقادیر R^+ می‌باشند که نشان می‌دهد الگوریتم MOCCE دارای عملکرد بهتری نسبت به الگوریتم‌های دیگر است. ضمناً چون به‌ازای مقایسه‌های MOCCE با MOSMA و MOALO، مقدار p کمتر از ۰٫۰۵ است و به‌ازای مقایسه MOCCE با MOAHA، مقدار p بیشتر از ۰٫۰۵ است، می‌توان نتیجه گرفت که الگوریتم MOCCE تفاوت معناداری با الگوریتم‌های MOSMA وMOALO دارد، اما بین الگوریتم‌های MOCCE و MOAHA تفاوت معناداری وجود

ندارد. علاوه بر این، شکل‌های ۴ و ۵، مجموعه بهینه پرِتو بدست آمده با الگوریتم MOCCE را نشان می‌دهند.

جدول ۲- نتایج الگوریتم‌ها (معیار GD) روی توابع تست UF

		MOCCE	MOAHA	MOSMA	MOALO
UF1	Avg	0.0003	0.0041	0.00012	0.00848
	Std	0.0001	0.00456	0.00001	0.00516
UF2	Avg	0.00025	0.00122	0.0009	0.02904
	Std	0.00004	0.00009	0.00041	0.03831
UF3	Avg	0.01087	0.00838	0.00362	0.02894
	Std	0.00383	0.00527	0.00261	0.01882
UF4	Avg	0.00392	0.0042	0.00458	0.00469
	Std	0.00019	0.00009	0.00006	0.00053
UF5	Avg	0.00712	0.01524	0.01916	0.18134
	Std	0.00136	0.00306	0.0222	0.0199
UF6	Avg	0.00989	0.11417	0.00186	0.08056
	Std	0.00218	0.11223	0.00261	0.03933
UF7	Avg	0.00043	0.00154	0.00025	0.00777
	Std	0.00013	0.00065	0.00019	0.0023
UF8	Avg	0.00123	0.1102	0.08487	0.03095
	Std	0.00029	0.01945	0.102	0.01236
UF9	Avg	0.00465	0.10968	0.031	0.0389
	Std	0.00202	0.03856	0.00391	0.02745
F10	Avg	0.00179	1.22294	0.00092	0.40671
	Std	0.0008	0.05325	0.00019	0.09135
Friedman Test	Mean Rank	1.60	3.10	1.80	3.50
	Rank	1	3	2	4

جدول ۳ - نتایج الگوریتم‌ها (معیار IGD) روی توابع تست UF

		MOCCE	MOAHA	MOSMA	MOALO
UF1	Avg	0.01285	0.01211	0.04243	0.11467
	Std	0.00097	0.00167	0.00242	0.01209
UF2	Avg	0.00803	0.01267	0.02835	0.08776
	Std	0.00079	0.0013	0.00156	0.01969
UF3	Avg	0.10955	0.35799	0.08661	0.5993
	Std	0.01744	0.02487	0.0128	0.109
UF4	Avg	0.04011	0.03923	0.04434	0.08237
	Std	0.00182	0.00058	0.00078	0.02068
UF5	Avg	0.07002	0.16434	0.3908	1.12983
	Std	0.01133	0.01988	0.102	0.20405
UF6	Avg	0.10832	0.15177	0.11596	0.52478
	Std	0.06102	0.06641	0.0222	0.27023
UF7	Avg	0.02284	0.01106	0.01739	0.09658
	Std	0.0054	0.00057	0.00201	0.0158
UF8	Avg	0.13993	0.13497	0.29876	0.37075
	Std	0.01615	0.00631	0.0128	0.05629
UF9	Avg	0.06461	0.07765	0.25876	0.31445
	Std	0.01029	0.00745	0.00229	0.09327
UF10	Avg	0.12476	0.31462	0.39607	1.54821
	Std	0.01325	0.01619	0.0947	0.5343
Friedman Test	Mean Rank	1.60	1.80	2.60	4.00
	Rank	1	2	3	4

جدول ۴- نتایج الگوریتم‌ها (معیار MS) روی توابع تست UF

		MOCCE	MOAHA	MOSMA	MOALO
UF1	Avg	1.3499	0.48687	1.06433	1.19877
	Std	0.2113	0.31518	0.23150	0.20216
UF2	Avg	0.82817	0.25009	1.21619	1.64531
	Std	0.07934	0.03491	0.13330	0.16741
UF3	Avg	1.22379	1.44716	1.39777	1.12943
	Std	0.13266	0.31786	0.16015	0.11152
UF4	Avg	1.08845	0.27639	0.94874	1.02313
	Std	0.05889	0.04347	0.14161	0.14401
UF5	Avg	1.8917	1.76628	0.99517	1.21441
	Std	0.11811	0.05814	0.01936	0.15362
UF6	Avg	1.5454	1.56929	1.00158	1.04516
	Std	0.26393	0.25819	0.00160	0.01725
UF7	Avg	1.46824	0.32605	0.96642	1.39706
	Std	0.06049	0.0487	0.16564	0.16499
UF8	Avg	1.32235	1.00967	1.10090	1.31489
	Std	0.01616	0.30715	0.09679	0.11577
UF9	Avg	0.88352	1.001	1.11281	1.38943
	Std	0.06084	0.17068	0.05751	0.07445
UF10	Avg	1.41976	0.52542	1.07452	1.41131
	Std	0.07117	0.08173	0.06601	0.06711
Friedman Test	Mean Rank	3.20	1.90	2.10	2.80
	Rank	1	4	3	2

جدول۵ - نتایج الگوریتم‌ها (معیار GD) روی توابع تست IMOP

		MOCCE	MOAHA	MOSMA	MOALO
IMOP1	Avg	0.00001	0.00004	0.03404	0.00013
	Std	0.00001	0.00001	0.00125	0.00016
IMOP2	Avg	0.00001	0.00020	0.00515	0.00593
	Std	0.00000	0.00016	0.00534	0.00515
IMOP3	Avg	0.00185	0.00193	0.01237	0.00348
	Std	0.00173	0.00119	0.00107	0.00166
IMOP4	Avg	0.00002	0.00150	0.02871	0.03713
	Std	0.00001	0.00033	0.02031	0.02451
IMOP5	Avg	0.00035	0.00168	0.00386	0.00000
	Std	0.00001	0.00007	0.00080	0.00000
IMOP6	Avg	0.00040	0.00320	0.05475	0.01046
	Std	0.00001	0.00090	0.03897	0.00551
IMOP7	Avg	0.00232	0.00135	0.02358	0.01145
	Std	0.00458	0.00014	0.00507	0.01226
IMOP8	Avg	0.00678	0.01221	0.06104	0.03129
	Std	0.00085	0.00082	0.02814	0.01187
Friedman Test	Mean Rank	1.25	2.00	3.75	3.00
	Rank	1	2	4	3

جدول ۶ - نتایج الگوریتم‌ها (معیار IGD) روی توابع تست IMOP

		MOCCE	MOAHA	MOSMA	MOALO
IMOP1	Avg	0.57920	0.08472	0.49059	0.16354
	Std	0.10139	0.00260	0.10628	0.06290
IMOP2	Avg	0.34743	0.10976	0.29143	0.51038
	Std	0.12599	0.01392	0.05591	0.24841
IMOP3	Avg	0.12676	0.16682	0.50325	0.09305
	Std	0.04446	0.00313	0.11569	0.02492
IMOP4	Avg	0.47196	0.14407	0.48560	0.35652
	Std	0.17573	0.04762	0.09367	0.14569
IMOP5	Avg	0.40717	0.22467	0.56900	0.71298
	Std	0.04299	0.03041	0.02128	0.00000
IMOP6	Avg	0.28303	0.17931	0.54018	0.45419
	Std	0.03423	0.03763	0.05583	0.00791
IMOP7	Avg	0.60277	0.33191	0.52446	0.84290
	Std	0.17714	0.05369	0.02399	0.15706
IMOP8	Avg	0.24184	0.21809	0.71596	0.57729
	Std	0.02425	0.02806	0.04916	0.08753
Friedman Test	Mean Rank	2.63	1.25	3.25	2.88
	Rank	2	1	4	3

جدول ۷ - نتایج الگوریتم‌ها (معیار MS) روی توابع تست IMOP

		MOCCE	MOAHA	MOSMA	MOALO
IMOP1	Avg	1.06819	0.94579	1.42164	1.43902
	Std	0.05528	0.00897	0.10662	0.32829
IMOP2	Avg	1.26031	1.15801	1.56909	1.44119
	Std	0.38454	0.06535	0.28765	0.38553
IMOP3	Avg	1.40578	1.52993	1.74366	1.67845
	Std	0.13130	0.03319	0.12057	0.05808
IMOP4	Avg	1.57230	1.21758	1.54812	1.37886
	Std	0.41840	0.11060	0.20374	0.08276
IMOP5	Avg	1.02386	0.63850	0.96379	1.00000
	Std	0.04822	0.02501	0.07361	0.00000
IMOP6	Avg	1.18548	0.75281	1.10523	1.12528
	Std	0.10873	0.21984	0.38016	0.06513
IMOP7	Avg	1.21415	1.42318	1.49893	1.05289
	Std	0.27403	0.02658	0.17034	0.06492
IMOP8	Avg	1.01307	0.62668	0.98974	0.93459
	Std	0.06180	0.05114	0.17574	0.05497
Friedman Test	Mean Rank	2.88	1.38	3.13	2.63
	Rank	2	4	1	3

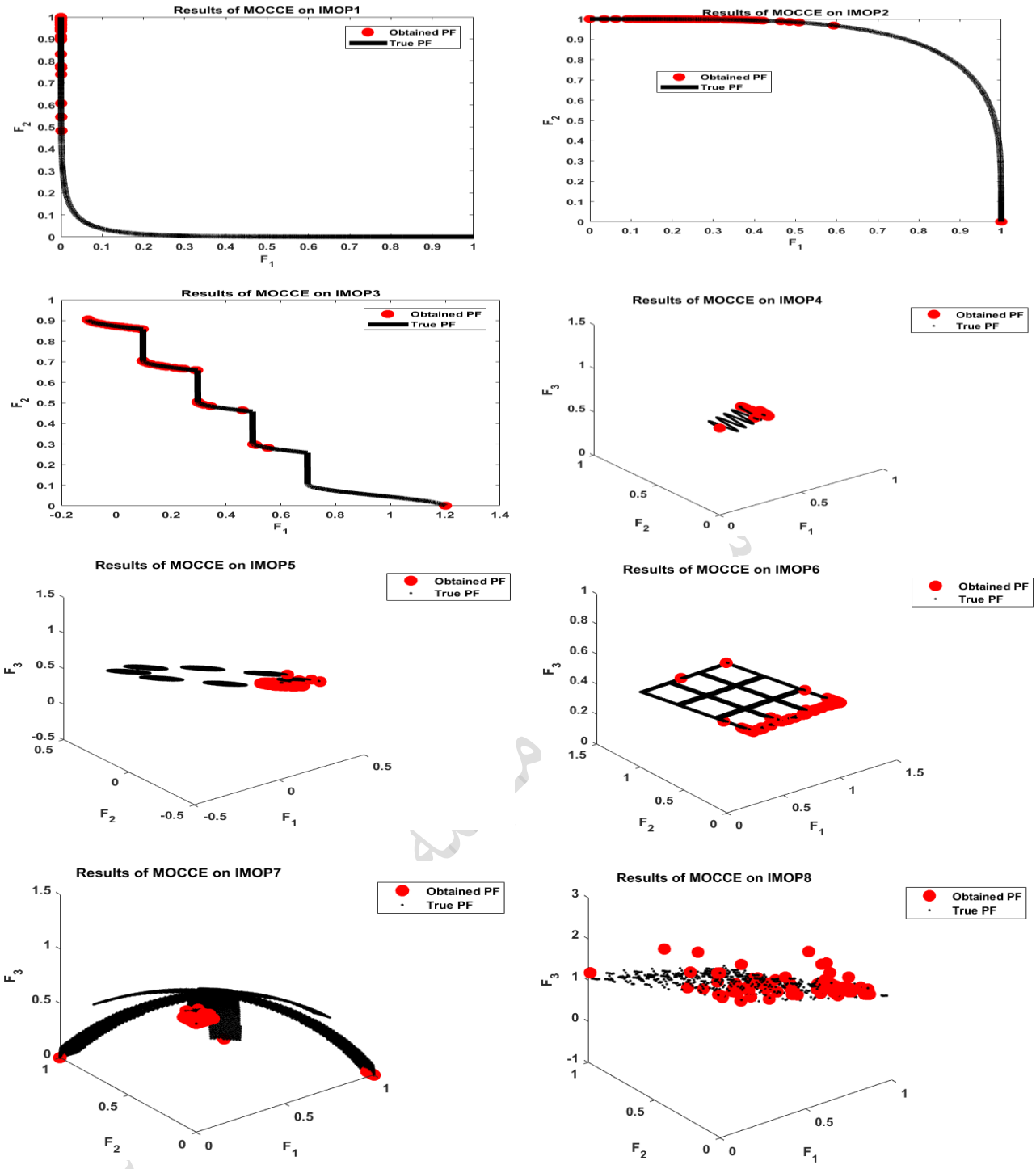
جدول ۸: نتایج آزمون رتبه علامت‌دار ویلکاکسون

	MOCCE – MOAHA	**MOCCE – MOSMA**	**MOCCE – MOALO**
R^-	28	33	37
R^+	26	21	17
$R^=$	0	0	0
Asymp. Sig.	$p > 0.05$	$p < 0.05$	$p < 0.05$

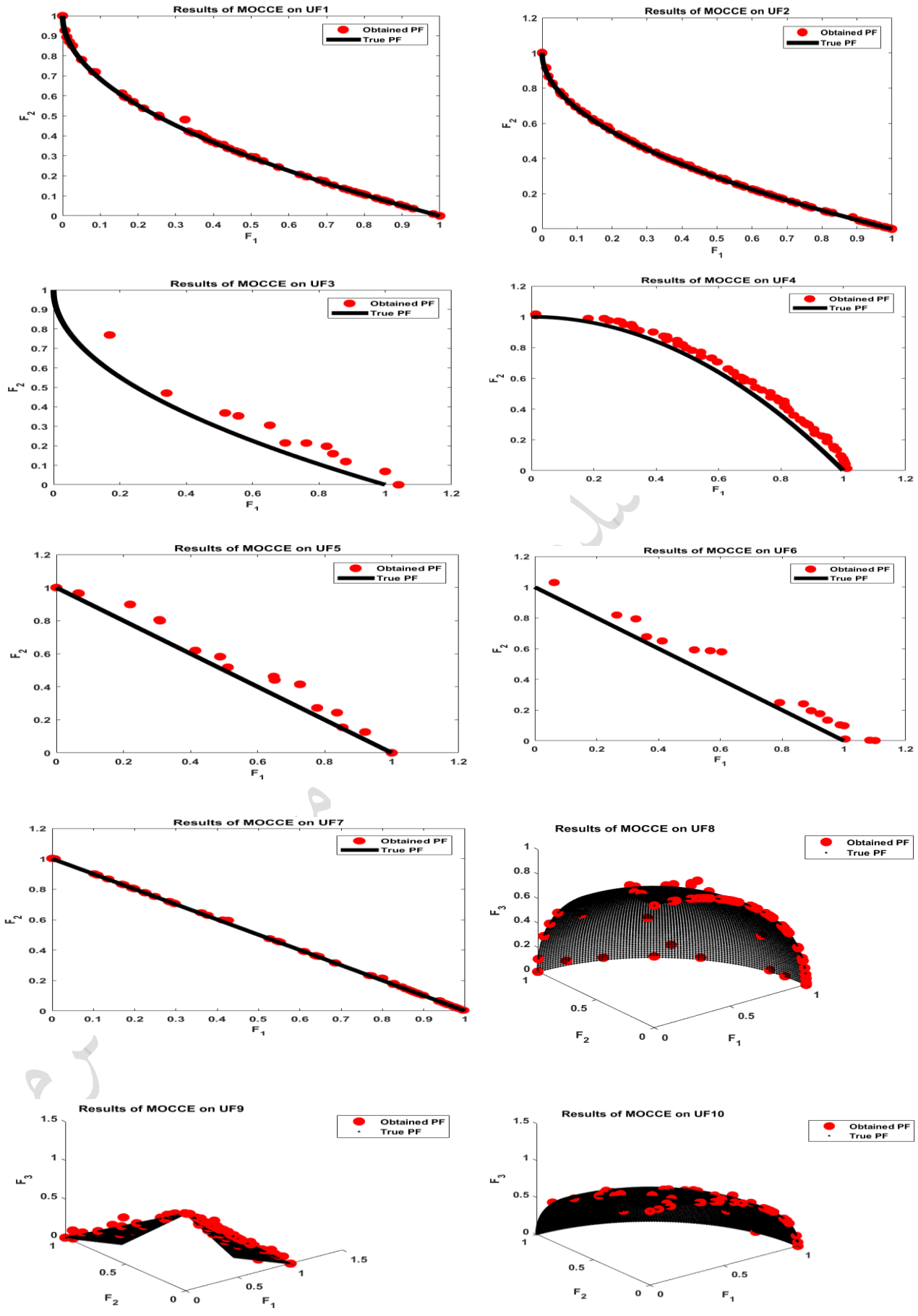
۵. نتیجه‌گیری و کارهای آینده

در این مقاله، نسخه‌ی چندهدفه‌ای از الگوریتم تکامل شوراهای شهر (CCE) با نام الگوریتم تکامل شوراهای شهر چندهدفه (MOCCE) ارائه شد. در این الگوریتم، یک آرشیو با اندازه ثابت برای ذخیره و بازیابی جواب‌های بهینه پرِتو در نظر گرفته می‌شود. همچنین، از این آرشیو برای تعریف ساختار هرم‌گونه شوراهای شهرها و شبیه‌سازی تکامل آن در فضاهای جستجوی چندهدفه استفاده کردیم. شرط ایجاد ساختار هرم‌گونه از راه‌حل‌های آرشیو و بقیه راه‌حل‌ها این است که مقدار هدف واحدی به‌ازای هر راه‌حل محاسبه شود که برای این منظور، مقدار هر تابع هدف را در یک وزنی (ضریبی) ضرب کرده و حاصل‌جمع آن‌ها بدست می‌آید.

برای ارزیابی کارایی الگوریتم ارائه شده، آن را روی ۱۸ تابع تست چندهدفه شناخته شده موسوم به UF و IMOP اجرا کرده و نتایج بدست آمده با نتایج الگوریتم‌های بهینه‌سازی شیر مورچه چندهدفه (MOALO)، کپک مخاطی چندهدفه (MOSMA) و مرغ مگس‌خوار مصنوعی چند هدفه (MOAHA) مقایسه شدند. نتایج آزمون میانگین رتبه‌ی فریدمن، کارایی بالای الگوریتم MOCCE را نسبت به الگوریتم‌های مذکور از نظر معیارهای فاصله نسلی (GD)، فاصله نسلی معکوس (IGD) و بیشنیه گستردگی (MS) تأیید می‌کنند. به کارگیری الگوریتم ارائه شده برای حل مسائل کاربردی و مهندسی و همچنین ارائه نسخه دودویی می‌توانند به‌عنوان کارهای آتی در نظر گرفته شوند.



شکل ۴- مجموعه بهینه پرِتو بدست آمده از الگوریتم MOCCE روی توابع تست UF



شکل ۵- مجموعه بهینه پرِتو بدست آمده با الگوریتم MOCCE روی توابع تست IMOP

۶. مراجع

[۱] ح. شفیعی، و. رافع، م. امیری، "بهینه‌سازی مسیریابی وسایل نقلیه مبتنی بر ترکیب الگوریتم‌های کلونی مورچه و ازدحام ذرات با تابع ابتکاری کسینوس زوایا"، *مجله محاسبات نرم*، ۱۴۰۲، doi: 10.22052/scj.2023.248702.1118.

[۲] S. Mirjalili, A. H. Gandomi, S. Z .Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems," *Advances in engineering software*, vol. 114, pp. 163-191, 2017, doi:https://doi.org/10.1016/j.advengsoft.2017.07.002.

[۳] U. Yüzgeç and M. Kusoglu, "Multi-objective harris hawks optimizer for multiobjective optimization problems," *BSEU Journal of Engineering Research and Technology*, vol. 1, no. 1, pp. 31-41, 2020.

[۴] B. Cao, M. Li, X. Liu, J. Zhao, W. Cao, and Z. Lv, "Many-objective deployment optimization for a drone-assisted camera network," *IEEE transactions on network science and engineering*, vol. 8, no. 4, pp. 2756-2764, 2021, doi: https://doi.org/10.1109/TNSE.2021.3057915.

[۵] J. Branke, T. Kaußler, and H. Schmeck , "Guidance in evolutionary multi-objective optimization," *Advances in engineering software*, vol. 32, no. 6, pp. 499-507, 2001, doi: https://doi.org/10.1016/S0965-9978(00)00110-1.

[۶] م. زمزمه، س. صدیقیان کاشی، ا. نیک‌انجام، "بازآرایی تکاملی چندهدفه آگاه از انرژی برای تصحیح نشانه‌های کد بد در کاربردهای اندرویدی،" *مجله محاسبات نرم*، ۱۴۰۲, doi: 10.22052/SCJ.2023.246479.1074.

[۷] C. Lin, F. Gao, and Y. Bai, "An intelligent sampling approach for metamodel-based multi-objective optimization with guidance of the adaptive weighted-sum method," *Structural and Multidisciplinary Optimization*, vol. 57, pp. 1047-1060, 2018, doi: https://doi.org/10.1007/s00158-017-1793-2.

[۸] J. Behnamian, M. Zandieh, and S. Fatemi Ghomi, "Bi-objective parallel machines scheduling with sequence-dependent setup times using hybrid metaheuristics and weighted min–max technique," *Soft Computing*, vol. 15, pp. 1313-1331, 2011, doi: https://doi.org/10.1007/s00500-010-0673-0.

[۹] R. W. Hanks, B. J. Lunday, and J. D. Weir, "Robust goal programming for multi-objective optimization of data-driven problems: A use case for the United States transportation command's liner rate setting problem," *Omega*, vol. 90, p. 101983, 2020, doi: https://doi.org/10.1016/j.omega.2018.10.013.

[۱۰] L. Lai, L. Fiaschi, M. Cococcioni, and K. Deb, "Solving mixed pareto-lexicographic multiobjective optimization problems: the case of priority levels," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 5, pp. 971-985, 2021, doi: https://doi.org/10.1109/TEVC.2021.3068816.

[۱۱] J. Zheng, Z. Zhang, J. Zou, S. Yang, J. Ou, and Y. Hu, "A dynamic multi-objective particle swarm optimization algorithm based on adversarial decomposition and neighborhood evolution," *Swarm and Evolutionary Computation*, vol. 69, p. 100987, 2022, doi: https://doi.org/10.1016/j.swevo.2021.100987.

[۱۲] S. Mirjalili, S. Saremi, S. M. Mirjalili, and L. d. S. Coelho, "Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization," *Expert systems with applications*, vol. 47, pp. 106-119 ,2016 ,doi: https://doi.org/10.1016/j.eswa.2015.10.039.

[۱۳] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary computation*, vol. 2, no. 3, pp. 221-248, 1994, doi: https://doi.org/10.11/٦٢evco.1994.2.3.221.

[۱۴] C. C. Coello and M. S. Lechuga, "MOPSO: A proposal for multiple objective particle swarm optimization," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, 2002, vol. 2: IEEE, pp. 1051-1056 ,doi: https://doi.org/10.1109/CEC.2002.1004388 .

[۱۵] ج. سلیمی، س. گلی بیدگلی، "ارائه یک الگوریتم ترکیبی با استفاده از الگوریتم کرم شب‌تاب، الگوریتم ژنتیک و جست‌وجوی محلی"، *مجله محاسبات نرم*، جلد ۸، شماره ۱، ص ۱۴–۲۸، ۱۳۹۸.

[۱۶] T. Murata and H. Ishibuchi, "MOGA: multi-objective genetic algorithms," in *IEEE international conference on evolutionary*

computation, 1995, vol. 1: IEEE Piscataway, NJ, USA, pp. 289-294.

[۱۷] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Parallel Problem Solving from Nature PPSN VI: 6th International Conference Paris, France, September 18–20, 2000 Proceedings 6*, 2000: Springer, pp. 849-858, doi: https://doi.org/10.1007/3-540-45356-3_83.

[۱۸] J. Horn ,N. Nafpliotis, and D. E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence*, 1994: Ieee, pp. 82-87, doi: https://doi.org/10.1109/ICEC.1994.350037.

[۱۹] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," *TIK report*, vol. 103, 2001, doi: doi.org/10.3929/ethz-a-004284029.

[۲۰] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen" ,Harris hawks optimization: Algorithm and applications," *Future generation computer systems*, vol. 97, pp. 849-872, 2019, doi: https://doi.org/10.1016/j.future.2019.02.028.

[۲۱] H. A. Abbass, R. Sarker, and C. Newton, "PDE: a Pareto-frontier differential evolution approach for multi-objective optimization problems," in *Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546)*, 2001, vol. 2: IEEE, pp. 971-978, doi: https://doi.org/10.1109/CEC.2001.934295 .

[۲۲] S.-Y. Zeng *et al*" ,.A dynamic multi-objective evolutionary algorithm based on an orthogonal design," in *2006 IEEE International Conference on Evolutionary Computation*, 2006: IEEE, pp. 573-580, doi: https://doi.org/10.1109/CEC.2006.1688361 .

[۲۳] K. Zhong, G. Zhou, W. Deng ,Y. Zhou, and Q. Luo, "MOMPA: Multi-objective marine predator algorithm," *Computer Methods in Applied Mechanics and Engineering*, vol. 385, p. 114029, 2021, doi: https://doi.org/10.1016/j.cma.2021.114029

[۲۴] S. Mirjalili, P. Jangir, and S. Saremi, "Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems," *Applied Intelligence*, vol. 46, pp. 79-95, 2017, doi: https://doi.org/10.1007/s10489-016-0825-8.

[۲۵] N. Khodadadi, S. M. Mirjalili, W. Zhao, Z. Zhang, L. Wang, and S. Mirjalili, "Multi-objective artificial hummingbird algorithm," in *Advances in Swarm Intelligence: Variations and Adaptations for Optimization Problems*: Springer, 2022, pp. 407-41, doi: https://doi.org/10.1007/978-3-031-09835-2_22

[۲۶] A. Sadollah, H. Eskandar, A. Bahreininejad, and J. H. Kim, "Water cycle algorithm for solving multi-objective optimization problems," *Soft Computing*, vol. 19, pp. 2587-2603, 2015, doi: https://doi.org/10.1007/s00500-014-1424-4.

[۲۷] H. R. Hassanzadeh and M. Rouhani, "A multi-objective gravitational search algorithm," in *2010 2nd international conference on computational intelligence, communication systems and networks*, 2010: IEEE, pp. 7-12, doi: https://doi.org/10.1109/CICSyN.2010.32.

[۲۸] J. J. Flores, R. López, and J. Barrera, "Gravitational interactions optimization," in *International Conference on Learning and Intelligent Optimization*, 2011: Springer, pp. 226-237, doi: https://doi.org/10.1007/978-3-642-25566-3_17.

[۲۹] M. Premkumar, P .Jangir, and R. Sowmya, "MOGBO: A new Multiobjective Gradient-Based Optimizer for real-world structural optimization problems," *Knowledge-Based Systems*, vol. 218, p. 106856, 2021, doi: https://doi.org/10.1016/j.knosys.2021.106856.

[۳۰] M. Premkumar, P. Jangir, R. Sowmya, H. H. Alhelou, S. Mirjalili, and B .S. Kumar, "Multi-objective equilibrium optimizer: Framework and development for solving multi-objective optimization problems," *Journal of Computational Design and Engineering*, vol. 9, no. 1, pp. 24-50, 2022, doi: https://doi.org/10.1093/jcde/qwab065.

[۳۱] F. Cao, Z. Tang, C. Zhu, and X. Zhao, "An Efficient Hybrid Multi-Objective Optimization Method Coupling Global Evolutionary and Local Gradient Searches for Solving Aerodynamic Optimization Problems," *Mathematics*, vol. 11, no. 18, p. 3844, 2023, doi: https://doi.org/10.3390/math11183844.

[۳۲] N. Khodadadi, M. Azizi, S. Talatahari, and P . Sareh, "Multi-objective crystal structure algorithm (MOCryStAl): Introduction and performance evaluation," *IEEE Access*, vol. 9, pp. 117795-117812, 2021, doi: https://doi.org/10.1109/ACCESS.2021.3106487.

[۳۳] S. Yacoubi, G. Manita, H. Amdouni, S. Mirjalili, and O. Korbaa, "A modified multi-objective slime mould algorithm with orthogonal learning for numerical association rules mining," *Neural Computing and Applications*, vol. 35, no. 8, pp. 6125-6151, 2023, doi: https://doi.org/10.1007/s00521-022-07985-w.

[۳۴] F. Zou, L. Wang, X. Hei, D. Chen, and B. Wang, "Multi-objective optimization using teaching-learning-based optimization algorithm," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 4, pp. 1291-1300, 2013, doi: https://doi.org/10.1016/j.engappai.2012.11.006.

[۳۵] R. V. Rao, V. J. Savsani, and D. Vakharia, "Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems," *Computer-aided design*, vol. 43, no. 3, pp. 303-315, 2011, doi: https://doi.org/10.1016/j.cad.2010.12.015.

[۳۶] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67-82, 1997, doi: https://doi.org/10.1109/4235.585893.

[۳۷] E. Pira, "City councils evolution: A socio-inspired metaheuristic optimization algorithm," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1-50, 2022, doi: https://doi.org/10.1007/s12652-022-03765-5.

[۳۸] J. Cao, J. Zhang, F. Zhao, and Z. Chen, "A two-stage evolutionary strategy based MOEA/D to multi-objective problems," *Expert Systems with Applications*, vol. 185, p. 115654, 2021, doi: https://doi.org/10.1016/j.eswa.2021.115654.

[۳۹] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm research: A history and analysis," Citeseer, 1998 .

[۴۰] M. R. Sierra and C .A. Coello Coello, "Improving PSO-based multi-objective optimization using crowding, mutation and∈-dominance," in *Evolutionary Multi-Criterion Optimization: Third International Conference, EMO 2005, Guanajuato, Mexico, March 9-11, 2005. Proceedings 3*, 2005 :Springer, pp. 505-519, doi: https://doi.org/10.1007/978-3-540-31880-4_35.

[۴۱] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86-92, 1940.

[۴۲] R. F. Woolson, "Wilcoxon signed- rank test," *Wiley encyclopedia of clinical trials*, pp. 1-3, 2007, doi: https://doi.org/10.1002/9780471462422.eoct979.

پیوست الف-۱ خلاصه‌ای از جزئیات الگوریتم‌ها از جمله تکنیک مورد استفاده، سال انتشار، معیارهای کارایی، مزایا و معایب

الگوریتم	سال انتشار	تکنیک مورد استفاده	مسائل موردنظر	معیارهای ارزیابی کارایی	مزایا	معایب
بهینه‌سازی ازدحام ذرات چند راهنما (MGPSO) [۱۲]	۲۰۲۳	اِعمال رویکرد تکاملی مشارکتی برای بهینه‌سازی ازدحام ذرات چند راهنما (MGPSO) برای مسائل بهینه‌سازی چندهدفه در مقیاس بزرگ	مقیاس‌بندی MGPSO برای مدیریت موثر مسائل بهینه‌سازی چندهدفه در مقیاس بزرگ	IGD,HV	۱-بهبود مقیاس‌پذیری MGPSO برای مسائل در مقیاس بزرگ. ۲-عملکرد رقابتی برای مسائل قابل تفکیک و تک‌وجهی. ۳-عملکرد رقابتی برای مسائل پویای چندهدفه.	۱-کاهش کارایی در مسائل چندوجهی، غیرقابل تفکیک، فریبنده یا جانب‌دارانه. ۲-صرف محاسبات زیاد در مسائل چندهدفه.
الگوریتم ترکیبی چندهدفه مبتنی بر گرادیان به کمک	۲۰۲۳	چارچوب الگوریتم ترکیبی موازی دولایه ترکیبی از جستجوی محلی چندهدفه و	حل مسائل بهینه‌سازی آیرودینامیکی در مقیاس بزرگ با اهداف	IGD, HV	۱-مدیریت مؤثر مسائل چندهدفه (MOPs) با حداکثر ۱۰۰۰ متغیر	۱-انتخاب مدل جایگزین مناسب همچنان یک چالش است

جایگزین (GS-MOHA) [۲۵]		مکانیسم‌های تکامل سراسری	متعدد و ارزیابی هزینه‌های بالا، موازنه کارایی بهینه‌سازی و دقت همگرایی در فضاهای طراحی با ابعاد بالا		تصمیم ۲-افزایش ۵ تا ۱۰ برابری بهره‌وری بهینه‌سازی در مقایسه با الگوریتم‌های چندهدفه موجود ۳-مواجهه مؤثر با مسائل چندهدفه پرهزینه در ابعاد بالا ۴-استفاده کارآمد از اطلاعات گرادیان برای بهبود دقت بهینه‌سازی ۵-مناسب برای بهینه‌سازی شکل آیرودینامیکی با ابعاد بالا	۲- زمان آموزش برای مدل‌های جایگزین ممکن است در موارد بسیاری با ابعاد بالا زیاد باشد ۳-مدیریت مدل‌های جایگزین و انتخاب نقاط به‌روزرسانی می‌تواند پیچیده باشد.
الگوریتم بهینه‌ساز چندهدفه شامپانزه (MOCHOA) [۲۳]	۲۰۲۳	الگوریتم بهینه‌سازی شامپانزه چندهدفه (MOChOA)، بر اساس الگوریتم بهینه‌ساز شامپانزه (ChOA).	مسائل بهینه‌سازی چندهدفه (MOO) در مهندسی و زمینه‌های مختلف که در آن چندین هدف باید به طور همزمان بهینه شوند.	CPF, GD, IGD, S, Delta-P, HA, PD, S	۱-ساختار بایگانی کارآمد برای ذخیره راه‌حل‌های بهینه پرتو. ۲-استراتژی انتخاب رهبر مؤثر. ۳-مکانیزم شبکه پویا برای مدیریت بایگانی. ۴-تضمین کاوش در فضای جستجو ۵-اجتناب از بهینه محلی با استفاده از نقشه‌های آشفته. ۶-ساختار موازی برای اجرای آسان. ۷- پارامترهای معدود قابل تنظیم. ۸-عملکرد رقابتی بالا. ۹. مناسب برای برنامه‌های مختلف بهینه‌سازی چندهدفه.	۱-عدم بحث در مورد نسخه‌های دودویی یا مختلط اعداد صحیح ۲-عدم اشاره به مطالعات موردی کاربردی خاص۳-ارزیابی محدود به مجموعه‌ای از مسائل معیار
الگوریتم قالب متعامد چندهدفه (MOOSMA) [۳۴]	۲۰۲۳	الگوریتم قالب لجن چندهدفه (MOOSMA) با قاعده‌کاوی انجمنی عددی (NARM) و ادغام. یادگیری متعامد برای بهینه‌سازی در MOSMA	۱-قاعده‌کاوی انجمنی عددی که یک مسئله‌ی چالش برانگیز در داده‌کاوی است. ۲- بهینه‌سازی چندهدفه برای ARM با معیارهای کارایی	GD, IGD, SP, HV, S, Delta-P	۱-الگوریتم بهبودیافته (MOOSMA) از نظر معیارهای مختلف عملکرد (Dp، IGD، HV) نتایج بهتری نسبت به MOSMA دارد ۲-عملکرد بهتری از سایر الگوریتم‌ها از نظر میانگین lift و پشتیبانی در NARM دارد	۱-نیاز به تخصص در پیاده‌سازی و تنظیم دقیق الگوریتم. ۲. وجود محدودیت‌های بالاقوه با توجه به اشاره به کارهای آینده برای پیشرفت‌های بیشتر
مرغ مگس‌خوار مصنوعی چندهدفه (MOAHA) [۲۲]	۲۰۲۲	الگوریتم مرغ مگس‌خوار مصنوعی چندهدفه (MOAHA)، توسعه الگوریتم مرغ مگس‌خوار مصنوعی (AHA)	بهینه‌سازی مسائل مهندسی چندهدفه، مقابله با محدودیت‌ها، مدیریت توابع هدف محاسباتی پرهزینه و پویا.	GD, IGD, S, MS	۱-بهینه‌سازی موثر مسائل چندهدفه که شامل انتخاب رهبر، مکانیسم شبکه و مکانیسم بایگانی است. ۲-عملکرد بهتر نسبت به سایر الگوریتم‌ها (MOPSO، MOWOA، MOHHO) در معیارهای مختلف ۳-مناسب برای رسیدگی به مسائل بیوانفورماتیک و تکنیک‌های خوشه‌بندی داده‌ها.	۱-طبیعت تصادفی می‌تواند نتایج را غیرقابل پیش‌بینی کند. ۲-ممکن است نیاز به تنظیم برای دامنه‌های مسائل خاص داشته باشد. ۳-اثربخشی ممکن است بر اساس پیچیدگی مسئله متفاوت باشد.

الگوریتم بهینه‌ساز تعادل چندهدفه (MOEO) [۳۱]	۲۰۲۱	الگوریتم بهینه‌ساز تعادل چندهدفه (MOEO)، ادغام مکانیسم‌های مرتب‌سازی غیرغالب (NDS) و فاصله ازدحام (CD) با یک بهینه‌ساز تعادل تک‌هدفه (EO) با الهام از فیزیک.	حل مسائل بهینه‌سازی چندهدفه، شامل مسائل محدود و غیرمحدود و همچنین مسائل طراحی مهندسی در دنیای واقعی.	SP, P, HV, CPF, PD, Delta-P, DM	۱-همگرایی با سرعت بالا به دلیل توانایی بهره‌برداری قوی. ۲-توانایی اکتشاف بالا با مدیریت کارآمد عامل. ۳-بهبود همگرایی و تنوع بر مبنای تطبیقی. ۴-توزیع یکنواخت راه‌حل‌های غیرغالب پرتو.	۱-بحث محدود در مورد رویکردهای مدیریت محدودیت برای MOEO. ۲-تاکید بر کاربرد در بهینه‌سازی دو و سه‌هدفه.
الگوریتم بهینه‌ساز چندهدفه مبتنی بر گرادیان (MOGBO) [۳۰]	۲۰۲۱	الگوریتم MOGBO (بهینه‌ساز گرادیان چندهدفه) بهینه‌سازی مبتنی بر گرادیان، مرتب‌سازی غیرغالب نخبه‌گرا و مکانیسم‌های فاصله ازدحام.	مسائل بهینه‌سازی چندهدفه، به‌ویژه در بهینه‌سازی سازه، مسائل محک بدون محدودیت و مسائل طراحی ساختاری با محدودیت.	SP, P, HV, CPF, PD, Delta-P, DM	۱-مؤثر در حل مسائل بهینه‌سازی دنیای واقعی در مقیاس بزرگ. ۲-برای همگرایی و تنوع بهتر از جستجوی گرادیان، مرتب‌سازی غیرغالب و فاصله ازدحام استفاده می‌کند. ۳-عملکرد بهتری از سایر الگوریتم‌های رقابتی از نظر پوشش، همگرایی و تنوع دارد. ۴-مناسب برای کاربردهای مختلف مهندسی فراتر از بهینه‌سازی سازه.	۱-زمان محاسبات بالاتر در مقایسه با برخی از الگوریتم‌های دیگر. ۲-ممکن است برای مسائل بهینه‌سازی چندهدفه با تعداد اهداف بسیار زیاد کارآمد نباشد.
الگوریتم بهینه‌ساز چندهدفه شکارچیان دریایی (MOMPA) [۲۴]	۲۰۲۱	الگوریتم شکارچی دریایی چندهدفه (MOMPA) بر اساس مکانیسم‌های مرتب‌سازی غیرغالب نخبه‌گرا و فاصله ازدحام با الهام از تعاملات شکارچی-شکار در طبیعت.	حل مسائل بهینه‌سازی چندهدفه با اهداف متناقض، از جمله مسائل طراحی نامحدود، محدود و مهندسی.	SP, GD, IGD, DM, CPF, S	۱-نرخ همگرایی سریع با دقت جواب خوب. ۲-قابلیت اکتشاف بالا. ۳-مدیریت مؤثر انواع مختلف مسائل چندهدفه. ۴-در بیشتر موارد از سایر الگوریتم‌های پیشرفته برتری دارد. ۵-مناسب برای مسائل پیچیده طراحی مهندسی در دنیای واقعی.	۱-نیاز به تنظیم فراپارامترها از طریق آزمون و خطا دارد. ۲-توصیه‌های تنظیم خاص ممکن است بسته به مسئله متفاوت باشد.
بهینه‌ساز چندهدفه شاهین هریس (MOHHO) [۲]	۲۰۲۰	بهینه‌سازی چندهدفه شاهین هریس (MOHHO)، توسعه‌ای از الگوریتم بهینه‌ساز شاهین هریس (HHO)، که شامل یک مخزن بایگانی برای نتایج بهینه پرتو است.	مسائل بهینه‌سازی چند هدفه، به ویژه توابع تست بدون محدودیت (معیارهای ZDT).	IGD	۱-بهبود رفتار همگرایی که از طریق متریک فاصله نسلی معکوس (IGD) نشان داده شده است. ۲-عملکرد بهتر نسبت به الگوریتم‌های MOALO و MODA از نظر متریک IGD برای توابع معیار ZDT. ۳-مکانیسم‌های اکتشاف و بهره‌برداری مؤثر در الگوریتم MOHHO.	۱-بحث محدود در مورد برنامه‌های کاربردی دنیای واقعی و حوزه‌های مسئله‌ی خارج از توابع تست. ۲-عدم مقایسه گسترده با طیف وسیع‌تری از الگوریتم‌های بهینه‌سازی چندهدفه.

					۴-نتایج زمان محاسباتی رقابتی.	
الگوریتم بهینه‌ساز چندهدفه گرگ - خاکستری (MOGWO) [۱۰]	۲۰۱۶	بهینه‌ساز گرگ خاکستری چندهدفه (MOGWO)، نوعی از بهینه‌ساز گرگ خاکستری (GWO) که برای بهینه‌سازی چندمعیاره اقتباس شده است.	مسائل بهینه‌سازی چندمعیاره، با تمرکز بر مهندسی دنیای واقعی و کاربردهای تجاری.	IGD, SP, MS	۱-با موفقیت برای انواع مختلفی از مشکلات دنیای واقعی، از جمله سناریوهای مهندسی و تجاری اعمال شده است. ۲-از غلبه پرتو و فاصله ازدحام برای حفظ مجموعه‌ای متنوع از جواب‌ها استفاده می‌کند. ۳-به طور مؤثر به جواب‌های پرتو بهینه، که از طریق آزمون‌های معیار نشان داده شده است، همگرا می‌شود.	۱-بحث محدود در مورد حساسیت الگوریتم به تنظیم پارامتر. ۲-اکتشاف محدود برنامه‌های کاربردی در حوزه‌های خارج از مهندسی و تجارت. ۳-عدم تجزیه‌وتحلیل گسترده در مورد مقیاس‌پذیری الگوریتم برای مسائل با ابعاد بالا.