مجله محاسبات نرم، جلد 13، شماره 2 (پاییز و زمستان 1403)، صفحه 56ـ77
https://doi.org/10.22052/scj.2024.253070.1154

مجله محاسبات نرم
SOFT COMPUTING JOURNAL
تارنمای مجله: scj.kashanu.ac.ir

University of Kashan

توسعه الگوریتم تکامل شوراهای شهر برای حل مسائل بهینه‌سازی چندهدفه✧

مهدیه غفار علیشاهی[1]، دانشجوی کارشناسی ارشد، عین‌الله پیرا[1*]، استادیار، علیرضا روحی[1]، استادیار

[1] دانشکده فناوری اطلاعات و مهندسی کامپیوتر، دانشگاه شهید مدنی آذربایجان، تبریز، ایران.

اطلاعات مقاله

تاریخچه مقاله:
دریافت 17 خرداد ماه 1402
پذیرش 24 آبان ماه 1402

کلمات کلیدی:
الگوریتم‌های فراابتکاری
بهینه‌سازی
چندهدفه
تکامل شورای شهر
جبهه پرِتو

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 به نسبت 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 18 تابع آزمون 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 رتبه فریدمن، در همه توابع آزمون 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 آزمون IMOP از لحاظ معیار GD و دومین رتبه را از لحاظ معیارهای IGD و MS به خود اختصاص می‌دهد.

1. مقدمه

بهینه‌سازی، فرآیندی است که در آن بهترین جواب (با توجه به مجموعه‌ای از معیارها) از میان مجموعه‌ای از جواب‌های ممکن برای یک مساله خاص انتخاب می‌شود [1]. هدف از حل مسائل بهینه‌سازی تک‌هدفه، یافتن بهترین جوابی است که شرایط مساله از جمله بیشینه‌سازی یا کمینه‌سازی تابع هدف و حفظ محدودیت‌های از پیش تعریف شده را برآورده کند [2]، در حالی که در مسائل بهینه‌سازی چندهدفه، هدف، کمینه‌سازی یا بیشینه‌سازی چندین تابع هدف با لحاظ محدودیت‌های متعدد است [3]. در این گونه مسائل، باید مجموعه‌ای از جواب‌ها که اهداف مدنظر را در سطح قابل قبولی برآورده می‌سازند، به‌عنوان

✧ نوع مقاله: پژوهشی

* نویسنده مسئول
پست(های) الکترونیک: mahdiehghafar@gmail.com (غفار علیشاهی)
pira@azaruniv.ac.ir (پیرا)
rouhi@azaruniv.ac.ir (روحی)

نحوه ارجاع به مقاله: مهدیه غفار علیشاهی، عین‌الله پیرا، علیرضا روحی، «توسعه الگوریتم تکامل شوراهای شهر برای حل مسائل بهینه‌سازی چندهدفه»، مجله محاسبات نرم، جلد 13، شماره 2، ص56ـ77، پاییز و زمستان 1403.

مجموعه جواب بهینه معرفی شوند. این مجموعه جواب بهینه که مجموعه پرِتو[1] نامیده می‌شود، شامل جواب‌هایی است که با همدیگر مصالحه[2] دارند [4]. به‌عبارت‌دیگر، مجموعه پرِتو شامل همه جواب‌های مناسبی است که نسبت به همدیگر هیچ‌گونه برتری ندارند. دو روش پیشینی[3] و پسینی[4] برای حل این قبیل مسائل وجود دارند [5]. در روش‌های پیشینی، مساله بهینه‌سازی چندهدفه با چندین تابع هدف به یک مساله بهینه‌سازی تک هدفه 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 [6]. مجموع وزن‌دار[5] [7]، بیشینه-کمینه وزن‌دار[6] [7]، برنامه‌نویسی هدف[7] [8] و لکسیکوگراف[8] [9]، از جمله روش‌های پیشینی به حساب می‌آیند. عدم وجود اطلاعات در مورد میزان اهمیت هر کدام از اهداف در بعضی از مسائل بهینه‌سازی از جمله محدودیت‌های روش‌های پیشینی محسوب می‌شود. علاوه بر این، با توجه به اینکه تنها یک راه‌حل به‌عنوان جواب تولید می‌شود طراح هیچ بینشی نسبت به مصالحه 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) [10]، سعی در بهبود اِشکالات روش‌های پیشینی دارند. این الگوریتم به صورت تدریجی وزن‌ها را تغییر داده و برنامه را برای هر مجموعه اجرا می‌کند تا نیاز قبلی به اطلاعات ترجیحی مساله را رفع کند. اگرچه پیاده‌سازی این روش ساده است ولی نمی‌تواند دغدغه مهم دیگر مربوط به عدم توانایی دستیابی به جبهه بهینه پرِتو را برطرف کند. برعکس، در روش‌های پسینی، توابع هدف با هم ترکیب نشده و ماهیت چندهدفه بودن مساله حفظ می‌شود تا امکان بررسی رفتار مساله در طیفی از پارامترهای طراحی و شرایط عملیاتی فراهم شود [11]. واضح است که در روش‌های پسینی، برخلاف پیشینی، با چندین جواب بهینه مواجه خواهیم شد که یکی از آنها توسط طراح انتخاب می‌شود. الگوریتم‌های بهینه‌سازی چند هدفه ژنتیک با مرتب‌سازی نامغلوب (NSGA) [12] و ازدحام ذرات (MOPSO) [13]، 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. نخبه‌گرایی نیز تضمین می‌کند که جواب‌های نامغلوب در طی فرآیند تکامل حفظ خواهند شد [14]. اکثر الگوریتم‌های بهینه‌سازی فراابتکاری 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 فرآیند تکامل بیولوژیکی نشات می‌گیرند و به صورت مکرر از عملگرهای بیولوژیکی مثل انتخاب، تولیدمثل و جهش استفاده می‌کنند تا جمعیت ابتدایی را در جهت رسیدن به مجموعه بهینه پرِتو تکامل ببخشند [15]. الگوریتم ژنتیک چندهدفه (MOGA) [16]، یکی از مشهورترین الگوریتم‌ها در این گروه است که از اصل تکاملی داروین (اصل بقای شایسته‌ترین‌ها) الهام می‌گیرد. این الگوریتم در واقع بیانگر نظریه 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 [17]، 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

[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

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 [17]، NPGA [18]، PDE [19] و DOMOEA [18] 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 فرآیند تکامل می‌باشند. بهینه‌سازی چندهدفه ازدحام ذرات (MOPSO) [14]، یکی از مشهورترین الگوریتم‌ها در این گروه است که از رفتار دسته‌جمعی پرندگان برای جستجوی غذا الهام گرفته شده است. ایده اصلی الگوریتم پرندگان، دنبال کردن 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 مولفه 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) [3]، الگوریتم دیگری در این گروه است که از رفتار غافلگیرانه 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) [20]، 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) [21]، الگوریتم بهینه‌ساز چندهدفه مرغ مگس‌خوار (MOAHA) [22]، الگوریتم بهینه‌ساز چندهدفه شامپانزه (MOCHOA) [20]، الگوریتم بهینه‌ساز چندهدفه گرگ خاکستری (MOGWO) [12]، الگوریتم بهینه‌ساز چندهدفه شکارچیان دریایی (MOMPA) [23]، بهینه‌سازی ازدحام ذرات چند راهنما (MGPSO) [24]، بهینه‌ساز شامپانزه کوتوله چندهدفه (MOBO) [25] 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 فرآیند چرخه آب در طبیعت الگو گرفته است [26]. این الگوریتم در فرار از بهینگی محلی و سرعت زیاد در رسیدن به جبهه پرِتو توانمندی بالایی دارد. الگوریتم جستجوی گرانشی چندهدفه (MOGSA) [27]، 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 [28]، هر راه‌حل به صورت یک شی در نظر گرفته می‌شود که جِرم آن، میزان برازندگی آن را نشان می‌دهد. در طی تکرارهای الگوریتم، نیروی گرانشی مابین اشیا باعث جذب اشیا سبک‌تر به سمت اشیا سنگین‌تر می‌شوند که این موضوع باعث می‌شود در نهایت شی با بیشترین جِرم یا برازنده‌ترین راه‌حل باقی بماند. الگوریتم بهینه‌ساز چندهدفه مبتنی بر گرادیان (MOGBO) [29]، الگوریتم بهینه‌ساز تعادل چندهدفه (MOEO) [30]، الگوریتم ترکیبی چندهدفه مبتنی بر گرادیان به کمک جانشین (GS-MOHA) [31] و الگوریتم بهینه‌سازی ساختار کریستال چندهدفه (MOCryStAl) [32] و الگوریتم قالب متعامد چندهدفه (MOOSMA) [33]، 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) [34]، یکی از الگوریتم‌های مشهور در این گروه است که فرآیند دو مرحله‌ای یادگیری و آموزش در یک کلاس درسی را شبیه‌سازی می‌کند [35]. در مرحله آموزش، بهترین فرد کلاس به عنوان

[1] Teaching-learning-based optimization

معلم انتخاب شده و شایستگی بقیه افراد جمعیت را ارتقا می‌دهد در حالی که در مرحله 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. جدول (1) خلاصه‌ای از جزئیات الگوریتم‌ها از جمله رویکرد مورد استفاده، سال ارائه، مزایا و معایب را نشان می‌دهد.

جدول (1): خلاصه‌ای از جزئیات الگوریتم‌ها از جمله تکنیک مورد استفاده، سال انتشار، مزایا و معایب

الگوریتم	سال	رویکرد مورد استفاده	مسائل مورد نظر	مزایا	معایب
بهینه‌سازی ازدحام ذرات چند راهنما (MGPSO) [12]	2023	اعمال 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	1-بهبود مقیاس‌پذیری MGPSO برای مسائل در مقیاس بزرگ. 2-عملکرد رقابتی برای مسائل قابل تفکیک و تک‌وجهی. 3-عملکرد رقابتی برای مسائل پویای چندهدفه.	1-کاهش کارایی در مسائل چندوجهی، غیرقابل تفکیک، فریبنده یا جانب‌دارانه. 2-صرف محاسبات زیاد در مسائل چندهدفه.
الگوریتم ترکیبی چندهدفه مبتنی بر گرادیان به کمک جایگزین (GS-MOHA) [25]	2023	چارچوب الگوریتم ترکیبی موازی دولایه ترکیبی از جستجوی محلی چندهدفه و مکانیسم‌های تکامل سراسری	حل مسائل بهینه‌سازی آیرودینامیکی در مقیاس بزرگ با اهداف متعدد و ارزیابی هزینه‌های بالا، موازنه کارایی بهینه‌سازی و دقت همگرایی در فضاهای طراحی با ابعاد بالا	1-مدیریت موثر مسائل چندهدفه (MOPs) با حداکثر 1000 متغیر تصمیم 2-افزایش 5 تا 10 برابری بهره‌وری بهینه‌سازی در مقایسه با الگوریتم‌های چندهدفه موجود 3-مواجهه موثر با مسائل چندهدفه پرهزینه در ابعاد بالا 4-استفاده کارآمد از اطلاعات گرادیان برای بهبود دقت بهینه‌سازی 5-مناسب برای بهینه‌سازی شکل آیرودینامیکی با ابعاد بالا	1-انتخاب مدل جایگزین مناسب همچنان یک چالش است 2- زمان آموزش برای مدل‌های جایگزین ممکن است در موارد بسیاری با ابعاد بالا زیاد باشد 3-مدیریت مدل‌های جایگزین و انتخاب نقاط به‌روزرسانی می‌تواند پیچیده باشد.
الگوریتم بهینه‌ساز چندهدفه شاهپانزه (MOCHOA) [23]	2023	الگوریتم بهینه‌سازی شاهپانزه چندهدفه (MOChOA)، بر اساس الگوریتم بهینه‌ساز شاهپانزه (ChOA).	مسائل بهینه‌سازی چندهدفه (MOO) در مهندسی و زمینه‌های مختلف که در آن چندین هدف باید به طور همزمان بهینه شوند.	1-ساختار بایگانی کارآمد برای ذخیره راه‌حل‌های بهینه پرتو. 2-استراتژی انتخاب رهبر مؤثر. 3-مکانیزم شبکه پویا برای مدیریت بایگانی. 4-تضمین کاوش در فضای جستجو 5-اجتناب از بهینه محلی با استفاده از نقشه‌های آشفته. 6-ساختار موازی برای اجرای آسان. 7- پارامترهای معدود قابل تنظیم. 8-عملکرد رقابتی بالا. 9-مناسب برای برنامه‌های مختلف بهینه‌سازی چندهدفه.	1-عدم بحث در مورد نسخه‌های دودویی یا مختلط اعداد صحیح 2-عدم اشاره به مطالعات موردی کاربردی خاص 3-ارزیابی محدود به مجموعه‌ای از مسائل معیار

ادامه جدول (1): خلاصه‌ای از جزئیات الگوریتم‌ها از جمله تکنیک مورد استفاده، سال انتشار، مزایا و معایب

الگوریتم	سال	رویکرد مورد استفاده	مسائل مورد نظر	مزایا	معایب
الگوریتم قالب متعامد چندهدفه (MOOSMA) [34]	2023	الگوریتم قالب لجن چندهدفه (MOOSMA) با قاعده‌کاوی انجمنی عددی (NARM) و ادغام. یادگیری متعامد برای بهینه‌سازی در MOSMA	1-قاعده‌کاوی انجمنی عددی که یک مساله چالش‌برانگیز در داده‌کاوی است. 2-بهینه‌سازی چندهدفه برای ARM با معیارهای کارایی	1-الگوریتم بهبودیافته (MOOSMA) از نظر معیارهای مختلف عملکرد (HV ،IGD ،Dp) نتایج بهتری نسبت به MOSMA دارد 2-عملکرد بهتری از سایر الگوریتم‌ها از نظر میانگین lift و پشتیبانی در NARM دارد	1-نیاز به تخصص در پیاده‌سازی و تنظیم دقیق الگوریتم. 2-وجود محدودیت‌های بالاقوه با توجه به اشاره به کارهای آینده برای پیشرفت‌های بیشتر
مرغ مگس‌خوار مصنوعی چندهدفه (MOAHA) [22]	2022	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.	1-بهینه‌سازی موثر مسائل چندهدفه که شامل انتخاب رهبر، مکانیسم شبکه و مکانیسم بایگانی است. 2-عملکرد بهتر نسبت به سایر الگوریتم‌ها (MOPSO، MOWOA، MOHHO) در معیارهای مختلف 3-مناسب برای رسیدگی به مسائل بیوانفورماتیک و تکنیک‌های خوشه‌بندی داده‌ها.	1-طبیعت تصادفی می‌تواند نتایج را غیرقابل پیش‌بینی کند. 2-ممکن است نیاز به تنظیم برای دامنه‌های مسائل خاص داشته باشد. 3-اثربخشی ممکن است بر اساس پیچیدگی مسئله متفاوت باشد.
الگوریتم بهینه‌ساز تعادل چندهدفه (MOEO) [31]	2021	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.	1-همگرایی با سرعت بالا به دلیل توانایی بهره‌برداری قوی. 2-توانایی اکتشاف بالا با مدیریت کارآمد عامل. 3-بهبود همگرایی و تنوع بر مبنای تطبیقی. 4-توزیع یکنواخت راه‌حل‌های غیرغالب پرتو.	1-بحث محدود در مورد رویکردهای مدیریت محدودیت برای MOEO. 2-تاکید بر کاربرد در بهینه‌سازی دو و سه هدفه.
الگوریتم بهینه‌ساز چندهدفه مبتنی بر گرادیان (MOGBO) [30]	2021	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.	1-موثر در حل مسائل بهینه‌سازی دنیای واقعی در مقیاس بزرگ. 2-برای همگرایی و تنوع بهتر از جستجوی گرادیان، مرتب‌سازی غیرغالب و فاصله ازدحام استفاده می‌کند. 3-عملکرد بهتری از سایر الگوریتم‌های رقابتی از نظر پوشش، همگرایی و تنوع دارد. 4-مناسب برای کاربردهای مختلف مهندسی فراتر از بهینه‌سازی سازه.	1-زمان محاسبات بالاتر در مقایسه با برخی از الگوریتم‌های دیگر. 2-ممکن است برای مسائل بهینه‌سازی چندهدفه با تعداد اهداف بسیار زیاد کارآمد نباشد.

ادامه جدول (1): خلاصه‌ای از جزئیات الگوریتم‌ها از جمله تکنیک مورد استفاده، سال انتشار، مزایا و معایب

الگوریتم	سال	رویکرد مورد استفاده	مسائل مورد نظر	مزایا	معایب
الگوریتم بهینه‌ساز چندهدفه شکارچیان دریایی (MOMPA) [24]	2021	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.	1-نرخ همگرایی سریع با دقت جواب خوب. 2-قابلیت اکتشاف بالا. 3-مدیریت موثر انواع مختلف مسائل چندهدفه. 4-در بیشتر موارد از سایر الگوریتم‌های پیشرفته برتری دارد. 5-مناسب برای مسائل پیچیده طراحی مهندسی در دنیای واقعی.	1-نیاز به تنظیم فراپارامترها از طریق آزمون و خطا دارد. 2-توصیه‌های تنظیم خاص ممکن است بسته به مساله متفاوت باشد.
بهینه‌ساز چندهدفه شاهین هریس (MOHHO) [2]	2020	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 آزمون بدون محدودیت (معیارهای ZDT).	1-بهبود رفتار همگرایی که از طریق متریک فاصله نسلی معکوس (IGD) نشان داده شده است. 2-عملکرد بهتر نسبت به الگوریتم‌های MOALO و MODA از نظر متریک IGD برای توابع معیار ZDT. 3-مکانیسم‌های اکتشاف و بهره‌برداری موثر در الگوریتم MOHHO. 4-نتایج زمان محاسباتی رقابتی.	1-بحث محدود در مورد برنامه‌های کاربردی دنیای واقعی و حوزه‌های مساله خارج از توابع آزمون. 2-عدم مقایسه گسترده با طیف وسیع‌تری از الگوریتم‌های بهینه‌سازی چندهدفه.
الگوریتم بهینه‌ساز چندهدفه گرگ خاکستری (MOGWO) [10]	2016	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.	1-با موفقیت برای انواع مختلفی از مشکلات دنیای واقعی، از جمله سناریوهای مهندسی و تجاری اعمال شده است. 2-از غلبه پرتو و فاصله ازدحام برای حفظ مجموعه‌ای متنوع از جواب‌ها استفاده می‌کند. 3-به طور موثر به جواب‌های پرتو بهینه، که از طریق آزمون‌های معیار نشان داده شده است، همگرا می‌شود.	1-بحث محدود در مورد حساسیت الگوریتم به تنظیم پارامتر. 2-اکتشاف محدود برنامه‌های کاربردی در حوزه‌های خارج از مهندسی و تجارت. 3-عدم تجزیه‌وتحلیل گسترده در مورد مقیاس‌پذیری الگوریتم برای مسائل با ابعاد بالا.

اگرچه الگوریتم‌های فراابتکاری چندهدفه زیادی در سال‌های اخیر ارائه شده‌اند اما مطابق با قضیه NFL[1] [36]، 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 تک هدفه، نسخه چندهدفه‌ای از این الگوریتم را ارائه می‌کنیم.

در این مقاله، نسخه چندهدفه‌ای از الگوریتم تکامل شوراهای

[1] No Free Lunch theorem

شهر با نام الگوریتم تکامل شوراهای شهر چندهدفه (MOCCE) ارائه می‌شود. در الگوریتم ارائه شده، از آرشیو راه‌حل‌های بهینه برای تعریف ساختار هرم‌گونه شوراهای شهرها و شبیه‌سازی تکامل آن در فضاهای جستجوی چندهدفه استفاده می‌شود. برای ارزیابی و مقایسه کارایی الگوریتم MOCCE با کارایی الگوریتم‌های بهینه‌سازی شیر مورچه چندهدفه (MOALO)، کپک مخاطی چندهدفه (MOSMA) و مرغ مگس‌خوار مصنوعی چندهدفه (MOAHA)، آنها را روی 18 تابع آزمون 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:

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

2- ارائه یک هیوریستیک جدید برای به دست 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 2، برخی مفاهیم پایه درباره بهینه‌سازی چندهدفه و الگوریتم تکامل شوراهای شهر، مطرح و مرور می‌شوند. بخش 3 به جزئیات الگوریتم MOCCE می‌پردازد. در بخش 4، جزئیات پیاده‌سازی و نتایج تجربی ارائه می‌شوند. بخش 5 نیز به نتیجه‌گیری مقاله و پیشنهادهایی برای کارهای بعدی اختصاص دارد.

2. پیش‌زمینه

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

1,2. بهینه‌سازی چندهدفه

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

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

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 <، > 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 [12].

تعریف 1 (غلبگی پرتو[2]): 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 x' غلبه ندارد (نامغلوب) اگر برای حداقل یک تابع هدف j، داشته باشیم: $y_j \le y'_j$.

تعریف 2 (بهینگی پرتو[3]): گفته می‌شود 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

[2] Pareto dominance
[3] Pareto optimality

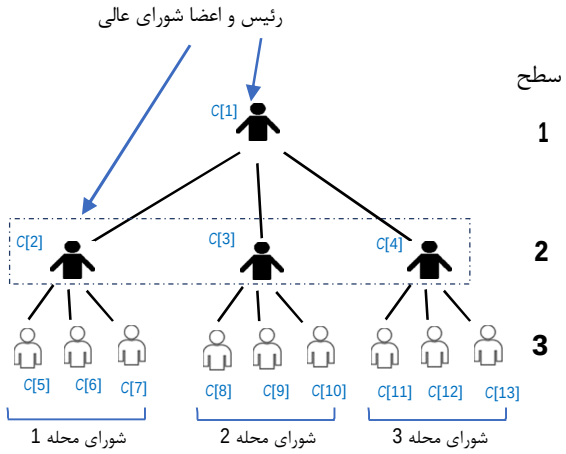
x' وجود نداشته باشد که مقدار برگشتی تابع $dominate(x, x')$ برابر true باشد.

تعریف 3 (مجموعه و جبهه بهینه پرِتو[4]): به مجموعه همه جواب‌های (بردارهای) نامغلوب، مجموعه بهینه پرِتو گفته می‌شود و بردارهای هدف این قبیل راه‌حل‌های نامغلوب، جبهه بهینه پرِتو را تشکیل می‌دهند.

2,2. الگوریتم تکامل شوراهای شهر

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 تکامل شوراهای شهر (CCE) یک الگوریتم فراابتکاری الهام‌گرفته از فرآیند تشکیل شورای عالی یک شهر است [37]. در این الگوریتم، همه اعضا و روسای شوراها در یک ساختار سلسله‌مراتبی، همچون 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 (1))، 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 اعضا و روسای شوراها و اینکه روسا در درخت شوراها در سطوح پایین‌تری نسبت به اعضا قرار می‌گیرند می‌توان چنین استنباط کرد که درخت شوراها به‌صورت یک هرم بیشینه است. در یک هرم بیشینه که به طور معمول در آرایه نگهداری می‌شود، مقدار هر گره والد از مقادیر همه گره‌های فرزندان بیشتر یا مساوی است. با فرض اینکه، تعداد اعضا هر شورا برابر d و ارتفاع درخت شوراها برابر h باشد تعداد کل اعضا (اندازه آرایه C) برابر با $N = 1 + d^1 + \cdots + d^h = (d^h - 1)/(d - 1)$ خواهد بود. اگر رئیس یک شورا در خانه j-ام آرایه باشد اعضا این شورا در خانه‌های $d \times j - d + 2, \ldots, d \times j + 1$ 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 (1): نمونه‌ای از درخت شوراهای یک شهر فرضی [37]

فرض کنید اندازه جمعیت (تعداد اعضا و روسای 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، فرآیند 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.

[4] Pareto optimal set and front

- **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 (2)، فلوچارت الگوریتم CCE را نشان می‌دهد.

3. الگوریتم تکامل شوراهای شهر چندهدفه

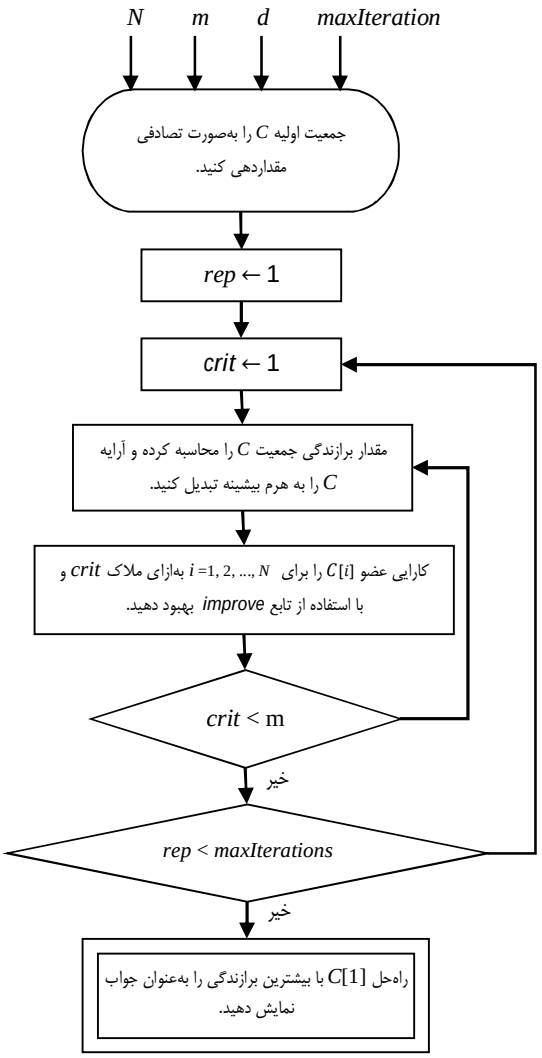
همان‌طورکه در بخش قبلی عنوان شد، الگوریتم CCE در طی فرآیند اجرا، راه‌حل‌های موجود در جمعیت جاری را در یک درخت شوراها که به صورت یک هرم بیشینه است، نگهداری می‌کند. در مسائل بهینه‌سازی تک‌هدفه که فقط با یک تابع هدف مواجه هستیم امکان ایجاد چنین درختی وجود دارد در حالی که در مسائل بهینه‌سازی چندهدفه با چندین تابع هدف نمی‌توان بین دو راه‌حل مشخص کرد که کدام یکی والد دیگری شود. بنابراین، مهمترین مساله در الگوریتم MOCCE نحوه ایجاد درخت شوراها هست. برای به دست آوردن مقدار هدف واحد به‌ازای هر راه‌حل، مقدار هر تابع هدف را در یک وزنی (ضریبی) ضرب کرده و حاصل‌جمع آنها را به دست 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).

بعد از مشخص کردن مجموعه بهینه پرتو با استفاده از تعریف 3 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 (2)، مقادیر آرایه temp به طول o محاسبه می‌شوند.

$$temp[j] = \sum_{i=1}^{nArch} \frac{f_j(Archive[i])}{\sum_{k=1}^{nArch} f_k(Archive[i])}, \quad 1 \le j \le o \qquad (2)$$

بعد از محاسبه عناصر آرایه temp، عناصر آرایه weightsCF از رابطه (3) محاسبه می‌شوند.

$$weightsCF[j] = \frac{temp[j]}{\sum_{k=1}^{o} temp[k]}, \quad 1 \le j \le o \qquad (3)$$



شکل (2): فلوچارت الگوریتم CCE [37]

به‌عنوان مثال، فرض کنید که یک مساله بهینه‌سازی 3 تابع هدف دارد (یعنی $o = 3$) و مقادیر توابع هدف برای 4 راه‌حل موجود در مجموعه بهینه پرتو (Archive) به صورت زیر باشد:

(0.12, 0.38, 0.21)
(0.22, 0.15, 0.43)
(0.49, 0.17, 0.11)
(0.35, 0.31, 0.20)

یعنی $nArch = 4$ است. ابتدا مقادیر آرایه temp را با استفاده از رابطه (2) 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 (3) 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 بر اساس ضرایب به دست 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 اینکه 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 1: شبکه‌بندی فضای هدف کشف شده (آرشیو) با توجه به اندازه شبکه و پارامتر تورم شبکه[5] (α)

مرحله 2: محاسبه موقعیت هر کدام از راه‌حل‌های نامغلوب در شبکه ایجاد شده

مرحله 3: محاسبه مقادیر وزن توابع هدف (آرایه weightsCF)

مرحله 4: انجام دو مرحله زیر به‌ازای هر کدام از m متغیر (ملاک) و برای هر یک از راه‌حل‌ها:

مرحله 4-1: تبدیل جمعیت فعلی به یک هرم بیشینه با استفاده از تابع sortPopulation

مرحله 4-2: بهبود راه‌حل فعلی (child) با استفاده از تابع Improve و با توجه به میانگینِ دو راه‌حل زیر: (1) والدش در درخت و (2) رهبرش که توسط تابع selectLeader انتخاب می‌شود. این تابع پارامترهای آرشیو (Archive) و فشار انتخاب[6] (β) را به عنوان ورودی گرفته و یک راه‌حل را به صورت زیر انتخاب کرده و به‌عنوان رهبر برمی‌گرداند. 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 (4) احتمال انتخاب هر خانه را محاسبه می‌کند. بعد از محاسبه احتمالات، از روش چرخ رولت

[5] Grid Inflation Parameter
[6] Selection Pressure

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.

$$P = k^{-\beta} \quad (4)$$

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 50%، راه‌حل Y جایگزین child می‌شود.

مرحله 5: محاسبه اعضای نامغلوب جمعیت فعلی و به‌روزرسانی آرشیو موجود

مرحله 6: نگهداری تعداد $nArch$ از راه‌حل‌های نامغلوب در آرشیو و حذف اضافه‌ها

الگوریتم (1)، شبه‌کد الگوریتم MOCCE را نشان می‌دهد. در خطوط 2-6 این الگوریتم، جمعیت اولیه به صورت تصادفی تولید شده و برازندگی آنها نیز محاسبه می‌شود. در خط 7، 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 8-29). در خط 9 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 هر کدام از راه‌حل‌های نامغلوب مشخص می‌شوند. در خط 11، مقادیر وزن توابع هدف (آرایه weightsCF)، با استفاده از تابع computeWCF محاسبه می‌شوند. در خطوط 12-26، به‌ازای هرکدام از m متغیر (ملاک)، ابتدا جمعیت فعلی به یک هرم بیشینه با استفاده از تابع sortPopulation تبدیل شده سپس هر یک از 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 27، اعضای نامغلوب جمعیت فعلی محاسبه شده و آرشیو موجود به‌روزرسانی می‌شود و در خط 28 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.

الگوریتم (1): شبه‌کد الگوریتم MOCCE

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*[$floor\left(\frac{i-2}{d}\right)$ + 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*;

1,3. پیچیدگی الگوریتم ارائه شده

برای محاسبه پیچیدگی الگوریتم MOCCE، ابتدا توابع کلیدی که در الگوریتم فراخوانی شده‌اند، را بررسی کرده و سپس به کل الگوریتم می‌پردازیم. با توجه به شبه‌کد الگوریتم MOCCE (الگوریتم (1))، توابع کلیدی به صورت زیر در این الگوریتم فراخوانی می‌شوند:

1. تابع getNonDom: چون این تابع، هر راه‌حل را با تمام راه‌حل‌های دیگر مقایسه می‌کند، لذا پیچیدگی این تابع از مرتبه $O(N^2)$ است؛
2. پیچیدگی توابع createHypercubes و setGridInde از مرتبه $O(nFunc)$ است؛
3. پیچیدگی تابع computeWCF با توجه به رابطه‌های (2) و (3)، از مرتبه $O(nFunc \times nArch)$ خواهد بود؛
4. 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)، بنابراین پیچیدگی آن از مرتبه $O(N + dom \times dom + (N - dom) \times (N - dom))$ و برابر با $O(N^2)$ خواهد بود، با این فرض که تعداد راه‌حل‌های مغلوب برابر dom باشد (حداکثر N)، و
5. پیچیدگی توابع selectLeader و deleteExtra از مرتبه $O(nArch)$ می‌باشد.

بعد از محاسبه پیچیدگی توابع مورد استفاده، به کل الگوریتم می‌پردازیم. پیچیدگی خطوط 2 تا 6 از مرتبه $O(N)$ است. پیچیدگی خط 7 که در آن تابع getNonDom فراخوانی شده برابر $O(N^2)$ است. خطوط 9 تا 28 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 (5) به دست می‌آید.

$$\begin{aligned} O\Big(N + N^2 + FESMax & \\ & \times \big(nFunc + nFunc \\ & + nFunc \times nArch + dim \\ & \times (N^2 + N \times Arch + N^2 \\ & + nArch)\big)\Big) \\ & = O\big(N^2 + FESMax \\ & \times (\, nFunc \times nArch \\ & + \; dim \times N^2)\big) \end{aligned} \qquad (5)$$

4. نتایج تجربی

در این بخش، کارایی الگوریتم MOCCE را از نظر معیارهای فاصله نسلی (GD)، فاصله نسلی معکوس (IGD) و بیشینه گستردگی (MS)، روی 18 تابع آزمون چندهدفه شناخته شده موسوم به UF [12] و IMOP [38] مورد بررسی قرار داده و نتایج به دست 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 راه‌حل‌های بهینه پرِتو درست بوده و با استفاده از رابطه (6) محاسبه می‌شود [39]:

$$GD = \frac{\sum_{i=1}^{l} d_i^2}{2} \qquad (6)$$

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 [40] و با استفاده از همان رابطه (6) محاسبه می‌شود با این تفاوت که l به تعداد راه‌حل‌های بهینه پرِتو درست و d_i به فاصله اقلیدسی بین i-امین راه‌حل بهینه پرِتو درست و نزدیکترین راه‌حل بهینه پرِتو محاسبه شده، اشاره دارند.

MS به گستردگی جبهه بهینه پرِتو پیدا شده اشاره دارد که باید بیشینه شود، به این معنی که به‌ازای هر هدف، باید دامنه گسترده‌ای از مقادیر راه‌حل‌های بهینه پرِتو پیدا شده تحت پوشش قرار بگیرد. این معیار از رابطه (7) محاسبه می‌شود:

$$MS = \sqrt{\frac{1}{o}\sum_{i=1}^{o}\left(\frac{min(f_i^{max}, F_i^{max}) - max(f_i^{min}, F_i^{min})}{F_i^{max} - F_i^{min}}\right)^2} \quad (7)$$

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 آزمون با جبهه بهینه پرتو نامنظم است که شامل 8 تابع IMOP1، IMOP2، ... و IMOP8 است که IMOP1، IMOP2 و IMOP3 دارای دو تابع هدف و IMOP4 تا 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 8 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 100 قطعه کوچک ناپیوسته تشکیل شده است.

مشابه با IMOP، تابع آزمون UF نیز فضاهای جستجوی چندهدفه متفاوتی را با جبهه‌های بهینه پرِتو محدب، غیرمحدب، ناپیوسته و چندوجهی ارائه می‌دهد. UF شامل 10 تابع آزمون با نام‌های UF1، UF2، ... و UF10 است که تعداد توابع هدف برای UF1 تا UF7 برابر 2 و برای UF8 تا UF10 برابر 3 می‌باشد. در جدول (2)، پیوند دسترسی به این توابع آزمون موجود است.

الگوریتم MOCCE دارای یک پارامتر اختصاصی با نام d است که نشان‌دهنده تعداد فرزندان هر گره در درخت شوراها است و جزو پارامترهای اختصاصی الگوریتم CCE می‌باشد. مشابه با الگوریتم‌های در نظر گرفته، این الگوریتم دارای 8 پارامتر عمومی است. جدول (2)، اسامی، شرح و مقادیر مناسب این پارامترها را نشان می‌دهد. چون بقیه الگوریتم‌ها به جز MOCCE دارای پارامتر اختصاصی نیستند، لذا فقط پارامترهای عمومی در جدول (2) آورده شده‌اند. لازم به ذکر است که مقدار پارامتر d برای الگوریتم MOCCE برابر 2 در نظر گرفته شده است.

نتایج گزارش شده در جداول نتایج حاصل اجرای الگوریتم‌ها روی هر کدام از توابع آزمون به تعداد 20 بار می‌باشند. نتایج، شامل معیارهای میانگین (Ave) و انحراف معیار (Std) است. در ضمن، جهت مقایسه عادلانه، پیکربندی پیاده‌سازی و اجرای همه الگوریتم‌ها با نرم‌افزار متلب نسخه 2017 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 [41].

جدول (2): اسامی، توصیف و مقادیر مناسب برای پارامترهای الگوریتم‌ها

اسم پارامتر	توصیف	مقدار مناسب
N	اندازه جمعیت	40
$nArch$	اندازه آرشیو	100
$FESmax$	حداکثر تعداد فراخوانی‌های تابع برازندگی	100000
$nGrid$	اندازه شبکه	10
dim	تعداد متغیرها (تعداد ملاک‌های کارایی)	10
α	اندازه تورم شبکه	0/1
β	فشار انتخاب برای انتخاب رهبر	4
γ	فشار انتخاب برای حذف از آرشیو	2

پیوند دسترسی به توابع آزمون
https://www.mathworks.com/matlabcentral/fileexchange/135984-uf-and-imop-test-functions

جداول (3) تا (8) نتایج حاصل از اجرای همه الگوریتم‌ها را روی توابع آزمون 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 آزمون 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 آزمون 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 هر کدام از الگوریتم‌های دیگر استفاده می‌شود. آزمون ویلکاکسون یک آزمون فرضیه آماری ناپارامتریک است که می‌تواند برای مقایسه دوگانه نمونه‌های مرتبط بکار رود [42]. 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 0/05 نشان‌دهنده این است که تفاوت معناداری بین عملکرد الگوریتم MOCCE و سایر الگوریتم‌ها وجود دارد. جدول (9)، 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. در ضمن چون به‌ازای مقایسه‌های MOCCE با MOSMA و MOALO، مقدار p کمتر از 0/05 است و به‌ازای مقایسه MOCCE با MOAHA، مقدار p بیشتر از 0/05 است، می‌توان نتیجه گرفت که الگوریتم MOCCE تفاوت معناداری با الگوریتم‌های MOSMA و MOALO دارد، اما بین الگوریتم‌های MOCCE و MOAHA تفاوت معناداری وجود ندارد. علاوه بر این، شکل‌های (3) و (4)، مجموعه بهینه پرتو به دست آمده با الگوریتم MOCCE را نشان می‌دهند.

جدول (3): نتایج الگوریتم‌ها (معیار GD) روی توابع آزمون 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 (4): نتایج الگوریتم‌ها (معیار IGD) روی توابع آزمون 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 (5): نتایج الگوریتم‌ها (معیار MS) روی توابع آزمون UF

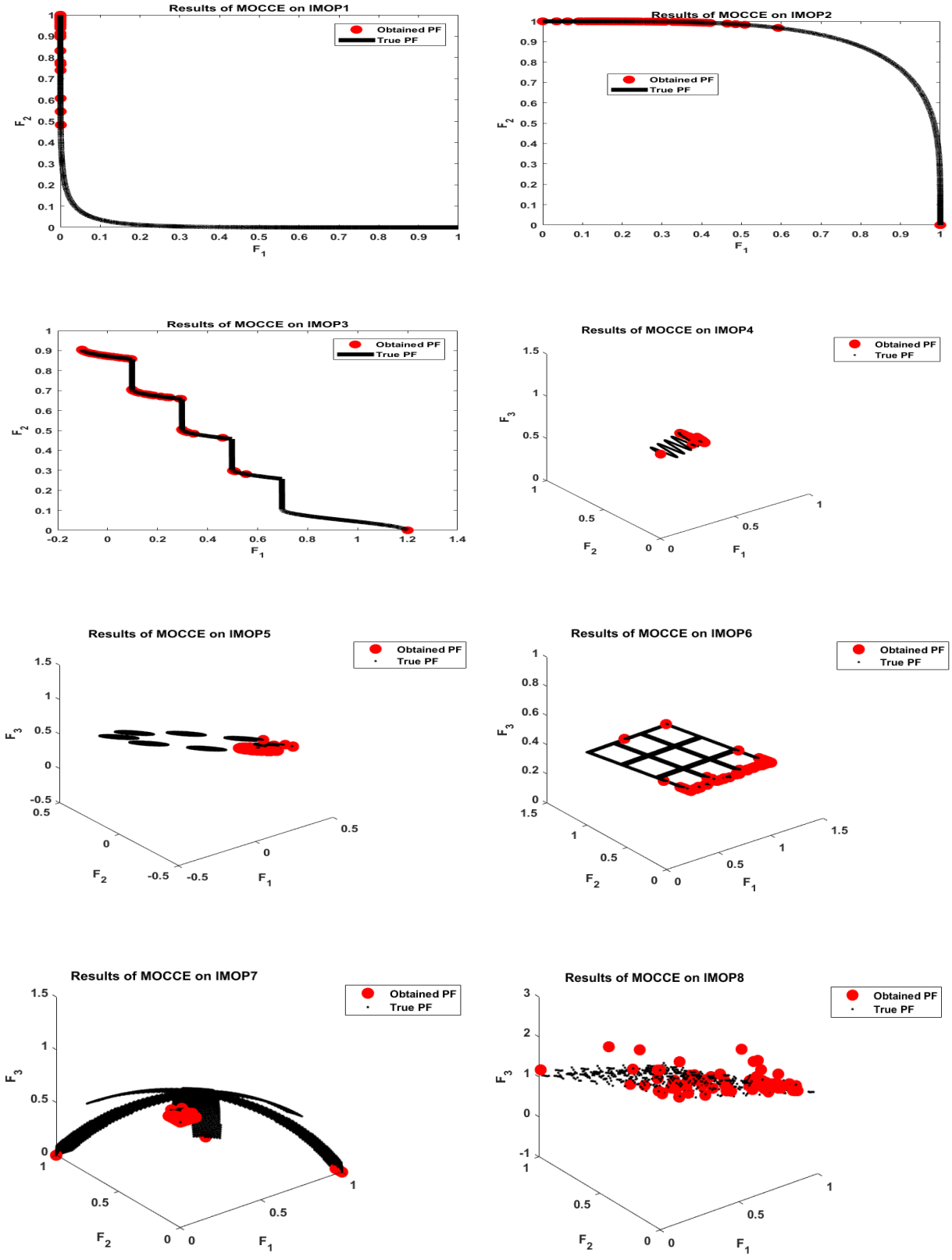
		MOCCE	MOAHA	MOSMA	MOALO
UF1	Avg	1.3499	0.48687	1.06433	1.19877
	Std	0.2113	0.31518	0.23150	0.20216
UF2	Avg	0.82817	0.25009	1.21619	1.64531
	Std	0.07934	0.03491	0.13330	0.16741
UF3	Avg	1.22379	1.44716	1.39777	1.12943
	Std	0.13266	0.31786	0.16015	0.11152
UF4	Avg	1.08845	0.27639	0.94874	1.02313
	Std	0.05889	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

جدول (6): نتایج الگوریتم‌ها (معیار GD) روی توابع آزمون IMOP

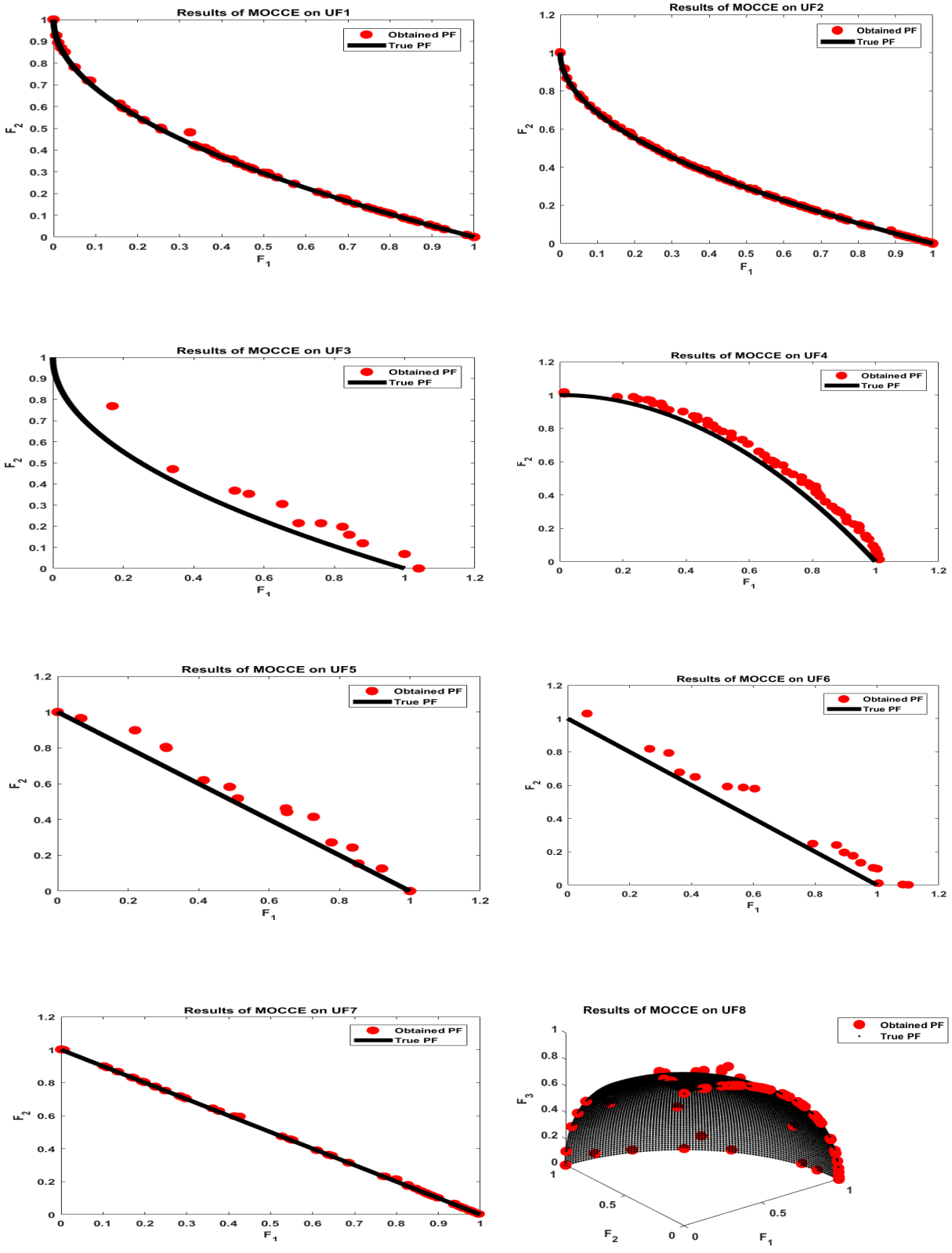
		MOCCE	MOAHA	MOSMA	MOALO
IMOP1	Avg	0.00001	0.00004	0.03404	0.00013
	Std	0.00001	0.00001	0.00125	0.00016
IMOP2	Avg	0.00001	0.00020	0.00515	0.00593
	Std	0.00000	0.00016	0.00534	0.00515
IMOP3	Avg	0.00185	0.00193	0.01237	0.00348
	Std	0.00173	0.00119	0.00107	0.00166
IMOP4	Avg	0.00002	0.00150	0.02871	0.03713
	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 (7): نتایج الگوریتم‌ها (معیار IGD) روی توابع آزمون IMOP

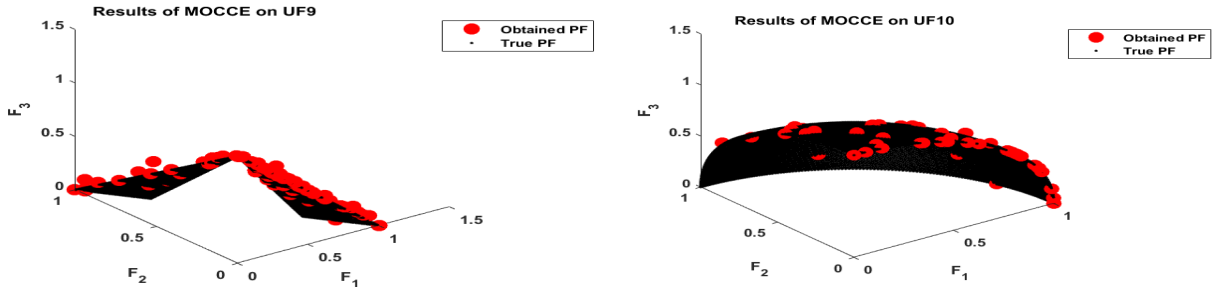
		MOCCE	MOAHA	MOSMA	MOALO
IMOP1	Avg	0.57920	0.08472	0.49059	0.16354
	Std	0.10139	0.00260	0.10628	0.06290
IMOP2	Avg	0.34743	0.10976	0.29143	0.51038
	Std	0.12599	0.01392	0.05591	0.24841
IMOP3	Avg	0.12676	0.16682	0.50325	0.09305
	Std	0.04446	0.00313	0.11569	0.02492
IMOP4	Avg	0.47196	0.14407	0.48560	0.35652
	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



شکل (3): مجموعه بهینه پرتو به دست آمده از الگوریتم MOCCE روی توابع آزمون UF



شکل (4): مجموعه بهینه پرِتو به دست آمده با الگوریتم MOCCE روی توابع آزمون IMOP



ادامه شکل (4): مجموعه بهینه پرِتو به دست آمده با الگوریتم MOCCE روی توابع آزمون IMOP

جدول (8): نتایج الگوریتم‌ها (معیار MS) روی توابع آزمون 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

5. نتیجه‌گیری و کارهای آینده

در این مقاله، نسخه 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

جدول (9): 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.	$p > 0.05$	$p < 0.05$	$p < 0.05$

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 به دست می‌آید. برای ارزیابی کارایی الگوریتم ارائه شده، آن را روی 18 تابع آزمون چندهدفه شناخته شده موسوم به UF و IMOP اجرا کرده و نتایج به دست آمده با نتایج الگوریتم‌های بهینه‌سازی شیر مورچه چندهدفه (MOALO)، کپک مخاطی چندهدفه (MOSMA) و مرغ مگس‌خوار مصنوعی چندهدفه (MOAHA) مقایسه شدند. نتایج آزمون میانگین رتبه فریدمن، کارایی بالای الگوریتم MOCCE را نسبت به الگوریتم‌های مذکور از نظر معیارهای فاصله نسلی (GD)، فاصله نسلی معکوس (IGD) و بیشنیه گستردگی (MS) تایید می‌کنند. بکارگیری الگوریتم ارائه شده برای حل مسائل کاربردی و مهندسی و همچنین ارائه نسخه دودویی می‌توانند به‌عنوان کارهای آتی در نظر گرفته شوند.

تعارض منافع: نویسندگان اعلام می‌کنند که هیچ تعارض منافعی ندارند.

مراجع

[1] H. Shafiei, V. Rafe, and M. Amiri, “Optimization of vehicle routing based on the combination of ant colony and particle swarm algorithms with the heuristic function of the cosine of angles,” Soft Comput. J., vol. 12, no. 2, pp. 146-164, 2024, doi: 10.22052/scj.2023.248702.1118 [In Persian].

[2] 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,” Adv. Eng. Soft., vol. 114, pp. 163-191, 2017, doi: 10.1016/j.advengsoft.2017.07.002.

[3] U. Yuzgec and M. Kusoglu, “Multi-objective harris hawks optimizer for multiobjective optimization problems,” BSEU J. Eng. Res. Technol., vol. 1, no. 1, pp. 31-41, 2020.

[4] B. Cao, M. Li, X. Liu, J. Zhao, W. Cao, and Z. Lv, “Many-objective deployment optimization for a drone-assisted camera network,” IEEE Trans. Network Sci. Eng., vol. 8, no. 4, pp. 2756-2764, 2021, doi: 10.1109/TNSE.2021.3057915.

[5] J. Branke, T. Kaubler, and H. Schmeck, “Guidance in evolutionary multi-objective optimization,” Adv. Eng. Soft., vol. 32, no. 6, pp. 499-507, 2001, doi: 10.1016/S0965-9978(00)00110-1.

[6] M. Zamzame, S. Sedighian Kashi, and A. Nikanjam, “Energy-aware evolutionary multi-objective refactoring for bad code smells correction of Android applications,” Soft Comput. J., vol. 12, no. 2, pp. 72-89, 2024, doi: 10.22052/scj.2023.246479.1074 [In Persian].

[7] 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,” Struct. Multidiscip. Optim., vol. 57, pp. 1047-1060, 2018, doi: 10.1007/s00158-017-1793-2.

[8] 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 Comput., vol. 15, pp. 1313-1331, 2011, doi: 10.1007/s00500-010-0673-0.

[9] 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: 10.1016/j.omega.2018.10.013.

[10] L. Lai, L. Fiaschi, M. Cococcioni, and K. Deb, “Solving mixed pareto-lexicographic multiobjective optimization problems: the case of priority levels,” IEEE Trans. Evol. Comput., vol. 25, no. 5, pp. 971-985, 2021, doi: 10.1109/TEVC.2021.3068816.

[11] 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 Evol. Comput., vol. 69, p. 100987, 2022, doi:

10.1016/j.swevo.2021.100987.

[12] 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 Syst. Appl., vol. 47, pp. 106-119, 2016, doi: 10.1016/j.eswa.2015.10.039.

[13] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," Evol. Comput., vol. 2, no. 3, pp. 221-248, 1994, doi: 10.1162/evco.1994.2.3.221.

[14] C.C. Coello and M.S. Lechuga, "MOPSO: A proposal for multiple objective particle swarm optimization," in Proc. Cong. Evol. Comput. (CEC), 2002, vol. 2, pp. 1051-1056, doi: 10.1109/CEC.2002.1004388.

[15] J. Salimisartaghti, and S. Goli Bidgoli, "A Hybrid Algorithm using Firefly, Genetic, and Local Search Algorithms," Soft Comput. J., vol. 8, no. 1, pp. 14-28, 2019, doi: 10.22052/8.1.14 [In Persian].

[16] T. Murata and H. Ishibuchi, "MOGA: multi-objective genetic algorithms," in IEEE Int. Conf. Evol. Comput., NJ, USA, 1995, vol. 1, pp. 289-294.

[17] 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 Int. Conf., Paris, France, 2000, Springer, pp. 849-858, doi: 10.1007/3-540-45356-3_83.

[18] J. Horn, N. Nafpliotis, and D.E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in Proc. 1st IEEE Conf. Evol. Comput., 1994, pp. 82-87, doi: 10.1109/ICEC.1994.350037.

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

[20] A.A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," Future Gener. Comput. Syst., vol. 97, pp. 849-872, 2019, doi: 10.1016/j.future.2019.02.028.

[21] H.A. Abbass, R. Sarker, and C. Newton, "PDE: a Pareto-frontier differential evolution approach for multi-objective optimization problems," in Proc. 2001 Cong. Evol. Comput. (IEEE Cat. No. 01TH8546), 2001, vol. 2, pp. 971-978, doi: 10.1109/CEC.2001.934295.

[22] S.-Y. Zeng et al., "A dynamic multi-objective evolutionary algorithm based on an orthogonal design," in 2006 IEEE Int. Conf. Evol. Comput., 2006, pp. 573-580, doi: 10.1109/CEC.2006.1688361.

[23] K. Zhong, G. Zhou, W. Deng, Y. Zhou, and Q. Luo, "MOMPA: Multi-objective marine predator algorithm," Comput. Methods Appl. Mech. Eng., vol. 385, p. 114029, 2021, doi: 10.1016/j.cma.2021.114029

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

[25] N. Khodadadi, S.M. Mirjalili, W. Zhao, Z. Zhang, L. Wang, and S. Mirjalili, "Multi-objective artificial hummingbird algorithm," in Adv. Swarm Intell. Variations Adapt. Optim. Probl., Springer, 2022, pp. 407-41, doi: 10.1007/978-3-031-09835-2_22

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

[27] H.R. Hassanzadeh and M. Rouhani, "A multi-objective gravitational search algorithm," in 2nd Int. Conf. Comput. Intell. Commun. Syst. Networks, 2010, pp. 7-12, doi: 10.1109/CICSyN.2010.32.

[28] J.J. Flores, R. Lopez, and J. Barrera, "Gravitational interactions optimization," in Int. Conf. Learn. Intell. Optim., 2011, pp. 226-237, doi: 10.1007/978-3-642-25566-3_17.

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

[30] 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," J. Comput. Des. Eng., vol. 9, no. 1, pp. 24-50, 2022, doi: 10.1093/jcde/qwab065.

[31] 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: 10.3390/math11183844.

[32] 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: 10.1109/ACCESS.2021.3106487.

[33] 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 Comput. Appl., vol. 35, no. 8, pp. 6125-6151, 2023, doi: 10.1007/s00521-022-07985-w.

[34] F. Zou, L. Wang, X. Hei, D. Chen, and B. Wang, "Multi-objective optimization using teaching-learning-based optimization algorithm," Eng. Appl. Artif. Intell., vol. 26, no. 4, pp. 1291-1300, 2013, doi: 10.1016/j.engappai.2012.11.006.

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

[36] D.H. Wolpert and W.G. Macready, "No free lunch theorems for optimization," IEEE Trans. Evol. Comput., vol. 1, no. 1, pp. 67-82, 1997, doi: 10.1109/4235.585893.

[37] E. Pira, "City councils evolution: A socio-inspired metaheuristic optimization algorithm," J. Ambient Intell. Humaniz. Comput., pp. 1-50, 2022, doi: 10.1007/s12652-022-03765-5.

[38] J. Cao, J. Zhang, F. Zhao, and Z. Chen, "A two-stage evolutionary strategy based MOEA/D to multi-objective problems," Expert Syst. Appl., vol. 185, p. 115654, 2021, doi: 10.1016/j.eswa.2021.115654.

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

[40] M.R. Sierra and C.A. Coello Coello, "Improving PSO-based multi-objective optimization using crowding, mutation and ∈-dominance," in 3rd Int. Conf. Evol. Multi-Criterion Optim. (EMO), Guanajuato, Mexico, 2005, Springer, pp. 505-519, doi: 10.1007/978-3-540-31880-4_35.

[41] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," Ann. Math. Statist., vol. 11, no. 1, pp. 86-92, 1940.

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