Journal of Systems Smartening and Data Processing
Vol. 2 - No. 3 - Fall 2024

Biogeography-Based Optimization with Brownian Population for the MultiObjective Next Release

Mohsen Ghasemi [1], *Assistant Professor*, Mehdi Talebi[2], *Assistant Professor*

[1] Department of Computer Engineering, Larestan Branch, Islamic Azad University, Larestan, Iran
mohsen.ghasemi@iau.ac.ir

[2] Department of Computer Engineering, Larestan Branch, Islamic Azad University, Larestan, Iran
mehdi.talebi@iau.ac.ir

Received: 10 November 2024	**Revised:** 24 January 2025	**Accepted:** 25 February 2025

Abstract:

Software developers must decide which requirements to implement in next version. Selecting a set of requirements for the next software release is a complex NP-hard problem known as the Next Release Problem (NRP). We propose a multi-objective biogeography-based optimization algorithm for this problem. Furthermore, inspired by the Brownian-like random-walk, a population called the Brownian population is produced and combined with the biogeography-based optimization algorithm. We propose a novel approach to calculate the migration probability to each habitat based on the ratio of objective functions. We also employ two approaches: merging solutions and populations with non-duplicate members in algorithm iterations. Two datasets and quality indicators of hypervolume (HV), Spread, and the number of non-dominated solutions (NDS) are used for evaluation. To investigate the impact of the Brownian population and non-duplicate population, we implement the proposed algorithm, along with three other versions of the biogeography-based algorithm, to solve the next release problem and compare their results. The results demonstrate that the proposed method improves the biogeography-based algorithm. The Brownian population is more effective in improving Spread, while the non-duplicate population is more effective in increasing the number of non-dominated solutions. Compared to other similar research, the proposed algorithm outperforms previously reported algorithms.

Keywords: Next release problem, Multiobjective optimization, Biogeography-based optimization, Brownian population, Search-based software engineering.

Corresponding Author: Dr. Mohsen Ghasemi
Corresponding Author Address: Department of Computer Engineering- Dadman Blvd. -Islamic Azad University- Larestan Branch, Iran

بهینه‌سازی جغرافیای زیستی با جمعیت براونی برای مسئله چندهدفی انتشار نسخه بعد

محسن قاسمی[۱]، استادیار، مهدی طالبی[۲]، استادیار

۱- دانشکده مهندسی کامپیوتر - واحد لارستان، دانشگاه آزاد اسلامی، لارستان، ایران
mohsen.ghasemi@iau.ac.ir

۲- دانشکده مهندسی کامپیوتر - واحد لارستان، دانشگاه آزاد اسلامی، لارستان، ایران
mehdi.talebi@iau.ac.ir

تاریخ ارسال مقاله: ۱۴۰۳/۰۸/۲۰ **تاریخ بازنگری مقاله:** ۱۴۰۳/۱۱/۰۵ **تاریخ پذیرش مقاله:** ۱۴۰۳/۱۲/۰۷

چکیده: توسعه‌دهندگان نرم‌افزار باید تصمیم بگیرند که چه نیازمندی‌هایی در نسخه‌های بعدی، پیاده‌سازی گردد. انتخاب یک مجموعه از نیازمندی‌ها در انتشار نسخه بعدی نرم‌افزار یک مسئله بغرنج (NP-Hard) است که بنام مسئله انتشار نسخه بعد (NRP) شناخته می‌شود. در این تحقیق، الگوریتم بهینه‌سازی جغرافیای زیستی بصورت چندهدفی برای این مسئله ارائه می‌کنیم. همچنین با الهام از حرکت تصادفی شبه براونی، جمعیتی بنام جمعیت براونی ایجاد کرده و با الگوریتم بهینه‌سازی جغرافیای زیستی ترکیب می‌کنیم. احتمال مهاجرت به هریک از زیستگاه‌ها را با رویکردی جدید و براساس نسبت توابع هدف، ایجاد می‌کنیم. رویکردهای ادغام پاسخ‌ها و جمعیت با اعضای غیرتکراری را نیز در تکرارهای الگوریتم، بکار می‌بریم. از دو مجموعه‌داده و شاخص‌های کیفیت ابرحجم، گستردگی و تعداد پاسخ‌های غیرمغلوب، برای ارزیابی استفاده می‌شود. برای بررسی تاثیر جمعیت براونی و جمعیت غیرتکراری؛ الگوریتم پیشنهادی؛ یعنی الگوریتم جغرافیای زیستی با جمعیت براونی و جمعیت غیرتکراری را با سه نسخه دیگر الگوریتم جغرافیای زیستی، برای حل مسئله انتشار نسخه بعد، پیاده‌سازی کرده و نتایجشان را مقایسه می‌نماییم. نتایج نشان می‌دهند که روش پیشنهادی، الگوریتم جغرافیای زیستی را بهبود داده است. جمعیت براونی در بهبود گستردگی و جمعیت غیرتکراری در افزایش تعداد پاسخ‌های غیرمغلوب موثرترند. در مقایسه با تحقیقات مشابه دیگران نیز، الگوریتم پیشنهادی، عملکرد بهتری از دیگر الگوریتم‌های گزارش‌شده قبلی دارد.

کلمات کلیدی: مسئله انتشار نسخه بعد، بهینه‌سازی چندهدفی، بهینه‌سازی جغرافیای زیستی، جمعیت براونی، مهندسی نرم‌افزار مبتنی بر جستجو

نام نویسنده‌ی مسئول: محسن قاسمی
نشانی نویسنده‌ی مسئول: لارستان- بلوار دکتر دادمان- دانشگاه آزاد اسلامی واحد لارستان - دانشکده مهندسی کامپیوتر

۱- مقدمه

در محیط کسب و کار امروزی، روابط و نیازهای مشتریان دایما در حال تغییر و پیچیده‌شدن است و بدون استفاده از نرم-افزارهای مناسب نمی‌توان با محیط کسب و کار امروز روبرو شد. بزرگ و پیچیده‌شدن نیازمندی‌ها و روابط بین بازیگران محیط‌های کسب و کار، بسیاری از شرکت‌ها و تولیدکنندگان نرم‌افزار را درگیر سیستم‌های بزرگ و پیچیده کرده است. شرکتهای نرم‌افزاری برای باقی‌ماندن در دنیای رقابتی امروز، نسخه‌های ارتقایافته نرم‌افزار خود را باید در بازه‌های زمانی کوتاهی منتشر کنند. یک مرحله مهم در فاز تحلیل نیازمندی‌های یک پروژه نرم‌افزاری، انتخاب نیازمندی‌های مناسب برای انتشار در نسخه بعدی نرم‌افزار است بطوریکه حداکثر سود تجاری همراه با هزینه محدود بدست آید. استخراج نیازمندی‌ها مهمترین گام در فرایند مهندسی نیازمندی‌هاست که موفقیت یا شکست پروژه، بستگی به کیفیت نیازمندی‌ها و درنظر گرفتن درخواست‌های مشتریان دارد.

محیط‌های پویا و رقابتی امروز و تغییرات در نیازمندی‌ها و مدل کسب و کار، شرکت‌های نرم‌افزاری را ملزم می‌کند که نسخه-های جدیدی از نرم‌افزار را در فاصله‌های زمانی کوتاهی انتشار دهند. یکی از اولین مراحل در مهندسی نرم‌افزار، استخراج و تعیین نیازمندی‌هاست. مهندسی نیازمندی‌ها، فرآیند جمع‌آوری، تحلیل و مستندسازی نیازمندی‌های نرم‌افزار از مشتریان است [۱]. اعمال نیارمندی‌های جدید طیف وسیعی از کاربران، مستلزم صرف زمان و هزینه از سمت شرکت‌های نرم‌افزاری از یک سو و حصول اطمینان از کسب رضایتمندی مشتریان از سوی دیگر می‌باشد.

مشتریان مجموعه‌ای از نیازمندی‌ها که برایشان مهم است را تقاضا می‌کنند. اما بعلت محدودیت منابع و بودجه، همه درخواست‌های مشتریان را نمی‌توان برآورده کرد. هر نیازمندی یک هزینه پیاده‌سازی دارد. مشتریان نیز اهمیت یکسانی برای شرکت ندارند. همچنین نیازمندی‌ها نیز ارزش یکسانی برای مشتریان ندارند. فرایند انتخاب را با در نظر گرفتن برخی اولویت‌ها می‌توان انجام داد. با افزایش تعداد مشتری و نیازمندی، پیچیدگی مسئله نیز افزایش می‌یابد.

با توجه به محدودیت بودجه و منابع، پیاده‌سازی همه نیازمندی‌های کاربران مقدور نیست، لذا شرکت‌های نرم‌افزاری باید تصمیم بگیرند که کدام نیازمندی‌ها و ویژگی‌ها را در نسخه بعدی نرم‌افزار پیاده‌سازی کنند. مهندسی نرم‌افزار مبتنی بر جستجو[۱](SBSE) ، یک حوزه در مهندسی نرم‌افزار است که از تکنیک‌های بهینه‌سازی مبتنی بر جستجو در مسائل مهندسی نرم‌افزار استفاده می‌کند [۲]. بسیاری از مسائل در مهندسی نرم‌افزار نیاز به بهینه‌سازی دو یا چند هدف دارند که منجر به پیچیدگی محاسباتی زیاد می‌شود که امکان جستجوی جامع را نمی‌دهد. تعریف مسائل مهندسی نرم‌افزار به‌عنوان مسائل بهینه‌سازی، راه را برای استفاده از روش‌های کلاسیک و جدید مانند الگوریتم‌های تکاملی باز می‌کند [۳]. انتخاب مجموعه بهینه نیازمندی‌ها برای شامل شدن در انتشار بعدی نرم‌افزار، یکی از پنج مسئله رایج در مهندسی نرم‌افزار مبتنی بر جستجو است [۴] که بعنوان مسئله انتشار نسخه بعد[۲] (NRP) شناخته می‌شود. این مسئله، اولین بار بوسیله بگنال و همکاران، در سال ۲۰۰۱ معرفی شد [۵]. ماهیت پیچیده این مسئله، آن را به یک مسئله پیچیده NP-Hard تبدیل می‌کند [۶]. بنابراین، الگوریتم های فراابتکاری به طور گسترده‌ای برای حل این مسائل استفاده می شود [۷]. جستجوی فراابتکاری به اعداد تصادفی برای یافتن پاسخ‌های مناسب برای مسائل بهینه‌سازی بدون تجزیه و تحلیل کل فضای جستجو متکی است. هدف اصلی آن‌ها، یافتن پاسخ بهینه یا نزدیک بهینه در زمان معقول است. الگوریتم‌های مبتنی بر جمعیت، الگوریتم های فراابتکاری هستند که با مجموعه‌ای از پاسخ‌ها در هر تکرار سر و کار دارند. دسته عمده‌ای از الگوریتم‌های مبتنی بر جمعیت، الگوریتم‌های بهینه‌سازی الهام گرفته از طبیعت هستند. آنها از تکامل در طبیعت الهام گرفته‌اند و مجموعه‌ای از پاسخ‌ها را در هر نسل از فرآیند بهینه-سازی، دستکاری می‌کنند [۸].

رضایتمندی بیشتر مشتریان و هزینه کمتر مصرف منابع دو هدف متناقض شرکت نرم‌افزاری در انتخاب نیازمندی‌ها برای پیاده‌سازی در نسخه بعدی نرم‌افزار است [۹]. به‌علت این دو هدف متضاد، این مسئله را می‌توان بعنوان یک مسئله بهینه‌سازی چند هدفی در نظر گرفت که بجای یک پاسخ، مجموعه‌ای از پاسخ‌ها بنام مجموعه بهینه پاسخ‌های پارتو[۳] یا پاسخ‌های غیرمغلوب[۴] دارند. بهینه‌سازی دو یا بیشتر تابع هدف بطور همزمان را بهینه‌سازی چندهدفی می‌نامند [۱۰]. الگوریتم‌های تکاملی چندهدفی کاندیدای مناسبی برای حل مسئله انتشار نسخه بعد هستند. این الگوریتم‌ها، چندهدفی هستند و از فرایند

تکامل در طبیعت الهام گرفته‌اند. الگوریتم‌های تکاملی چندهدفی، یک مجموعه از پاسخ‌ها را در یک اجرا بدست می‌آورند [۱۱]. بسیاری از الگوریتم‌های چندهدفی معروف مانند NSGA2، PESA، Pareto GA و SPEA2 برای حل این مسئله استفاده شده‌اند [۱۲، ۱۳].

الگوریتم‌های چندهدفی لزوما پاسخ‌های بهینه را بدست نمی‌آورند. جبهه‌های پارتو بدست‌آمده با الگوریتم‌های تکاملی چندهدفی در مسئله انتشار نسخه بعد، ممکن است بسیار متفاوت از جبهه پارتو بهینه باشد. برخی محققان ادعای محاسبه پاسخ‌های خوب برای مسئله انتشار نسخ بعد دارند، ولی جزئیات پاسخ‌ها را ارائه نداده‌اند [۱۴]. آنها نتایج شاخص‌های کیفیت را روی جبهه‌های پارتوی بدست‌آمده، گزارش کرده‌اند. نتایج آن‌ها بسیار متفاوت از شاخص‌های کیفیت جبهه پارتو بهینه است و پاسخ‌های آن‌ها به اندازه کافی دقیق نیست. بنابراین، بهبود الگوریتم‌های تکاملی چندهدفی برای حل مسئله انتشار نسخ بعد مورد نیاز است. ما تلاش می‌کنیم که جبهه پارتو با دقت بیشتری را بدست آوریم.

با توجه به تئوری NFL[۵]، هیچ تکنیک بهینه‌سازی برای حل تمامی مسائل بهینه‌سازی وجود ندارد. اگر الگوریتمی روی دسته‌ای از مسائل خوب عمل کند، نمی‌توانیم نتیجه بگیریم که روی دیگر کلاس‌های مسائل به خوبی کار می‌کند [۱۵]. این قضیه به محققان اجازه می‌دهد تا الگوریتم‌های بهینه‌سازی فعلی یا جدید را برای کلاس‌های جدید مسائل تطبیق دهند. روش‌های تجمعی یک مسئله بهینه‌سازی چندهدفی را به چندین زیر مسئله تک‌هدفی تقسیم می‌کنند. این روش‌ها یک پاسخ را به دست می آورند. با این حال، بهینه‌سازی چندهدفی، مجموعه ای از پاسخ‌ها را به عنوان جبهه پارتو ارائه می‌دهند. مجموعه‌ای از پاسخ‌ها گزینه‌های بیشتری را در اختیار تصمیم‌گیرندگان قرار می‌دهد و تصویر بهتری از آنچه آنها باید بدانند، ارائه می‌دهد [۱۷،۱۶]. در نتیجه محققان می‌توانند تکنیک‌های مختلفی را به‌ویژه به‌صورت ترکیبی برای مسائل بهینه‌سازی بکار گیرند. تمرکز ما در این مقاله، استفاده از الگوریتم‌های تکاملی چندهدفی مبتنی بر غلبه پارتو برای حل مسئله انتشار نسخه بعد است.

در این مقاله، الگوریتم تکاملی بهینه‌سازی مبتنی بر جغرافیای زیستی[۶] (BBO) را برای حل مسئله انتشار نسخه بعد، بصورت چندهدفی ارائه می‌کنیم. احتمال مهاجرت به هر زیستگاه را با روش جدید؛ براساس نسبت توابع هدف، ایجاد می‌کنیم. برای بهبود پاسخ‌ها، با روشی جدید، جمعیتی کمکی بنام جمعیت براونی براساس حرکت تصادفی شبه براونی[۷]، ایجاد می‌کنیم. در هر مرحله الگوریتم نیز از جمعیت با اعضای غیرتکراری و ادغام پاسخ‌ها استفاده می‌کنیم.

نتایج بدست آمده با الگوریتم پیشنهادی را روی دو مجموعه‌داده استاندارد، با نسخه‌های دیگر همین الگوریتم و نتایج دیگر محققان، مقایسه می‌کنیم. مقایسه براساس شاخص‌های بهینه‌سازی چندهدفی انجام می‌گیرد. در ادامه، در بخش ۲، سوابق مربوطه را بیان می‌کنیم. در بخش ۳، مسئله انتشار نسخه بعد چندهدفی معرفی شده است. در بخش ۴، عملگرهای استفاده شده برای چندهدفی کردن الگوریتم‌ها، بیان شده‌است. الگوریتم چندهدفی مبتنی بر جغرافیای زیستی و جمعیت براونی در بخش ۵، تشریح شده‌است. در بخش ۵، آزمایش‌ها و نتایج بیان شده است و در پایان، بخش ۶ شامل نتیجه‌گیری است.

۲- سوابق مربوطه

بطور فنی، تکنیک‌های بهینه‌سازی نیازمندی‌های نرم‌افزار را می‌توان بصورت بهینه‌سازی نیازمندی‌ها مبتنی بر اولویت، بهینه-سازی نیازمندی‌ها مبتنی بر جستجو و بهینه‌سازی نیازمندی‌ها به روش‌های دقیق دسته‌بندی کرد [۱۸].

بگنال و همکاران که مسئله نسخه انتشار بعدی را معرفی کردند، برنامه‌ریزی خطی صحیح را با ابزار Cplex برای این مسئله بکار بردند و با نتایج حاصل از الگوریتم‌های حریصانه، تپه نوردی، شبیه‌سازی تبریدی مقایسه کردند. در این مقاله مسئله انتشار نسخه بعدی نرم‌افزار را به روش کلاسیک و با تبدیل توابع هدف به یک تابع هدف حل کردند [۵]. مسائل بهینه‌سازی چندهدفی را با جمع وزنی توابع هدف، می‌توان به مسائل بهینه‌سازی تک هدفی کاهش داد [۱۹]. در [۲۰] روش‌های تجزیه و غلبه پارتو را برای NRP ترکیب کردند. تکنیک‌های دقیق نیز برای این مسئله استفاده شده‌است [۲۱].

در سال ۲۰۰۷ ، ژانگ و همکاران، مسئله انتشار نسخه بعد را بصورت چندهدفی بازنویسی کردند و با الگوریتم NSGA2 و الگوریتم ژنتیک سلولی چندهدفی، آن را حل کردند [۱۲]. الگوریتم ژنتیک بصورت یک هدفی [۲۲] و چندهدفی [۲۳،۱۲]، برای مسئله NRP بکار رفته است.

در [۲۴]، الگوریتم کلونی مورچگان را در متدلوژی توسعه نرم‌افزار افزایشی و متدلوژی چابک بکار بردند. آن‌ها مجددا مسئله انتشار نسخه بعد را با در نظر گرفتن انواع روابط بین نیازمندی‌ها، با الگوریتم مورچگان چندهدفی حل کردند [۱۳].
در [۲۵]، الگوریتم چندهدفی تکامل تفاضلی را براساس تورنمنت پارتو ارائه کردند و سپس شبیه به آن نسخه جدید دوهدفی الگوریتم بهینه‌سازی مبتنی بر آموزش-یادگیری را ارائه داند [۱۴]. در [۲۶]، از ترکیب بهینه‌سازی مبتنی بر آموزش-یادگیری و کلونی زنبور مصنوعی، الگوریتم MOTLABC را ارائه کردند و برای حل NRP استفاده نمودند.
در [۲۷] الگوریتم ازدحام ذرات و روش مقداردهی اولیه جمعی تصادفی-حریصانه را برای حل NRP ترکیب کردند. دو بردار اضافی سرعت بر هر ذره تعریف کردند. تابع هدف را بصورت جمع وزنی سود و هزینه نیازمندی‌های انتخاب شده در نظر گرفتند.
در [۲۸]، رویکرد موازی‌سازی را برای حل NRP ارائه کردند. آنها چند الگوریتم کلونی زنبور مصنوعی چندهدفی را همزمان اجرا کردند. روش موازی از اجرای سریال نتایج بهتری بدست داد. در [۲۹]، یک الگوریتم ترکیبی از کلونی مصنوعی زنبور عسل چندهدفی و تکامل تفاضلی به نام HABC-DE، برای حل مسئله انتشار نسخه بعدی نرم‌افزار ارائه شد. رویکرد پیشنهادی شامل مدیریت از کلونی مصنوعی زنبور عسل اصلی با عملگرهای الگوریتم تکامل تفاضل برای متعادل کردن مراحل بهره‌برداری و اکتشاف فرآیند بهینه‌سازی است. الگوریتم پیشنهادی را روی دو مجموعه‌داده گریر و ساگرادو و با معیارهای ارزیابی ابرحجم[۸] (HV)، گستردگی (spread) و تعداد پاسخ‌های غیرمغلوب (NDS) ارزیابی کردند. نتایج نشان داد که الگوریتم HABC-DE پیشنهادی آن‌ها، در مقایسه با ۷ الگوریتم دیگر، مجموعه بهتری از پاسخ‌های غیرمغلوب را با گستردگی کمتر و ابرحجم بیش‌تر تولید می‌کند. در [۱۷] نیز با ترکیب الگوریتم‌های ژنتیک و کلونی زنبور مصنوعی، الگوریتم چند هدفی بنام HGABC، برای مسئله انتشار نسخه بعد نرم‌افزار ارائه شد. الگوریتم پیشنهادی را روی دو مجموعه‌داده و با معیارهای ارزیابی ابرحجم، گستردگی و تعداد پاسخ‌های غیرمغلوب با ۴ الگوریتم دیگر ارزیابی کردند. نتایج نشان داد که الگوریتم پیشنهادی آن‌ها، در مقایسه با ۴ الگوریتم دیگر، نتایج بهتری داشت. در این مقاله، شرط توقف الگوریتم، تعداد فراخوانی‌های تابع هدف نیست و الگوریتم ۱۰۰ بار تکرار می‌گردد.
در [۹]، عملکرد نوزده الگوریتم تکاملی براساس سه شاخص، ابرحجم، گستردگی و زمان اجرا، برای حل مسئله انتشار نسخه مقایسه شدند. از ۱۷ مسئله با مقیاس کوچک تا بزرگ در دو مجموعه‌داده classic و realistic برای ارزیابی الگوریتم‌ها استفاده کردند. به طور کلی، NSGA2 بهترین زمان اجرا را در تمام مقیاس‌های تست نشان داد. NNIA بهترین نتیجه برای ابر حجم و NNIA و Spea2 بهترین مقادیر برای گستردگی را بدست آوردند.
در [۳۰]، استراتژی‌های گروه‌بندی را به عنوان وسیله‌ای برای کاهش تعداد مشتریان برای مدیریت در انتخاب نیازمندی‌ها و در عین حال حفظ پوشش مشتریان پیشنهاد کردند. بر اساس برجستگی مشتریان بر حسب قدرت، مشروعیت و فوریت آن‌ها، استراتژی های متنوعی برای انتخاب گروه های مهم مشتریان اعمال کردند. پس از تعیین تعداد خوشه‌ها بر اساس شاخص‌های اعتبارسنجی، از k-means، k-medoids و خوشه‌بندی سلسله مراتبی استفاده نمودند. هر تکنیک مورد استفاده، گروه‌های مختلفی را پیدا کرد و تعداد گروه‌های مشتریان که به صورت تجربی پیشنهاد شد، ۴ یا ۳ بود.
در [۳۱]، چارچوب تکاملی چندهدفی برای حل مسئله انتشار نسخه بعدی نرم‌افزار ارائه شد. در این چارچوب، در هر تکرار الگوریتم تکاملی، جمعیت با اعضای غیرتکراری استفاده گردید و عملگرهای تکاملی، جمعیت جدیدی ایجاد می‌کنند که با جمعیت اصلی ادغام می‌گردد. پاسخ‌ها بر اساس مرتب‌سازی غیرمغلوب[۹] و فاصله ازدحامی[۱۰] رتبه‌بندی می‌شوند. همچنین برای تبدیل پاسخ‌های تصادفی به پاسخ‌هایی که محدودیت‌های مسئله را برآورده می‌کنند، روشی مبتنی بر تعداد وابستگی‌های هر نیازمندی و نسبت هزینه به سود ارائه شد. در این مقاله نیز از این چارچوب پیشنهادی استفاده می‌گردد.
در [۳۲]، الگوریتم‌های بهینه‌سازی گرگ خاکستری و بهینه‌سازی نهنگ بصورت چندهدفی برای حل مسئله انتشار نسخه بعد نرم‌افزار ارائه شد. نتایج این دو الگوریتم و سه الگوریتم بهینه سازی تکاملی دیگر، روی چهار مجموعه‌داده؛ براساس هشت شاخص کیفیت ارزیابی شدند. در این مقاله، سه جنبه از عدالت بین مشتریان و همچنین عدم قطعیت، به عنوان چهار شاخص کیفیت برای مسئله انتشار نسخه بعد نرم‌افزار، بازتعریف شد. نتایج نشان داد که الگوریتم چندهدفی بهینه‌سازی نهنگ بهتر از

سایرین عمل می‌کند.

در [۷]، تاثیر روابط و تعاملات بین نیازمندی‌ها در هنگام جستجوی پاسخ‌های مسئله انتشار نسخه بعد نرم‌افزار بررسی گردید. روابط بین نیازمندی‌ها را در دو الگوریتم (یک الگوریتم دقیق شاخه و کران و یک الگوریتم تخمین توزیع[۱۱] (EDA)، ادغام شدند. نتایج روی سه مجموعه‌داده، نشان دادند که استفاده از روابط نیازمندی‌ها برای ایجاد پاسخ مسئله، فرآیند جستجو را بهبود می‌بخشد.

در [۳۳]، با استفاده از سیستم استنتاج فازی و روش شاخه و کران، مدل بهینه‌سازی شاخه و کران فازی[۱۲] (FBBO)، برای مسئله انتشار نسخه بعد نرم‌افزار ارائه شد.

در [۶]، یک الگوریتم بهینه‌سازی چندهدفی جدید با نام الگوریتم بهینه‌سازی کرکس آفریقایی چندهدفی کوانتومی با ساختارهای سلسله مراتبی، معرفی شد، که در آن ساختار سلسله مراتبی و ایده‌های محاسبات درون کوانتومی برای بهبود عملکرد الگوریتم معرفی می‌شوند.

۳-مسئله انتشار نسخه بعدی چندهدفی

۳-۱- مفاهیم بهینه‌سازی چندهدفی

بهینه‌سازی چندهدفی، انتخاب بهترین پاسخ یا پاسخ‌ها با توجه به اهداف مختلف و معمولاً متضاد است. در بهینه‌سازی چندهدفی معمولا یک پاسخ بهترین وجود ندارد و یک مجموعه از پاسخ‌ها، بنام پاسخ‌های غیرمغلوب را جستجو می‌کنیم. معمولا بهبود یک تابع هدف منجر به بدترشدن یک یا بیشتر توابع هدف دیگر می‌گردد. در ادامه مفاهیم اصلی بهینه‌سازی چندهدفی آورده شده است [۱۷].

یک مسئله بهینه‌سازی چندهدفی را می‌توان بعنوان کمینه (بیشینه) همزمان چند تابع هدف، بصورت رابطه (۱)، تعریف کرد.

$$min\ F(x) = [f_1(x), f_2(x), ..., f_k(x)]$$
$$subject\ to: \begin{cases} x = [x_1, x_2, ..., x_n] \in X \\ h_i(x) \le g_j \ ; i = 1, 2, ..., p \end{cases} \quad (۱)$$

$\mathbf{x}$ یک پاسخ شدنی است، $\mathbf{X}$ مجموعه بردارهای شدنی است. پاسخ شدنی، پاسخی است که محدودیت‌های مسئله را برآورده می‌کند. f_i ها توابع هدف هستند. $\mathbf{x}$ برداری از n متغیر تصمیم است. $h_i(\mathbf{x})$، توابع محدودیت مسئله هستند. g_i مقادیر محدودیت‌ها هستند.

فرض کنیم، بردارهای $\mathbf{u}, \mathbf{v} \in \mathbf{X}$، دو پاسخ مختلف باشند، می‌گوییم که $\mathbf{u}$ بر $\mathbf{v}$ غلبه می‌کند (با $\mathbf{u} \preceq \mathbf{v}$ نشان می‌دهیم)، اگر و فقط اگر هیچ مقدار $f_i(\mathbf{u})$ بدتر از $f_i(\mathbf{v})$ نباشد و حداقل برای یک j، $f_j(\mathbf{u})$ بهتر از $f_j(\mathbf{v})$ باشد.

$$\mathbf{u} \preceq \mathbf{v} \Leftrightarrow \forall i \in \{1,2,\ldots,k\}, f_i(\mathbf{u}) \le f_i(\mathbf{v}) \text{ and } \exists j \in \{1,2,\ldots,k\}, f_i(\mathbf{u}) < f_i(\mathbf{v}) \quad (۲)$$

به پاسخ $\mathbf{u}$ یک بهینه پارتو یا غیرمغلوب گفته می‌شود؛ اگر و فقط اگر هیچ پاسخ $\mathbf{v}$ نباشد که بر آن غلبه کند. همه پاسخ‌های بهینه پارتو، مجموعه بهینه پارتو را می‌سازند ($Ps = \{\mathbf{u} \in \mathbf{X} \mid \sim\exists\, \mathbf{v} \in \mathbf{X} \text{ and } \mathbf{u} \preceq \mathbf{v}\}$). به تصویر مجموعه بهینه پارتو در فضای هدف، جبهه پارتو می‌گویند ($PF = \{\mathbf{F(x)} \mid \mathbf{x} \in Ps\}$).

۳-۲-تعریف چندهدفی مسئله انتشار نسخه بعدی

فرض کنید برای یک شرکت نرم‌افزاری، مجموعه مشتریان $C = \{c_1, \ldots, c_m\}$ را داریم. m تعداد مشتریان و c_i مشتری i ام می‌باشد. کل نیازمندی‌های ممکن برای انتشار نسخه بعد را با مجموعه $R = \{r_1, \ldots, r_n\}$ مشخص می‌کنیم. n تعداد نیازمندی‌ها و r_i، نیازمندی i ام می‌باشد. در تولید نرم‌افزار، برای پیاده‌سازی نیازمندی‌ها لازم است منابعی تخصیص داده شود. در مسئله NRP میزان منابع مورد نیاز برای تکمیل نیازمندی را با عددی بعنوان هزینه مشخص می‌کنیم. مجموعه هزینه‌های نیازمندی‌ها را با $Cost = \{cost_1, \ldots, cost_n\}$ مشخص می‌نماییم. $cost_i$ هزینه پیاده‌سازی r_i است. اهمیت و ارزش مشتریان برای شرکت نرم‌افزاری متفاوت است. اهمیت هر مشتری برای شرکت را با فاکتور وزنی معین می‌کنیم. مجموعه $W = \{w_1, \ldots, w_m\}$

وزن‌هایی است که به مشتریان نسبت داده می‌شود. w_j اهمیت مشتری j ام (c_j) است. w_j عددی بین صفر و یک است. مجموع w_j ها یک است.

نیازمندی‌ها درجه اهمیت یکسانی برای مشتریان ندارند. مشتری c_j ارزشی به نیازمندی r_i نسبت می‌دهد که با v_{ij} نشان می‌دهیم. اگر $v_{ij} > 0$ باشد آنگاه مشتری j ام، تقاضای نیازمندی i را دارد، در غیر این صورت مقدار آن صفر است. ماتریس V یک ماتریس m در n است که شامل v_{ij} هاست. براساس توضیحات بالا امتیاز کلی یا درجه اهمیت نیازمندی r_i برای شرکت بصورت زیر محاسبه می‌گردد:

$$score_i = \sum_{j=1}^{m} (w_j * v_{ij}) \quad (۳)$$

بردار تصمیم‌گیری، $\mathbf{x} = [x_1, \ldots, x_n] \in \{0,1\}^*$، نیازمندی‌هایی را تعیین می‌کند که در انتشار نسخه بعد پیاده‌سازی می‌شوند. اگر نیازمندی r_i انتخاب شده باشد، $x_i = 1$ و در غیر این صورت $x_i = 0$. هدف نهایی، تعیین بردار $\mathbf{x}$ به‌گونه‌ای است که رضایتمندی مشتریان را بیشینه و در عین حال هزینه را کمینه نگه دارد. در مسئله NRP چندهدفی (MONRP) توازن بین score و cost ایجاد می‌گردد. تابع هدف زیر جهت به حداکثر رساندن مجموع ارزش‌ها (رضایتمندی مشتریان) است:

$$max \quad \text{Satisfy}_i = \sum_{i=1}^{n} (\text{score}_i * x_i) \quad (۴)$$

تابع هدف دوم برای کمینه‌کردن مجموع هزینه‌های نیازمندی‌ها، در زیر آمده است:

$$min \quad \text{TotalCost}_i = \sum_{i=1}^{n} (\text{cost}_i * x_i) \leq B \quad (۵)$$

B سقف بودجه است و معمولا، کسری از مجموع کل هزینه‌ها در نظر گرفته می‌شود.

مدل اولیه این مسئله شبیه مسئله کوله‌پشتی صفر و یک است، یعنی زیر مجموعه‌ای از مشتریان انتخاب می‌شد و تمام نیازمندی‌های آن مشتریان با پیش‌نیازهایشان برآورده می‌شد. در مدل ارائه شده ممکن است کسری از نیازمندی‌های یک مشتری برآورده شود. بین نیازمندی‌ها در مجموعه‌داده‌های استفاده شده روابطی وجود دارد. این روابط را می‌توان از نظر معنایی گروه‌بندی کرد [۱۳]. در مدل اولیه که بگنال ارائه کرد، رابطه پیش‌نیازی بین نیازمندی‌ها وجود دارد [۵]. در صورتی یک نیازمندی پیاده‌سازی می‌شود که تمام پیش‌نیازهایش نیز پیاده‌سازی شوند. از یک گراف غیرچرخشی و جهت‌دار G(R,E) برای نمایش وابستگی بین نیازمندی‌ها استفاده می‌شود. رأس‌های گراف، نیازمندی‌ها را مشخص می‌کنند. یال‌ها نیز روابط بین نیازمندی‌ها را نشان می‌دهند. در مجموعه‌داده‌های استفاده شده در این تحقیق، سه نوع رابطه بین نیازمندی‌ها داریم.

- رابطه جفتی: رابطه جفتی بین r_i و r_j با $r_i \Theta r_j$ نشان داده می‌شود. نیازمندی‌های r_i و r_j باید یا با هم پیاده‌سازی شوند یا هیچ کدام پیاده‌سازی نگردند.

- رابطه انحصاری: رابطه انحصاری بین r_i و r_j با $r_i \otimes r_j$ نشان داده می‌شود. نیازمندی‌های r_i و r_j نمی‌توانند همزمان پیاده‌سازی شوند.

- رابطه پیش‌نیازی: رابطه پیش‌نیازی بین r_i و r_j با (r_i,r_j) نشان داده می‌شود. اگر r_i پیش‌نیاز r_j باشد، قبل از پیاده‌سازی r_j، باید r_i پیاده‌سازی شود. پیاده‌سازی r_j وابسته به r_i است.

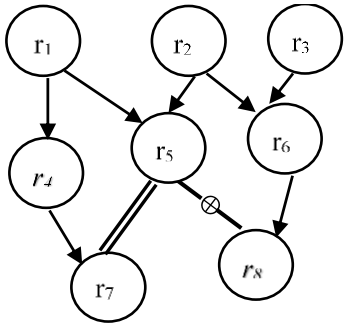
شکل (۱)، یک مثال از نیازمندی‌ها و روابط بین آن‌ها را نمایش می‌دهد. فرض کنید که هشت نیازمندی و سه مشتری داریم. هزینه نیازمندی‌ها را $\text{Cost}=\{۵,۳,۳,۲,۲,۱,۲,۲\}$ و وزن مشتریان را با $\text{W}=\{۰.۴,۰.۳,۰.۳\}$ در نظر می‌گیریم. ارزش هر نیازمندی برای هر مشتری را با عددی بین ۰ تا ۵ مشخص کرده‌ایم که در ماتریس زیر آمده است.

$$V = \begin{bmatrix} 4 & 3 & 2 & 2 & 4 & 1 & 3 & 5 \\ 5 & 3 & 4 & 1 & 3 & 1 & 4 & 2 \\ 3 & 3 & 0 & 3 & 5 & 1 & 2 & 4 \end{bmatrix}$$

روابط بین نیازمندی‌ها در مجموعه I نمایش داده شده‌است.

$$I=\{(r_1 , r_4),(r_1 , r_5),(r_1 , r_7), (r_2 , r_5),(r_2 , r_6), (r_2 , r_8), (r_3 , r_6), (r_3 , r_8), (r_4 , r_7), (r_6 , r_8), r_5 \Theta r_7, r_5 \otimes r_8 \}$$

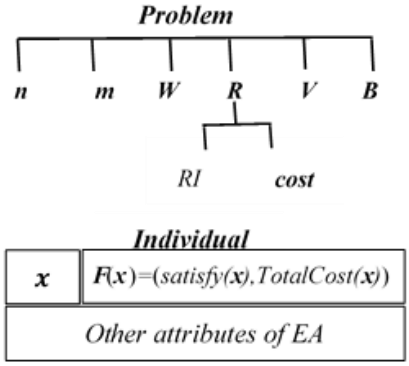
طبق رابطه (۱)، score هر نیازمندی را بدست می‌آوریم. $\text{Score} = \{۴, ۳, ۳, ۲, ۴, ۱, ۳,۳.۸\}$



شکل (۱): مثالی از روابط نیازمندی‌ها

۳-۳- ساختارهای داده برای مسئله انتشار نسخه بعد چندهدفی

در شکل (۲)، ساختارهای داده‌ای برای مسئله انتشار نسخه بعد چندهدفی، نمایش داده شده‌اند. اطلاعات مسئله در ساختار Problem نگهداری می‌شود. n تعداد نیازمندی‌ها و m تعداد مشتریان است. W بردار وزن مشتریان است. R برداری به طول n است که هزینه و روابط هر نیازمندی را نگهداری می‌کند. cost عددی معادل هزینه پیاده‌سازی نیازمندی است. RI روابط بین نیازمندی با سایر نیازمندی‌هاست. B سقف بودجه را مشخص می‌کند. V ماتریسی است که ارزش هر نیازمندی برای هر مشتری را مشخص می‌کند. در الگوریتم‌های تکاملی، جمعیتی از اعضا (individuals) را داریم. **x**، بردار تصمیم دودویی است و **F** برداری دو بعدی از مقادیر تابع هدف است. پارامترهای الگوریتم تکاملی مربوطه نیز در آن قرار می‌گیرند.



شکل (۲): ساختارهای داده برای مسئله چندهدفی انتشار نسخه بعد

۴- عملگرهای چند هدفی

در اینجا عملگرهایی که در ادامه برای پیاده‌سازی چندهدفی الگوریتم بهینه‌سازی زیستی استفاده شده‌است، تشریح می‌گردد.

۴-۱- مرتب‌سازی غیر مغلوب

اعضای جمعیت را براساس سطح غیرمغلوب بودن، به چند جبهه (Front) تقسیم می‌کنیم. اعضایی که در جبهه اول قرار دارند، توسط هیچ یک از پاسخ‌ها مغلوب نشده‌اند (هیچ کدام از پاسخ‌های دیگر بر اعضای جبهه اول، غلبه نمی‌کنند)، اعضایی که فقط توسط حداقل یکی از اعضای جبهه اول مغلوب می‌شوند، در جبهه دوم قرار می‌گیرند و اعضایی که فقط توسط حداقل یکی از پاسخ‌های جبهه دوم مغلوب می‌شوند، در جبهه سوم قرار می‌گیرند، به همین ترتیب سایر جبهه‌ها ساخته می‌شوند. این فرایند را مرتب‌سازی غیر مغلوب، می‌نامند [۳۴].

۴-۲- فاصله ازدحامی

فاصله ازدحامی یک معیار برای پراکندگی است که برای اعضای یک جبهه محاسبه می‌گردد[۳۵]. از آن‌جا که اعضای جبهه، همدیگر را مغلوب نمی‌کنند، برای مقایسه پاسخ‌ها نیاز به معیار دیگری نیز داریم. براساس هرکدام از توابع هدف می‌توان اعضا را مرتب کرد، بطوریکه عضو قبل و بعد هر عضو مشخص می‌گردد. فاصله ازدحامی طبق رابطه (۶)، محاسبه می‌شود. فاصله ازدحامی اولین و آخرین عضو را بینهایت در نظر می‌گیریم.

$$Crowd(P_i)=\sum_{j=1}^{d}\frac{\left|P_{i+1}.Fitness_j - P_{i-1}.Fitness_j\right|}{max(Fitness_j)-min(Fitness_j)} \qquad (۶)$$

P از تمام اعضای یک جبهه تشکیل شده است که براساس یکی از توابع هدف به دلخواه، مرتب شده است. P_i ، i امین عضو جمعیت P است. $Fitness_j$، j امین تابع هدف است. d تعداد توابع هدف است. در واقع، فاصله ازدحامی هر عضو، فاصله منهتن عضو بعدی و قبلی، است.

۴-۳- عملگر رتبه‌بندی

این عملگر براساس مرتب‌سازی غیرمغلوب و فاصله ازدحامی، اعضای جمعیت را رتبه‌بندی و مرتب می‌کند. ابتدا با عملگر مرتب‌سازی غیرمغلوب، جبهه‌ها ایجاد می‌گردند. سپس فاصله ازدحامی برای اعضای هر جبهه محاسبه می‌شود. اعضای جبهه اول رتبه بالاتری از اعضای جبهه دوم دارند و اعضای جبهه دوم رتبه بالاتری از اعضای جبهه سوم دارند و ... . بین اعضای یک جبهه، عضوی که فاصله ازدحامی بیشتری دارد، رتبه بالاتری دارد.

۴-۴- عملگر حذف اعضای تکراری جمعیت

هنگام ایجاد جمعیت اولیه و همچنین پس از اعمال عملگرهای تکاملی، ممکن است اعضای تکراری تولید شوند. اگر این اعضای تکراری پاسخ‌های غیرمغلوب باشند، در جمعیت اصلی باقی می‌مانند و در پایان، خروجی الگوریتم شامل اعضای تکراری است. برای جلوگیری از این رخداد، از جمعیت با اعضای غیرتکراری استفاده می‌گردد. در این مقاله از رویکردهای جمعیت غیرتکراری و ادغام پاسخ‌ها که در [۳۱] ارائه شده‌است، استفاده می‌گردد. اعضای جمعیت از نظر موقعیت (بردار تصمیم) مقایسه می‌شوند و در صورت تکراری بودن، اعضای تکراری حذف می‌شوند. در نتیجه جمعیتی با اعضای یکتا، بدست می‌آید. استفاده از جمعیتی با اعضای غیرتکراری، باعث می‌شود که تنوع متغیر تصمیم تولیدشده، افزایش یابد و احتمال پیمایش فضای بزرگتر مسئله افزایش یابد. همچنین پاسخ‌های بهینه پارتو متفاوتی که تابع هدف یکسانی دارند، حذف نمی‌شوند.

۴-۵- ادغام جمعیت‌ها

درالگوریتم‌های تکاملی، عملگرهای تکاملی مانند تقاطع و جهش، بر موقعیت اعضای فعلی جمعیت اعمال می‌گردند و موقعیت جدیدی بدست می‌آید. در BBO استاندارد، اعضای جدیدی که با عملگرهای مهاجرت و جهش ایجاد می‌شوند، جایگزین عضو متناظر فعلی می‌شوند. البته اعضای نخبه جمعیت قبلی باقی می‌مانند. رویکرد دیگر؛ این است که از موقعیت‌های ایجادشده جدید، جمعیتی جداگانه ساخت و با جمعیت فعلی ادغام کرد.

عملگر ادغام جمعیت‌ها، اعضای چند جمعیت را با هم ادغام کرده و سپس اعضای تکراری جمعیت را حذف می‌کند. در نتیجه، جمعیتی از اعضای غیرتکراری ایجاد می‌شود. در الگوریتم BBO، با استفاده از موقعیت‌هایی که با عملگرهای مهاجرت و جهش ایجاد می‌شوند، جمعیت جدیدی ایجاد می‌شود.

۵- الگوریتم چندهدفی بهینه‌سازی جغرافیای زیستی با جمعیت براونی

الگوریتم BBO یک الگوریتم تکاملی براساس مهاجرت جانوران از یک زیستگاه به زیستگاه دیگر است که در سال ۲۰۰۸ معرفی شد [۳۶]. زیستگاه‌هایی که برای اقامت گونه‌های زیستی مناسب‌تر، شاخص تناسب زیستگاه[۱۳] (HSI) بالاتری دارند. متغیرهای مشخص کننده‌ی کیفیت محل سکونت، متغیرهای شاخص شایستگی[۱۴] (SIV) نامیده می‌شوند. SIV را می‌توان

بعنوان متغیر تصمیم و HSI را تابع هدف در نظر گرفت. برای هر زیستگاه، نرخ مهاجرت به زیستگاه و نرخ مهاجرت از زیستگاه را در نظر می‌گیریم. زیستگاه i دارای نرخ مهاجرت به زیستگاه λ_i و نرخ مهاجرت از زیستگاه، μ_i است. λ و μ رابطه عکس با هم دارند. با افزایش تعداد گونه‌ها، زیستگاه شلوغتر شده و ممکن است گونه‌های کمتری به آنجا مهاجرت کنند(λ کاهش یابد) و برعکس احتمال مهاجرت از این زیستگاه به زیستگاه دیگر افزایش می‌یابد(μ افزایش یابد). زیستگاه‌ها ممکن است در اثر بیماری، بلایای طبیعی و ... تغییرات ناگهانی کنند. تغییرات ناگهانی را می‌توان با جهش مدل کرد. روابط و بحث بیشتر برای BBO اولیه در [۳۶] آمده است.

الگوریتم چندهدفی BBO نیز ارائه شده است [۳۸،۳۷]. در این تحقیق با رویکردی جدید نسخه چندهدفی الگوریتم بهینه-سازی مبتنی بر جغرافیای زیستی را ارائه می‌دهیم. تاکنون از این الگوریتم برای حل NRP استفاده نشده است. از غلبه پارتو و عملگرهای معرفی شده در بخش قبل، برای چندهدفی کردن BBO استفاده می‌گردد. ابتدا یک جمعیت اولیه تصادفی غیرتکراری (Pop) با N زیستگاه ایجاد می‌شود. در هر مرحله، با عملگرهای مهاجرت و جهش، جمعیتی جدید (BBoPop) ایجاد می‌گردد. با حرکت براونی نیز جمعیت براونی BrownianPop تولید می‌شود. این جمعیت‌ها با هم ادغام شده و اعضای تکراری حذف می‌شوند. سپس با مرتب‌سازی غیرمغلوب و فاصله ازدحامی، اعضای جمعیت طبق عملگر رتبه‌بندی در بخش ۳-۴. مرتب می‌شوند و N عضو با رتبه بالاتر نگهداری شده و در تکرار بعد الگوریتم استفاده می‌شوند. در هر تکرار الگوریتم، پاسخ‌های غیرمغلوب در یک مخزن نگهداری می‌شوند. خروجی الگوریتم جبهه پارتو بدست آمده در این مخزن است. این مخزن تاثیری در عملگرهای الگوریتم ندارد و صرفا شامل همه پاسخ‌های غیرمغلوب است که در تکرارهای مختلف الگوریتم بدست می‌آیند. الگوریتم مجددا تا رسیدن به نقطه توقف تکرار می‌گردد.

در شکل (۳)، شبه‌کد الگوریتم چند هدفی BBO برای مسئله انتشار نسخه بعدی نرم‌افزار، نمایش داده شده است. PM از نوع ساختار Problem است که اطلاعات مسئله را نگهداری می‌کند. N اندازه جمعیت وD ابعاد بردار تصمیم X است. تابع generateRndU، جمعیتی N عضوی غیرتکراری (بردار تصمیم X غیرتکراری) می‌سازد که محدودیت‌های مسئله را بر آورده می‌کنند.

```
Inputs:  PM: problem, D: number of Requirements,
         N: Population Size, MaxNFE: max of Fitness Evaluation
Output: Archive
    Pop.X = generateRndU (N, D, PM)
    Pop. Fitness = evaluate (Pop)
    Archive = [ ]
  While (NFE < MaxNFE)
    BBoPop = GenerateBboPop (Pop, PM)
    BrownianPop = Generatebrownianpop (Pop, PM)
    Pop = mergeU (Pop, newPop, BrownianPop)
    Archive = nonDominated (mergeU (Archive, Pop))
   Pop = Ranking (Pop)
   Pop = Pop (1: N)
 End
```

شکل (۳): شبه کد الگوریتم چندهدفی پیشنهادی

تابع evaluate، اعضای جمعیت را براساس توابع هدف ارزیابی می‌کند تا بردار تابع هدف Fitness بدست آید. تابع GenerateBboPop جمعیت BBoPop را با عملگرهای مهاجرت و جهش الگوریتم BBO می‌سازد. در ادامه این تابع تشریح می-گردد. تابع Generatebrownianpop جمعیت براونی را ایجاد می‌کند. تابع mergeU، جمعیت‌ها را ادغام کرده و اعضای تکراری را حذف می‌کند. در هر تکرار الگوریتم، پاسخ‌های غیرمغلوب جمعیت با تابع nonDominated بدست می‌آید. Archive مخزن بدون محدودیت اندازه است که برای ذخیره جبهه پارتو استفاده می‌شود. دستور nonDominated (Merge (Archive, Pop))،

Archive را به‌روزرسانی می‌کند و بهترین جبهه پارتو را که تا تکرار فعلی بدست آمده است در آن قرار می‌دهد. تابع Ranking، جمعیت را مرتب می‌کند. N عضو اول با رتبه بالاتردر Pop حفظ شده و در تکرار بعد از آن استفاده می‌گردد. الگوریتم مجددا تا رسیدن به نقطه توقف تکرار می‌شود. برای مقایسه منصفانه با دیگر الگوریتم‌ها، شرط توقف الگوریتم، با تعداد فراخوانی‌های تابع هدف[۱۵] (NFE) تعیین می‌شود.

خروجی الگوریتم جبهه پارتو بدست آمده در Archive است. این مخزن تاثیری در عملگرهای الگوریتم ندارد و صرفا شامل همه پاسخ‌های غیرمغلوب است که در تکرارهای مختلف الگوریتم بدست می‌آیند.

۵-۱- جمعیت جغرافیای زیستی

جمعیت جغرافیای زیستی، جمعیت ایجاد شده با عملگرهای مهاجرت و جهش در الگوریتم BBO است. در مسئله انتشار نسخه بعد، SIV، بردار تصمیم X و HSI، تابع هدف Fitness است. BBO الگوریتم تکاملی مبتنی بر جمعیت است. هر زیستگاه یک عضو جمعیت است. فرض می‌کنیم که N زیستگاه (یعنی جمعیتی با N عضو) داریم. D ابعاد بردار تصمیم X است که در اینجا برابر با تعداد نیازمندی‌هاست. با توجه به اینکه در مسئله انتشار نسخه بعدی نرم‌افزار، متغیر تصمیم دودویی است، مختصات هر عضو برداری از صفر و یک است. زیستگاه i ام، با Pop_i مشخص می‌شود.

موقعیت زیستگاه‌ها، طبق رابطه (۷)، بصورت تصادفی مقداردهی اولیه می‌شوند و با توابع هدف ارزیابی می‌شوند.

$$Pop_i.X = Rand(1,D) < rand \quad (۷)$$

تابع Rand(1,D)، D عدد تصادفی بین صفر و یک تولید می‌کند. پس از مقایسه با rand، به صفر یا یک تبدیل می‌شوند.

به احتمال λ_i، عملگر مهاجرت برای موقعیت هر عضو جمعیت، اعمال می‌شود. λ_i برای زیستگاه i ام، ازرابطه (۸) تعیین می‌شود.

$$\lambda_i = \frac{i-1}{N-1} \quad (۸)$$

در ادامه، باید مشخص شود که مهاجرت از زیستگاه فعلی به سوی کدام زیستگاه انجام گیرد. بصورت احتمالی و با چرخه رولت، زیستگاهی که مهاجرت از زیستگاه فعلی به سوی آن انجام می‌گیرد، انتخاب می‌شود. قبلا ذکر شد که مسئله انتشار نسخه بعد، دو تابع هدف دارد. احتمال مهاجرت به هریک از زیستگاه‌ها طبق رابطه (۹)؛ براساس نسبت توابع هدف یعنی Satisfy به TotalCost ساخته می‌شوند.

$$EP_i = \frac{\frac{Satisfy_i}{TotalCost_i}}{\sum_{j=1}^{n} \frac{Satisfy_j}{TotalCost_j}} \quad (۹)$$

Satisfy که باید افزایش یابد، در صورت و TotalCost که باید کاهش یابد، در مخرج قرار گرفته‌است، تا احتمال مهاجرت به زیستگاهی که توابع هدف بهتری دارد، افزایش یابد. برای اینکه احتمال‌ها باید در محدوده صفر و یک باشند، نسبت بدست آمده، بر مجموع نسبت‌ها تقسیم شده‌اند. در واقع از نسبت Satisfy به TotalCost برای مشخص کردن μ زیستگاه‌ها استفاده شده‌است. طبق الگوریتم BBO استاندارد، احتمال اینکه ویژگی‌های زیستگاه i تغییر کند، متناسب با λ_i است و احتمال اینکه منبع تغییر، زیستگاه j باشد، متناسب با μ_j است.

$$\mu_i = \frac{Satisfy_i}{TotalCost_i} \quad (۱۰)$$

Pop_i، با رابطه (۱۱) به سوی Pop_j مهاجرت می‌کند:

$$newPop_i.X = Pop_i.X + \alpha * (Pop_j.X - Pop_i.X) \quad (۱۱)$$

α پارامتر کنترلی است. موقعیت جدید در بردار تصمیم $newPop_i$ قرار می‌گیرد.

سپس با احتمال P_{mu}، طبق رابطه (۱۲)، عملگر جهش روی جمعیت Pop عمال می‌گردد.

$$newPop_i.X = NewPop_i.X + \sigma * randn \quad (۱۲)$$

randn یک عدد تصادفی با توزیع نرمال استاندارد تولید می‌کند. σ پارامتر کنترلی است.

newPop جمعیت با N زیستگاه است که موقعیت‌های جدید ایجاد شده پس از عملگرهای مهاجرت و جهش را ذخیره می‌کند.

بردارهای X بدست آمده از عملگرهای مهاجرت و جهش دودویی نیستند. الگوریتم BBO استاندارد، الگوریتم پیوسته است. درحالیکه در مسئله انتشار نسخه بعدی نرم‌افزار، بردار تصمیم دودویی است و لذا باید موقعیت زیستگاه‌ها را دودویی کرد. در اینجا از تابع Sigmoidal استفاده می‌شود. این تابع مقادیر را به محدوده صفر ویک نگاشت می‌کند. سپس با مقایسه حاصل تابع با یک عدد تصادفی، به صفر یا یک تبدیل می‌شود. در اینجا x هر بعد از بردار تصمیم X است.

$$Sigmoid(x) = \frac{1}{1+e^{-10(x-0.5)}} \quad (۱۳)$$

سپس محدودیت‌های روابط نیازمندی و بودجه برآورده می‌گردد.

در شکل (۴)، شبه کد تولید جمعیت با عملگرهای الگوریتم BBO آمده است.

```
Function GenerateBboPop (Pop, Problem)
    N = Size (Pop)
    D = Problem.D
    λ = linspace (0, 1, N)
    PMu =1/ Problem.D
    EP = CalcEmigrationProbability (Pop)
    NewPop = Pop
    For i = 1: N
        For k = 1: D
            if rand ≤ λi
                j = selectHabitat (EP, i)
                newPopi.Xk = Popi. Xk + α . (Popj.Xk - Popi. Xk)
            end
            if rand ≤ PMu
                newPopi.Xk = Mutate (newPopi. Xk)
            end
        end
        newPopi.X = SatisfyConstraints (newPopi.X, PM)
        newPopi.Fitness = Evaluate (newPopi)
    end
  Return newPop
End

Function CalcEmigrationProbability (Pop)
    For j =1: N
      EP (j) = Popj.Satisfy / Popj.TotalCost
    end
   EP = EP / sum (EP)
   Return EP
End

Function SelectHabitat (EP, i)
  EP (i) = 0
  j = RouletteWheelSelection (EP)
  Return j
End
```

شکل (۴): شبه‌کد تولید جمعیت با عملگرهای مهاجرت و جهش الگوریتم جغرافیای زیستی

تابع CalcEmigrationProbability احتمال مهاجرت به زیستگاه‌ها را طبق رابطه (۹)، برای اعضای جمعیت، محاسبه می‌کند.

تابع SelectHabitat طبق احتمال‌های بدست آمده، زیستگاه مقصد (Pop_j) را برای مهاجرت زیستگاه مبدا Pop_i، انتخاب می‌کند. تابع Mutate عمل جهش را انجام می‌دهد. تابع SatisfyConstraints، پس از دودویی‌کردن بردار تصمیم، محدودیت روابط بین نیازمندی‌ها و محدودیت بودجه را بررسی می‌کند و اگر پاسخ نشدنی باشد، آن را به پاسخ ممکن که محدویت‌ها را برآورده می‌کند تبدیل می‌کند.

۵-۲- جمعیت براونی

الگوریتم جستجوی تفاضلی، یک الگوریتم تکاملی است که حرکت تصادفی شبه براونی را که یک ارگانیسم برای مهاجرت استفاده می‌کند، شبیه سازی می‌کند [۳۹]. گونه‌های جانداران به علت تغییر آب و هوایی در سال، مهاجرت فصلی می‌کنند. گونه‌های مهاجر هنگام مهاجرت، یک ابرارگانیسم تشکیل می‌دهند. ابرارگانیسم‌ها به سوی نواحی حاصلخیز تغییر مکان می‌دهند. حرکت و جابجایی آن‌ها را با حرکت تصادفی شبه براونی می‌توان توصیف کرد. ابرارگانیسم‌ها در مکان‌های جدید برای مدتی موقت می‌مانند و سپس مهاجرتشان را ادامه می‌دهند تا نواحی بهتری را بیابند. در شکل (۵)، شبه کد حرکت براونی آورده شده‌است.

```
Function BR = Brownian-motion (N, D)
    p1 = 0.3* rand;
    p2 = 0.3* rand;
  if rand < rand
   if rand < p1
       BR = rand (N, D)
      For i =1: N
        BRi = BRi < rand
      end
    else
       BR = ones (N, D)
      For i =1: N
        BRi, Randi (D) = BRi, Randi (D) < rand
      end
    end
  else
     BR = ones (N, D)
    For k =1: N
       d = Randi (n, 1, ceil (p2 * D))
       BRk,d = 0
     end
  end
End
```

شکل (۵): شبه کد حرکت براونی

با حرکت براونی، ماتریس دودویی BR ایجاد می‌گردد. این ماتریس N سطر و D ستون دارد و با اعداد تصادفی با توزیع یکنواخت ساخته می‌شود. D ابعاد مسئله و N اندازه جمعیت است. طبق الگوریتم حرکت براونی، درایه‌های ماتریس BR با مقداردهی تصادفی بین صفر و یک تعیین می‌شوند و سپس به صفر یا یک تبدیل می گردند یا در ابتدا مقدار تمام عناصر BR یک می‌شوند و ستون‌هایی تصادفی از هر سطر BR تغییر می‌کند. ستون‌هایی از هر سطر ماتریس BR که یک هستند، در تغییر موقعیت اعضای جمعیت موثرند. Rand (N, D)، D ماتریسی N*D از اعداد تصادفی بین صفر و یک تولید می‌کند که هر کدام از آن‌ها، پس از مقایسه با عدد تصادفی rand به صفر یا یک تبدیل می‌شوند.

Randi (D)، یک عدد صحیح تصادفی بین صفر و D تولید می‌کند.

شبه کد تولید جمعیت با حرکت براونی، در شکل (۶)، آمده است.

```
Function GenerateBrownianPop (Pop, Problem)
    donor = Random_shuffling (Pop)
    BrownianPop = Pop
  For i =1: N
    BR = Brownianmotion (N, D)
    γ = 1 ./ Gamrnd (a, b)
    BrownianPop_i.X = Pop_i. X+ γ. BR_i . (donor_i .X- Pop_i. X)
    BrownianPop_i.X = SatisfyConstraints (BrownianPop_i.X, Problem)
    BrownianPop_i. Fitness = Evaluate (BrownianPop_i .X)
  end
  Return BrownianPop
End
```

شکل (۶): شبه کد تولید جمعیت براونی

با به هم ریختن ترتیب اعضا، جمعیتی از همان اعضا ولی با ترتیب متفاوت بنام donor ایجاد می‌کنیم. donor شامل جمعیت Pop است که ترتیب اعضایش بطور تصادفی تغییر یافته است (Random_shuffling). هر عضو جمعیت به سمت عضو متناظرش در donor حرکت می‌کند تا موقعیت‌های جمعیت براونی را ایجاد کنند.

$$BrownianPop_i.X = Pop_i.X + \gamma \bullet BR_i \bullet (donor_i.X - Pop_i.X) \quad (۱۴)$$

BR_i سطر i ماتریس BR است. عملگر '•' ضرب مولفه‌ای است. γ تغییر مکان را کنترل می‌کند و از رابطه (۱۵)، بدست می‌آید.

$$\gamma = \frac{1}{Gamrnd(a,b)} \quad (۱۵)$$

Gamrnd عدد تصادفی را طبق توزیع گاما تولید می‌کند. معمولا a = 1 و b = 0.5 در نظر گرفته شده است[۳۹].

با تابع SatisfyConstraints، موقعیت‌های جمعیت براونی (X)، برای برآورده‌کردن محدودیت‌های مسئله بررسی شده و تبدیل به بردار تصمیم دودویی قابل قبول می‌شوند. بردار تابع هدف (Fitness) هر عضو نیز با ارزیابی بردار تصمیم آن عضو با تابع Evaluate بدست می‌آید.

۶- آزمایش‌ها

تمام آزمایش‌ها با کامپیوتر در برنامه متلب ۲۰۱۶ و در محیط ویندوز۱۰، ۶۴ بیتی انجام شده است. از نظر سخت‌افزاری از Intel Core i7-7700HQ 2.8 GHz Processor with 16 GB RAM استفاده کرده‌ایم. از آن‌جا که با الگوریتم‌های تصادفی درگیر هستیم، برای کاهش تاثیر پارامترها، برای هر آزمایش، الگوریتم‌ها را بطور مستقل ۱۰۰ بار اجرا می‌کنیم. برای مقایسه عادلانه الگوریتم‌ها از تعداد فراخونی تابع هدف (NFE)، برای توقف هر بار اجرای مستقل الگوریتم‌ها استفاده می‌کنیم و حداکثر NFE را ۱۰۰۰۰ قرار می‌دهیم [۱۳].

پارامترهای الگوریتم‌ها را مطابق با تحقیق دیگران تنظیم می‌کنیم. میانگین و انحراف معیار نتایج این اجراها در ادامه ارائه می‌گردد. برای تست کارایی الگوریتم‌های ارائه‌شده، دو مجموعه‌داده استفاده کرده‌ایم. برای هر مجموعه‌داده، چهار محدودیت بودجه ۳۰ و ۵۰ و۷۰ و ۱۰۰ درصد مجموع هزینه نیازمندی‌هایش را در نظر گرفته‌ایم. در واقع ۸ نمونه مسئله NRP داریم.

۶-۱- شاخص‌های کیفی

در این تحقیق از سه شاخص کیفیت؛ ابرحجم، گستردگی و تعداد پاسخ‌های غیرمغلوب؛ برای مقایسه کارایی الگوریتم‌ها استفاده شده است. جزئیات این سه شاخص کیفیت در [۴۰] آمده است. جبهه پارتو بدست‌آمده با الگوریتم را PF_{Known} می‌نامیم. PF_{True}

جبهه پارتو بهینه است. در مسائل بزرگ که محاسبه جبهه پارتو بهینه سخت است، از اجتماع جبهه‌های پارتو بدست‌آمده با الگوریتم‌های پیاده‌سازی‌شده و جدا کردن پاسخ‌های غیرمغلوب، جبهه پارتو مرجع ($PF_{reference}$) بدست می‌آید.
نسبت ناحیه پوشش داده شده با جبهه پارتو بدست آمده PF_{Known} به فضای هدف را ابرحجم یا HV می‌نامیم که با رابطه زیر محاسبه می‌گردد:

$$HV \triangleq \left\{ \bigcup volume(x) \mid x \in PF_{known} \right\} \quad (۱۶)$$

در NRP که تابع برازش دوهدفی داریم، HV برابر با اجتماع مساحت مستطیل‌هایی است که با توابع هدف و نقطه‌های مرجع، حاصل می‌شوند. کمترین مقادیر تابع هدف نقطه مرجع کمینه و بیشترین مقادیر تابع هدف، نقطه مرجع بیشینه است. توابع هدف را نرمال می‌کنیم. مقادیر بزرگتر HV مطلوب است. HV، پراکندگی و همگرایی جبهه پارتو را اندازه گیری می‌کند.
گستردگی، میزان توزیع و گسترش پاسخ‌های جبهه پارتو بدست‌آمده PF_{Known} را اندازه‌گیری می‌کند و با رابطه (۱۷) محاسبه می‌شود.

$$\Delta = \frac{\sum_{k=1}^{M} d_k + \sum_{j=1}^{N-1} \left| d_i - \bar{d} \right|}{\sum_{k=1}^{M} d_i + (N-1).\bar{d}} \quad (۱۷)$$

M تعداد توابع هدف، k شماره تابع هدف، d_k فاصله اقلیدسی پاسخ‌های جبهه پارتو بدست‌آمده با نقاط مرزی هستند. N تعداد پاسخ‌های جبهه پارتو بدست آمده است. d_j فاصله اقلیدسی بین پاسخ‌های متوالی در جبهه پارتو است. $\bar{d}$ میانگین d_j هاست. هرچه گستردگی کمتر باشد، تنوع بهتر است.
شاخص کیفی دیگر، NDS یعنی تعداد پاسخ‌های جبهه پارتو بدست‌آمده با الگوریتم است. مقادیر بیشتر NDS مطلوب است.

۶-۲- مجموعه‌های داده

از دو مجموعه‌داده استاندارد که در کارهای دیگران زیاد استفاده شده است، استفاده می‌کنیم. اولین مجموعه‌داده دارای ۵ مشتری و ۲۰ نیازمندی است که بوسیله گریر و روهه معرفی شده است [۲۲]. ارزش هر نیازمندی از نظر هر مشتری، با عددی صحیح بین ۱ تا ۵ مشخص شده است. هزینه هر نیازمندی عددی بین ۱ تا ۱۰ است. بین نیازمندی‌ها روابط پیش‌نیازی و جفتی وجود دارد. در جدول (۱)، هزینه نیازمندی‌ها، اهمیت هر نیازمندی برای هر مشتری و روابط بین نیازمندی‌ها نمایش داده شده است. مجموعه‌داده دوم، مقیاس بزرگتر از مجموعه‌داده اول است. مجموعه‌داده دوم بوسیله ساگرادو معرفی شد [۱۳]. شامل ۱۰۰ نیازمندی و ۵ مشتری است. هزینه نیازمندی‌ها عددی بین ۱ تا ۲۰ است. ارزش هر نیازمندی برای هر مشتری با اعداد ۱ تا ۳ مشخص شده‌اند. در جدول (۲)، هزینه نیازمندی‌ها، ارزش و اهمیت هر نیازمندی برای هر مشتری و روابط بین نیازمندی‌ها نمایش داده شده‌اند. وزن و اهمیت مشتریان هر دو مجموعه‌داده، در جدول (۳)، آمده است.

جدول (۱): ارزش، هزینه و روابط بین نیازمندی‌ها در مجموعه داده گریر

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}	r_{11}	r_{12}	r_{13}	r_{14}	r_{15}	r_{16}	r_{17}	r_{18}	r_{19}	r_{20}
C_1	۴	۲	۱	۲	۵	۵	۲	۴	۴	۴	۲	۳	۴	۲	۴	۴	۴	۱	۳	۲
C_2	۴	۴	۲	۲	۴	۵	۱	۴	۴	۵	۲	۳	۲	۴	۴	۲	۳	۲	۳	۱
C_3	۵	۳	۳	۳	۴	۵	۲	۴	۴	۴	۲	۴	۱	۵	۴	۱	۲	۳	۳	۲
C_4	۴	۵	۲	۳	۳	۴	۲	۴	۲	۳	۵	۲	۳	۲	۴	۳	۵	۴	۳	۲
C_5	۵	۴	۲	۴	۵	۴	۲	۴	۵	۲	۴	۵	۳	۴	۴	۱	۱	۲	۴	۱
Cost	۱	۴	۲	۴	۳	۲	۱۰	۲	۱	۳	۲	۵	۸	۲	۱	۴	۱۰	۴	۸	۴
Interactions	$\{(۴,۸),(۴,۱۷),(۸,۱۷),(۹,۳),(۹,۶),(۹,۱۲),(۹,۱۹),(۱۱,۱۹)\}$, $r_3 \Theta r_{12}$, $r_{11} \Theta r_{13}$																			

جدول (۲): ارزش، هزینه و روابط بین نیازمندی‌ها در در مجموعه داده ساگرادو

r_{20}	r_{19}	r_{18}	r_{17}	r_{16}	r_{15}	r_{14}	r_{13}	r_{12}	r_{11}	r_{10}	r_9	r_8	r_7	r_6	r_5	r_4	r_3	r_2	r_1	
۳	۱	۳	۲	۲	۳	۲	۳	۱	۱	۳	۱	۱	۳	۳	۲	۱	۱	۲	۱	C_1
۱	۳	۳	۲	۳	۱	۲	۳	۳	۲	۱	۲	۲	۱	۲	۱	۲	۱	۲	۳	C_2
۳	۳	۲	۲	۱	۳	۱	۳	۳	۳	۲	۲	۳	۱	۱	۱	۲	۱	۱	۱	C_3
۳	۳	۱	۲	۳	۲	۳	۱	۲	۳	۲	۳	۲	۳	۱	۳	۱	۲	۲	۳	C_4
۱	۱	۱	۱	۱	۲	۱	۳	۲	۲	۱	۱	۳	۲	۱	۳	۱	۳	۲	۱	C_5
۳	۹	۲	۱۶	۴	۹	۲۰	۱۶	۱۲	۱۵	۱۸	۶	۱۰	۸	۱۵	۱۹	۷	۱۶	۱۹	۱۶	Cost
r_{40}	r_{39}	r_{38}	r_{37}	r_{36}	r_{35}	r_{34}	r_{33}	r_{32}	r_{31}	r_{30}	r_{29}	r_{28}	r_{27}	r_{26}	r_{25}	r_{24}	r_{23}	r_{22}	r_{21}	
۲	۳	۳	۱	۲	۲	۱	۲	۳	۲	۲	۱	۳	۳	۳	۳	۱	۱	۱	۲	C_1
۳	۳	۲	۳	۲	۱	۲	۲	۳	۱	۳	۳	۲	۲	۱	۳	۲	۳	۳	۳	C_2
۲	۱	۳	۱	۱	۲	۲	۱	۲	۳	۳	۳	۱	۳	۳	۲	۳	۲	۱	۲	C_3
۱	۱	۳	۲	۲	۳	۲	۲	۲	۱	۱	۱	۱	۲	۳	۳	۲	۱	۱	۱	C_4
۱	۱	۱	۱	۲	۲	۳	۳	۱	۱	۳	۲	۳	۲	۲	۳	۳	۳	۱	۱	C_5
۵	۱۸	۱۲	۸	۱۶	۲	۶	۱۷	۱	۵	۱۱	۹	۲۰	۸	۱۵	۷	۲	۴	۱۰	۲	Cost
r_{60}	r_{59}	r_{58}	r_{57}	r_{56}	r_{55}	r_{54}	r_{53}	r_{52}	r_{51}	r_{50}	r_{49}	r_{48}	r_{47}	r_{46}	r_{45}	r_{44}	r_{43}	r_{42}	r_{41}	
۱	۳	۱	۳	۱	۲	۳	۱	۳	۳	۳	۳	۲	۲	۱	۱	۱	۳	۲	۲	C_1
۱	۱	۲	۳	۳	۲	۲	۱	۳	۳	۳	۱	۲	۲	۲	۳	۱	۱	۳	۳	C_2
۳	۲	۱	۲	۳	۲	۳	۲	۱	۳	۲	۳	۱	۳	۳	۳	۳	۱	۳	۱	C_3
۳	۲	۱	۳	۳	۱	۲	۳	۱	۲	۲	۳	۱	۱	۲	۱	۳	۱	۱	۳	C_4
۲	۳	۱	۲	۱	۳	۲	۳	۳	۱	۲	۲	۳	۳	۲	۱	۲	۱	۱	۳	C_5
۷	۱۸	۱۶	۱۴	۳	۱	۸	۱۷	۲	۶	۶	۶	۶	۱۶	۹	۱۴	۲۰	۱۵	۱۴	۶	Cost
r_{80}	r_{79}	r_{78}	r_{77}	r_{76}	r_{75}	r_{74}	r_{73}	r_{72}	r_{71}	r_{70}	r_{69}	r_{68}	r_{67}	r_{66}	r_{65}	r_{64}	r_{63}	r_{62}	r_{61}	
۱	۳	۳	۲	۱	۱	۳	۲	۳	۱	۳	۲	۳	۱	۳	۱	۳	۳	۲	۲	C_1
۲	۱	۳	۳	۱	۲	۳	۱	۳	۱	۱	۳	۲	۱	۲	۱	۳	۲	۳	۱	C_2
۲	۱	۳	۳	۲	۱	۱	۳	۱	۳	۱	۳	۳	۳	۱	۳	۳	۲	۱	۱	C_3
۱	۳	۳	۱	۲	۲	۳	۳	۳	۲	۱	۲	۱	۲	۱	۳	۳	۳	۲	۲	C_4
۳	۱	۳	۲	۳	۱	۲	۳	۲	۲	۱	۲	۱	۲	۳	۱	۲	۱	۲	۲	C_5
۲۰	۳	۲۰	۱۰	۲۰	۴	۱۵	۳	۸	۵	۱	۲۰	۸	۱۱	۱۵	۱۷	۱۹	۱۶	۷	۱۰	Cost
r_{100}	r_{99}	r_{98}	r_{97}	r_{96}	r_{95}	r_{94}	r_{93}	r_{92}	r_{91}	r_{90}	r_{89}	r_{88}	r_{87}	r_{86}	r_{85}	r_{84}	r_{83}	r_{82}	r_{81}	
۱	۱	۳	۱	۲	۱	۱	۱	۳	۲	۲	۳	۱	۲	۲	۲	۱	۳	۱	۲	C_1
۱	۱	۳	۱	۲	۲	۳	۲	۲	۳	۲	۲	۲	۳	۱	۲	۲	۱	۲	۱	C_2
۳	۲	۳	۳	۲	۱	۱	۲	۳	۳	۳	۳	۲	۲	۱	۳	۲	۳	۲	۱	C_3
۳	۱	۳	۲	۱	۲	۱	۳	۳	۱	۱	۳	۱	۱	۱	۲	۲	۲	۱	۳	C_4
۳	۳	۳	۳	۳	۱	۳	۱	۱	۳	۳	۳	۱	۲	۲	۲	۲	۱	۲	۳	C_5
۳	۷	۷	۸	۷	۲	۷	۴	۱۸	۱۵	۷	۶	۱	۱۵	۱۶	۱۲	۳	۱۹	۱۶	۱۰	Cost

{(۲,۲۴),(۳,۲۶),(۳,۲۷), (۳,۲۸), (۳,۲۹), (۴,۵), (۶,۷), (۷,۳۰), (۱۰,۳۲), (۱۰,۳۳), (۱۴,۳۲), (۱۴,۳۴), (۱۴,۳۷), (۱۴,۳۸), (۱۶,۳۹), (۱۶,۴۰), (۱۷,۴۳), (۲۹,۴۹), (۲۹,۵۰), (۲۹,۵۱), (۳۰,۵۲), (۳۰,۵۳), (۳۱,۵۵), (۳۲,۵۶), (۳۲,۵۷), (۳۳,۵۸), (۳۶,۶۱), (۳۹,۶۳), (۴۰,۶۴), (۴۳,۶۵), (۴۶,۶۸), (۴۷,۷۰), (۵۵,۷۹), (۵۶,۸۰), (۵۷,۸۰), (۶۲,۸۳), (۶۲,۸۴), (۶۴,۸۷)} , $\{r_{21} \Theta r_{22}, r_{32} \Theta r_{33}, r_{46} \Theta r_{47}, r_{65} \Theta r_{66}\}$

جدول (۳): وزن مشتریان در مجموعه‌داده‌ها

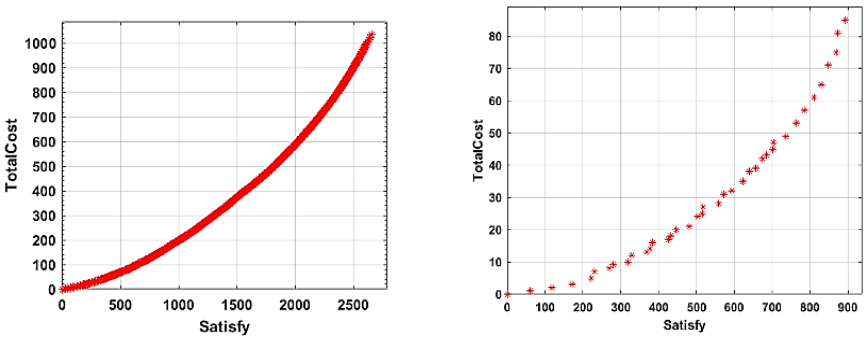
C_5	C_4	C_3	C_2	C_1	
۴	۳	۲	۴	۱	گریر
۱	۳	۳	۵	۱	ساگرادو

جدول (۴): تعداد پاسخ‌های غیرمغلوب در مجموعه‌داده‌ها

٪۱۰۰	٪۷۰	٪۵۰	٪۳۰	
۳۹	۳۳	۲۷	۱۹	گریر
۹۵۲	۸۰۰	۵۴۶	۳۴۶	ساگرادو

برای مجموعه‌داده گریر، جبهه پارتو بهینه را حساب کرده‌ایم ولی ابعاد مجموعه‌داده ساگرادو بزرگ است و محاسبه جبهه پارتو بهینه برای آن زمان‌بر و پرهزینه است. بنابراین الگوریتم‌های مختلف را با تعداد اجرای زیاد روی آن اجرا کرده‌ایم و بهترین

پاسخ‌های بدست‌آمده از الگوریتم‌های پیاده‌سازی‌شده روی مجموعه‌داده ساگرادو را بایکدیگر ادغام می‌کنیم و پاسخ‌های غیرمغلوب آن را بعنوان جبهه پارتو مرجع در نظر می‌گیریم. جبهه پارتو بهینه برای مجموعه‌داده گریر و جبهه پارتو مرجع برای مجموعه‌داده ساگرادو در شکل (۷)، نمایش داده شده‌اند. جدول (۴)، تعداد پاسخ‌های هر جبهه پارتو را نشان می‌دهد.



الف- PF_{True} برای گریر

ب- $PF_{Reference}$ برای ساگرادو

شکل (۷): جبهه پارتو بدست آمده برای مجموعه‌داده‌ها

تنظیم پارامترها در جدول (۵)، نمایش داده شده است. پارامترهای کنترلی α و σ را با اجرای مختلف الگوریتم و بررسی نتایج تنظیم کرده‌ایم. سایر پارامترها طبق مقالات مرتبط قبلی تنظیم شده‌اند.

جدول (۵): تنظیم پارامترها

	#Pop	P_{Mu}	α	σ	a	b	MaxNFE
گریر	۵۰	$(۳۵)^{-۱}$	۰.۵	۰.۸	۱	۰.۵	۱۰۰۰۰
ساگرادو	۱۰۰	۰.۰۱	۰.۵	۰.۸	۱	۰.۵	۱۰۰۰۰

۶-۳- نتایج

در این قسمت، برای بررسی تاثیر جمعیت براونی و جمعیت غیرتکراری، چهار نسخه از الگوریتم BBO را پیاده‌سازی می‌کنیم.

BBO_Base: از جمعیت براونی و جمعیت غیرتکراری استفاده نمی‌کند.

BBO_Unique: از جمعیت براونی استفاده نمی‌کند ولی از جمعیت غیرتکراری استفاده می‌کند.

BBO_Brownian_R: از جمعیت براونی استفاده می‌کند ولی از جمعیت غیرتکراری استفاده نمی‌کند.

BBO_Brownian: از جمعیت براونی و جمعیت غیرتکراری استفاده می‌کند.

الگوریتم‌ها را براساس شاخص‌های کیفیت HV، Spread و NDS با یکدیگر مقایسه می‌کنیم. چهار محدودیت بودجه نیز در نظر گرفته شده است. ۳۰٪، ۵۰٪، ۷۰٪ و ۱۰۰٪ از مجموع هزینه نیازمندی‌ها، چهار محدودیت بودجه هستند. الگوریتم‌ها را بطور مستقل ۱۰۰ بار اجرا می‌کنیم.

میانگین نتایج ۱۰۰ بار اجرای شاخص‌های کیفیت HV، Spread و NDS، روی مجموعه‌داده گریر در جدول (۶)، نمایش داده شده است. نتایج الگوریتم‌ها، نزدیک به هم هستند. برای هر محدودیت بودجه، بهترین پاسخ‌ها، بولد شده‌اند.

در محدودیت بودجه ۳۰٪، شاخص HV در الگوریتم BBO_Base که از جمعیت براونی و جمعیت غیرتکراری استفاده نمی‌کند، بهتر از سه الگوریتم دیگر است ولی در بقیه شاخص‌ها، ضعیف‌تر است. در محدودیت بودجه ۳۰٪، الگوریتم‌های BBO_Unique، BBO_Brownian و BBO_Brownian_R، نتایج یکسانی دارند؛ یعنی، در این محدودیت بودجه، رویکرد جمعیت غیرتکراری از جمعیت براونی، موثرتر است. با کاهش محدودیت بودجه (۵۰٪، ۷۰٪ و بدون محدودیت)، استفاده از جمعیت براونی و جمعیت غیرتکراری با هم، موثر عمل می‌کند و نتایج را بهبود می‌دهد. در محدودیت‌های بودجه ۵۰٪، ۷۰٪ و ۱۰۰٪، الگوریتم BBO_Brownian و BBO_Brownian_R، بهترین نتایج را دارند.

بطور کلی، در مجموعه‌داده گریر، نتایج نشان می‌دهند که:

- الگوریتم‌های BBO_Brownian و BBO_Brownian_R که از جمعیت براونی استفاده می‌کنند، بهترین نتایج را دارند.
- الگوریتم BBO_Brownian از BBO_Brownian_R، عملکرد بهتری دارد. در نتیجه استفاده از جمعیت براونی و جمعیت غیرتکراری با هم، موثرتر عمل می‌کند.

جدول (۶): تاثیر جمعیت براونی و جمعیت غیرتکراری در مجموعه‌داده گریر

درصد بودجه	الگوریتم	HV	Spread	NDS
۳۰	BBO_Base	**۱۱.۲۸۴**	۰.۲۵۳	۱۸.۸۱
	BBO_Unique	۱۱.۲۷۲	**۰.۲۴۴**	**۱۹**
	BBO_Brownian_R	۱۱.۲۷۲	**۰.۲۴۴**	**۱۹**
	BBO_Brownian	۱۱.۲۷۲	**۰.۲۴۴**	**۱۹**
۵۰	BBO_Base	۲۴.۹۹۲	۰.۳۰۷	۲۵.۷۱
	BBO_Unique	۲۵.۰۱۸	۰.۳۰۱	۲۵.۹
	BBO_Brownian_R	۲۵.۰۱۸	**۰.۲۹۷**	**۲۶**
	BBO_Brownian	**۲۵.۰۲**	**۰.۲۹۷**	**۲۶**
۷۰	BBO_Base	۳۹.۶۴۱	۰.۳۳۶	۳۱.۱۷
	BBO_Unique	۳۹.۷۱۳	۰.۳۲۱	۳۱.۸
	BBO_Brownian_R	**۳۹.۷۲۳**	۰.۳۲۱	**۳۲**
	BBO_Brownian	**۳۹.۷۲۳**	**۰.۳۱۹**	**۳۲**
۱۰۰	BBO_Base	۷۰.۸۶۱	۰.۳۵۵	۳۶.۰۶
	BBO_Unique	۷۱.۲۶۱	۰.۳۴۱	۳۷.۹
	BBO_Brownian_R	۷۱.۲۶۲	**۰.۳۳۸**	**۳۸**
	BBO_Brownian	**۷۱.۲۶۳**	**۰.۳۳۸**	**۳۸**

- الگوریتم BBO_Unique از BBO_Base عملکرد بهتری دارد که نشان می‌دهد، استفاده از جمعیت غیرتکراری، الگوریتم را بهبود می‌دهد.

در جدول (۷)، میانگین نتایج ۱۰۰ بار اجرای شاخص‌های کیفیت HV، Spread و NDS، روی مجموعه‌داده ساگرادو، نمایش داده شده است. برای هر محدودیت بودجه، بهترین پاسخ‌ها، بولد شده است.

در محدودیت بودجه ۳۰٪، الگوریتم BBO_Unique، بیشترین HV را دارد. در محدودیت‌های بودجه ۵۰٪، ۷۰٪ و ۱۰۰٪، الگوریتم BBO_Brownian، بهترین HV را به‌دست آورده‌است. در تمام محدودیت‌های بودجه، الگوریتم BBO_Brownian، بیشترین NDS و الگوریتم BBO_Brownian_R، بهترین نتیجه Spread را دارند. الگوریتم BBO_Base، بدترین نتایج HV و NDS را در تمام شاخص‌ها دارد.

بطور کلی، در مجموعه‌داده ساگرادو، نتایج نشان می‌دهند که:

- الگوریتم‌های BBO_Brownian و BBO_Brownian_R که از جمعیت براونی استفاده می‌کنند، بهترین نتایج را دارند.
- در شاخص Spread، الگوریتم‌های BBO_Brownian_R، BBO_Brownian، BBO_Base و BBO_Unique، به‌ترتیب؛ رتبه اول تا آخر را دارند. الگوریتم BBO_Base از BBO_Unique، الگوریتم BBO_Brownian_R از BBO_Brownian، الگوریتم BBO_Brownian از BBO_Base، نتایج Spread بهتری دارند. به‌عبارت دیگر؛ استفاده از جمعیت غیرتکراری، Spread را بدتر و جمعیت براونی، Spread را بهتر می‌کند.
- در شاخص NDS، الگوریتم‌های BBO_Brownian، BBO_Unique، BBO_Brownian_R و BBO_Base به‌ترتیب؛ رتبه اول

تا آخر را دارند. به‌عبارت دیگر؛ استفاده از جمعیت غیرتکراری، تاثیر بیشتری از جمعیت براونی، در NDS دارد.

جدول (۷): تاثیر جمعیت براونی و جمعیت غیرتکراری در مجموعه‌داده ساگرادو

درصد بودجه	الگوریتم	HV	Spread	NDS
۳۰	BBO_Base	۸.۶۱	۰.۴۹۵	۱۱۷.۶۳
	BBO_Unique	**۸.۹۷۲**	۰.۴۹۶	۱۶۵.۹
	BBO_Brownian_R	۸.۸۰	**۰.۴۷۲**	۱۶۱
	BBO_Brownian	۸.۸۱	۰.۴۸۸	**۱۶۶.۲**
۵۰	BBO_Base	۱۹.۸۵	۰.۵۵۸	۱۴۳.۶۹
	BBO_Unique	۲۰.۵۰۲	۰.۵۳۰	۱۷۸.۲
	BBO_Brownian_R	۲۰.۳۳	**۰.۴۶۷**	۱۶۸
	BBO_Brownian	**۲۰.۵۵۱**	۰.۴۸۵	**۱۹۱.۲**
۷۰	BBO_Base	۳۴.۱۳۸	۰.۵۵۵	۱۷۱.۶۴
	BBO_Unique	۳۴.۸۴۸	۰.۵۷۷	۲۱۲.۸
	BBO_Brownian_R	۳۵.۰۴۴	**۰.۴۸۳**	۱۹۸
	BBO_Brownian	**۳۵.۶۷۹**	۰.۴۸۹	**۲۱۵**
۱۰۰	BBO_Base	۵۹.۰۸۵	۰.۵۰۲	۱۵۹.۹۴
	BBO_Unique	۶۱.۵۱۴	۰.۵۷۲	۲۲۰
	BBO_Brownian_R	۶۲.۲۹۴	**۰.۴۷۴**	۲۱۲
	BBO_Brownian	**۶۳.۶۸۹**	۰.۴۸۲	**۲۵۴.۹**

- در شاخص HV، استفاده از دو رویکرد جمعیت غیرتکراری و جمعیت براونی با هم، نتایج را بهبود می‌دهد. در محدودیت‌های بیشتر بودجه، جمعیت غیرتکراری و در محدودیت‌های کمتر بودجه، جمعیت براونی موثرترند.

در کل، نتایج نشان داد که استفاده از جمعیت براونی و جمعیت تکراری، نتایج را بهبود می‌بخشند. در شاخص Spread، جمعیت براونی و در شاخص NDS، جمعیت غیرتکراری موثرتر هستند.

۶-۴- مقایسه با نتایج دیگران

HV، Spread و NDS مستقل از جبهه پارتو بهینه محاسبه می‌شوند و در مقالات بیشتر استفاده می‌گردند. مجموعه‌داده‌های گریر و ساگرادو نیز در بیشتر پژوهش‌های مشابه استفاده شده‌اند. بهترین نتایج بدست آمده در تحقیق دیگران را استخراج کرده و با نتایج BBO_Brownian مقایسه می‌کنیم. این نتایج در شرایط مشابه، با حداکثر تعداد فراخوانی تابع هدف ۱۰۰۰۰ بار و ۱۰۰ بار اجرای مستقل، روی مجموعه‌داده‌های گریر و ساگرادو بدست آمده‌اند.

با توجه به اینکه نتایج بصورت میانگین است و جزییات پاسخ‌های دیگر تحقیقات را دسترسی نداریم، شاخص‌های HV، Spread و NDS را برای جبهه پارتو بهینه بدست می‌آوریم و میزان نزدیکی HV، Spread و NDS جبهه پارتو بدست‌آمده با الگوریتم‌ها را با مقادیر متناظرشان در جبهه پارتو بهینه، مبنای مقایسه قرار می‌دهیم. برای هر مجموعه‌داده، چهار محدودیت بودجه نیز در نظر گرفته شده است. ۳۰٪، ۵۰٪، ۷۰٪ و ۱۰۰٪ از مجموع هزینه نیازمندی‌ها، چهار محدودیت بودجه هستند.

در جدول‌های (۸) و (۹)، میانگین نتایج HV، Spread و NDS برای مجموعه‌داده‌های گریر و ساگرادو نمایش داده شده‌اند. PF_{True} جبهه پارتو بهینه برای گریر و $PF_{Reference}$ جبهه پارتو مرجع برای ساگرادو هستند. اولین و دومین نزدیکترین نتایج به جبهه پارتو بهینه را به ترتیب بولد و قرمز کرده‌ایم. در مجموعه‌داده گریر، تقریبا در همه شاخص‌های کیفیت؛ الگوریتم پیشنهادی، نزدیکترین یا دومین نزدیکترین نتایج را به PF_{True} دارد. الگوریتم‌های ارائه شده در [۳۱] و [۳۲] نیز همانند

الگوریتم چندهدفی پیشنهادی جغرافیای زیستی؛ از جمعیت غیرتکراری و ادغام پاسخ‌ها و مخزن پاسخ خارجی استفاده می‌کنند. بهترین نتایج بجز دو مورد، مربوط به همین الگوریتم‌هاست.

جدول (۸): مقایسه نتایج HV، Spread و NDS با نتایج دیگران روی مجموعه‌داده گریر

	محدودیت بودجه ۳۰ درصد			محدودیت بودجه ۵۰ درصد			محدودیت بودجه ۷۰ درصد			بدون محدودیت بودجه		
$PF_{Refrence}$	HV	Spread	NDS	HV	Spread	NDS	HV	Spread	NDS	HV	Spread	NDS
	۱۱.۲۷۲	**۰.۲۴۴**	**۱۹**	**۲۴.۹۳**	**۰.۲۹۳**	**۲۷**	**۳۹.۶۴**	**۰.۳۲۲**	**۳۳**	**۷۱.۱۷۲**	**۰.۳۴۴**	**۳۹**
Proposed BBO	**۱۱.۲۷۲**	**۰.۲۴۴**	**۱۹**	۲۵.۰۲	**۰.۲۹۷**	۲۶	۳۹.۷۲۳	۰.۳۱۹	۳۲.۶۵	**۷۱.۲۶**	۰.۳۳۸	**۳۸**
MOGWO [۳۲]	۱۱.۲۷	۰.۴۴۳	۱۸.۹۷	۲۵.۰۲	۰.۳۹۸	۲۵.۹۶	**۳۹.۷۱**	۰.۳۷۲	۳۱.۸۹	۷۱.۵۲	۰.۳۳۸	۳۷.۹۰
MOWOA [۳۲]	۱۱.۲۷	۰.۴۴۱	**۱۹**	۲۵.۰۲	۰.۳۹۸	۲۶	۳۹.۷۳	۰.۳۷۰	۳۲.۰	۷۱.۵۷	۰.۳۳۸	**۳۸**
NSGA2 [۳۱]	۱۱.۲۷	۰.۲۴۰	**۱۹**	۲۵.۰۲	۰.۳۰	۲۶.۱۲	۳۹.۷۷	**۰.۳۲**	۳۱.۸	**۷۱.۲۶**	**۰.۳۴**	۳۷.۹۸
MODSA[۳۱]	۱۱.۲۸	۰.۲۵	۱۸.۷۹	**۲۴.۹۹**	۰.۳۱	۲۵.۷۷	۳۹.۸۲	۰.۳۳	۳۱.۲۹	**۷۱.۲۶**	۰.۳۲	۳۷.۸۶
MOTLBO [۱۴]	۴۲.۹۸۱	۰.۵۲	۱۵	۵۶.۲۱۹	۰.۴۶	۲۴.۱۶	۶۲.۸۳۷	۰.۴۰	**۳۲.۹۵**	۶۶.۵۶۲	۰.۳۸	۴۱.۸۵
MOTLABC [۲۶]	۵۴.۸۹۲	۰.۴۸	۱۷.۸	۶۲.۵۶۷	۰.۴۴	۲۸.۷۶	۶۸.۷۶۹	۰.۳۷	۳۶.۲۱	۷۴.۸۹۷	۰.۳۵	۴۸.۹۸
GRASP[۱۳]	۷.۷۱	۰.۶۴	۱۱.۳۷	–	–	–	۳۲.۲۵	۰.۶۹	۲۰.۲۶	–	–	–
NSGA2 [۱۳]	۹.۰۲	۰.۷۶	۹.۶۹	–	–	–	۳۲.۱۶	۰.۸۰	۱۱.۷	–	–	–
ACS [۱۳]	۱۰.۲۸	۰.۵۲	۱۳.۶۶	–	–	–	۳۸.۴۷	۰.۴۸	۲۰.۵۷	–	–	–
DEPT[۲۵]	۳۸.۸۸۱	۰.۵۲	۱۵	۵۰.۱۱۲	۰.۴۸	۱۹.۷۶	۵۸.۹۵۴	۰.۴۲	۲۶.۲۲	۶۰.۷۷۶	۰.۴۰	۳۰.۵۱
SPEA2 [۱۴]	۳۲.۸۹	–	–	۴۷.۱۱۲	–	–	۵۳.۴۶۴	–	–	۵۸.۵۵۹	–	–
MOABC [۴۱]	۴۱.۸۸۰	۰.۵۲	۱۵	۵۴.۷۱۵	۰.۴۸	۲۳.۶۶	۶۰.۸۵۵	۰.۴۳	۳۲.۳۵	۶۳.۷۸	۰.۳۹	۴۰.۵۵
PMOABC2[۲۸]	۴۵.۰۴	۰.۴۸	۱۷.۱۶	۵۶.۴۳	۰.۴۶	**۲۶.۳۱**	۶۱.۹۵	۰.۴۱	۳۶.۳۱	۶۵.۱۸	۰.۳۷	۴۳.۱۱
PMOABC4[۲۸]	۴۶.۶۵	۰.۴۶	۲۰.۲۶	۵۷.۲۹	۰.۴۵	۳۰.۶۳	۶۲.۶۷	۰.۴۰	۳۷.۵۴	۶۶.۰۵	۰.۳۶	۴۴.۸۹
HABC-DE [۲۹]	۴۵.۵	۰.۵۰	۲۲.۵	۵۶.۸۱	۰.۴۵	۳۰	۴۵.۵	۰.۴۱	۳۷.۷۵	۶۷.۵	۰.۳۸	۴۵.۱

برای محدودیت بودجه ٪۳۰، در همه شاخص‌ها بهترین نتیجه را دارد. NSGA2[۳۱] و MOWOA[۳۲] و MOGWO [۳۲]، به‌ترتیب در رتبه‌های بعد قراردارند. برای محدودیت بودجه ٪۵۰، MODSA[۳۱] بهترین ابرحجم و الگوریتم چندهدفی کلونی زنبور مصنوعی موازی (PMOABC 2[۳۱])، نزدیکترین تعداد پاسخ‌های غیرمغلوب را به PF_{True} دارد. الگوریتم پیشنهادی بهترین گستردگی و دومین بهترین ابرحجم را بدست آورده‌است.

برای محدودیت بودجه ٪۷۰، الگوریتم پیشنهادی، دومین بهترین نتایج را دارد. MOWOA[۳۲] بهترین ابرحجم، NSGA2[۳۱]، بهترین گستردگی و MOTLBO[۱۴]، بهترین تعداد پاسخ را کسب کردند. برای حالت بدون محدودیت بودجه، الگوریتم پیشنهادی و MODSA[۳۱] و NSGA2[۳۱]، بهترین ابرحجم را بدست آورده‌اند. NSGA2[۳۱]، بهترین گستردگی و الگوریتم پیشنهادی، بهترین NDS را بدست آورده‌اند.

در مجموعه‌داده ساگرادو، تقریبا در همه شاخص‌ها؛ الگوریتم بهینه‌سازی جغرافیای زیستی پیشنهادی، نزدیکترین یا دومین نزدیکترین نتایج را به جبهه پارتو مرجع $PF_{Refrence}$ دارد. بیشترین تعداد پاسخ‌های غیرمغلوب را نیز الگوریتم پیشنهادی، بدست آورده است.

در شاخص HV، الگوریتم پیشنهادی، در تمام محدودیت‌های بودجه، بهترین نتیجه را دارد. در محدودیت بودجه ٪۳۰، MOWOA[۳۲]، در محدودیت بودجه ٪۵۰ و ٪۷۰، MOGWO[۳۲] و در حالت بدون محدودیت بودجه، MOTLBO[۱۴]، دومین بهترین HV را دارند.

در شاخص Spread، در محدودیت بودجه ٪۳۰، MOTLBO[۱۴] و MOABC[۴۱]، بهترین نتیجه را دارند. الگوریتم پیشنهادی، دومین بهترین نتیجه را بدست آورده‌است. در بقیه موارد، الگوریتم پیشنهادی و الگوریتم‌های ارائه شده در [۳۱] و [۳۲]، نتایج بهتری کسب کرده‌اند.

جدول (۹): مقایسه نتایج HV، Spread و NDS با نتایج دیگران روی مجموعه‌داده ساگرادو

$PF_{Refrence}$	محدودیت بودجه ۳۰ درصد			محدودیت بودجه ۵۰ درصد			محدودیت بودجه ۷۰ درصد			بدون محدودیت بودجه		
	HV	Spread	NDS	HV	Spread	NDS	HV	Spread	NDS	HV	Spread	NDS
	۹.۱۸	۰.۴۶۷	۳۴۶	۲۱.۲۶	۰.۴۶۸	۵۳۶	۳۶.۷۶	۰.۵۶۳	۸۰۰	۶۳.۷۳	۰.۵۸۹	۹۵۲
Proposed BBO	۸.۸۱	۰.۴۸۸	۱۶۶.۲	۲۰.۵۵	۰.۴۸۵	۱۹۱.۲	۳۵.۶۸	۰.۴۸۹	۲۱۵	۶۳.۶۹	۰.۴۸۲	۲۵۴.۹
MOGWO [۳۲]	۸.۷۱	۰.۶۲۵	۱۵۸.۸۲	۲۰.۵۳	۰.۵۶۲	**۱۹۶.۱۰**	۳۵.۶۲	۰.۵۲	۲۱۰.۴۶	۶۲.۹۲	۰.۴۸۱	۲۰۸.۳۷
MOWOA [۳۲]	۸.۷۵	۰.۶۶۳	۱۲۴.۹۴	۲۰.۰۳	۰.۵۹	۱۴۰.۲۸	۳۵.۶۰	**۰.۵۳۸**	۱۵۰.۸۱	۶۳.۱۱	۰.۴۷۴	۲۱۸.۳۷
NSGA2 [۳۱]	**۸.۸۱**	۰.۵۷	۱۶۴.۴۶	۱۹.۸۱	۰.۵۸	۱۸۷.۸۵	۳۴.۶۹	۰.۵۹	**۲۲۱.۴۱**	۶۳.۰۷	**۰.۵۵**	۲۴۲.۴۵
MODSA[۳۱]	۸.۴۹	۰.۵۲	۱۴۶.۳۸	۱۹.۵۳	۰.۵۱	۱۶۳.۰۳	۳۴.۱۴	۰.۴۹	۱۸۶.۹۱	۶۲.۸۴	۰.۴۸	۲۲۳.۳۰
MOTLBO [۱۴]	۴۳.۱۸۲	**۰.۴۵**	۱۲۹	۵۳.۱۲۲	۰.۴۱	۱۳۶.۱۳	۵۹.۹۹۲	۰.۳۶	۱۴۴.۸۵	۶۴.۱۲۶	۰.۳۴	۱۵۲.۵۵
MOTLABC[۲۶]	۵۵.۵۴۱	۰.۴۰	۱۴۴.۲۵	۶۱.۲۳۴	۰.۳۸	۱۵۴.۷۸	۶۷.۸۷۹	۰.۳۴	۱۶۰.۱۲	۷۲.۶۵۴	۰.۳۱	۱۶۹.۰۹
GRASP[۱۳]	۴.۰۹	۰.۶۰	۵۷.۹۹	۱۵.۴۶	۰.۷۴	۷۵.۸۱	۲۷.۹۵	۰.۷۰	۱۲۰.۱۴	–	–	–
NSGA2 [۱۳]	۷.۹۱	۰.۸۰	۵۴.۳۴	۱۸.۰۱	۰.۸۱	۶۵.۵۴	۳۱.۷۲	۰.۷۷	۸۳.۳۲	–	–	–
ACS [۱۳]	۸.۵۲	۰.۶۸	۴۷.۱۲	۱۹.۱۶	۰.۶۶	۵۷.۶۸	۳۲.۷۸	۰.۶۱	۷۰.۹۸	–	–	–
DEPT[۲۵]	۳۶.۵۰۸	۰.۵۶	۱۱۰.۱۱	۴۶.۶۵	۰.۵۱	۱۲۳.۶۴	۵۲.۷۵۳	۰.۴۷	۱۳۹.۷۳	۵۸.۰۲۶	۰.۴۴	۱۴۴.۵۰
SPEA2 [۱۴]	۳۳.۱۱۳	–	–	۴۳.۹۵	–	–	۵۰.۵۳۸	–	–	۵۵.۹۶۶	–	–
MOABC [۴۱]	۴۱.۲۳۲	**۰.۴۵**	۱۲۵.۳۷	۵۱.۲۱۲	۰.۴۲	۱۳۵.۹۳	۵۸.۲۱۲	۰.۳۸	۱۳۹.۳۱	۶۱.۷۰۲	۰.۳۵	۱۴۷.۵۱
PMOABC2[۲۸]	۴۲.۶۳۴	۰.۴۴	۱۳۱.۲۱	۵۸.۶۷۲	۰.۳۸	۱۴۱.۰۸	۶۰.۹۶۵	۰.۳۷	۱۴۵.۱۷	۶۲.۳۱۰	۰.۳۴	۱۵۸.۱۲
PMOABC4[۲۸]	۴۲.۸۷۷	۰.۴۲	۱۳۵.۳۳	۶۰.۳۱۲	۰.۳۸	۱۴۳.۷۳	۶۱.۸۷۲	۰.۳۵	۱۵۲.۲۲	۶۲.۷۹۸	۰.۳۳	۱۶۱.۳۹
HABC-DE [۲۹]	۴۳.۲۲	۰.۴۰	۱۳۸.۱۴	۶۰.۶۱۴	۰.۳۷	۱۴۵.۲۲	۶۱.۷۱	۰.۳۶	۱۵۹.۰۸	۶۴.۳۲	۰.۳۴	۱۶۳.۴۸

در شاخص NDS، در محدودیت بودجه ۳۰٪ و ۱۰۰٪، الگوریتم پیشنهادی، بهترین نتیجه را به‌دست آورده‌است. در محدودیت بودجه ۵۰٪ و ۷۰٪، به‌ترتیب؛ MOGWO[۳۲] و NSGA2[۳۱] بهترین NDS را دارند. الگوریتم پیشنهادی در رتبه دوم قرار دارد. با توجه به نتایج، مشخص می‌گردد که بطورکلی؛ در هر دو مجموعه‌داده – مجموعه‌داده کوچک گریر و مجموعه‌داده نسبتا بزرگ ساگرادو - الگوریتم پیشنهادی و NSGA2[۳۱]، به‌ترتیب؛ بهترین نتایج را دارند.

۷-نتیجه‌گیری

در این مقاله، نسخه چندهدفی الگوریتم بهینه‌سازی جغرافیای زیستی را برای مسئله انتشار نسخه بعد استفاده کردیم. برای بهبود جستجو، جمعیت براونی را براساس حرکت تصادفی شبه براونی ایجاد کردیم و با الگوریتم بهینه‌سازی جغرافیای زیستی ترکیب کردیم. در هر مرحله الگوریتم از جمعیت با اعضای غیرتکراری و ادغام پاسخ‌ها استفاده شد. احتمال مهاجرت به هریک از زیستگاه‌ها را نیز با رویکردی جدید و براساس نسبت توابع هدف و چرخه رولت ایجاد کردیم.

از دو مجموعه‌داده و شاخص‌های کیفیت ابرحجم، گستردگی و تعداد پاسخ‌های غیرمغلوب، برای ارزیابی، استفاده کردیم. چهار محدودیت بودجه ۳۰٪، ۵۰٪، ۷۰٪ و ۱۰۰٪ از مجموع هزینه نیازمندی‌ها را روی مجموعه‌داده‌ها در نظر می‌گیریم.

براساس استفاده یا عدم استفاده از جمعیت براونی و جمعیت غیرتکراری، چهار نسخه از الگوریتم جغرافیای زیستی را پیاده‌سازی کردیم. الگوریتم جغرافیای زیستی پیشنهادی که از جمعیت براونی و جمعیت غیرتکراری، استفاده می‌کند؛ نتایج بهتری از سه الگوریتم دیگر، بدست آورد. در شاخص Spread، جمعیت براونی و در شاخص NDS، جمعیت غیرتکراری موثرتر هستند.

در پایان، نتایج الگوریتم پیشنهادی را با نتایج الگوریتم‌های ارائه‌شده در کارهای مرتبط، مقایسه کردیم. میزان نزدیکی HV، Spread و NDS جبهه پارتو بدست‌آمده با الگوریتم‌ها را با مقادیر متناظرشان در جبهه پارتو بهینه، مبنای مقایسه قرار دادیم. در تقریبا تمام شاخص‌ها، الگوریتم پیشنهادی، رتبه اول یا دوم دارد.

جمعیت براونی معرفی‌شده در این مقاله را می‌توان با دیگر الگوریتم‌های تکاملی، ترکیب کرد و برای حل مسئله انتشار نسخه بعد بکار برد. برای افزایش سرعت می‌توان با شرایط موازی‌سازی به پاسخ‌های مناسب‌تر رسید. همچنین از توابع و عملگرهای این چارچوب، برای حل مسائل دیگر مبتنی بر جستجو، می‌توان بهره برد. به ویژه زمانی‌که تعداد بیشتر پاسخ‌های غیرمغلوب

مهم باشد.

با توجه به افزایش همکاری و ارتباطات شرکت‌ها و سازندگان نرم‌افزار با یکدیگر، مسئله NRP را می‌توان بصورت چند شرکتی، باز تعریف کرد، بطوری‌که چند شرکت نرم‌افزاری با همکاری یکدیگر بخواهند، نیازمندی‌های مشتریانشان را با بیشترین سود و کمترین هزینه پیاده‌سازی کنند.

References

مراجع

[1] G. Brau, J. Hugues, N. Navet, "Towards the systematic analysis of non-functional properties in Model-based engineering for real-time embedded systems", *Science of Computer Programming*, vol. 156, pp. 1–20, Dec. 2017, doi: https://doi.org/10.1016/j.scico.2017.12.007.

[2] M. Harman, S. A. Mansouri, Y. Zhang, "Search-based software engineering", *ACM Computing Surveys*, vol. 45, no. 1, pp. 1–61, Nov. 2012, doi: https://doi.org/10.1145/2379776.2379787.

[3] S. Katoch, S. S. Chauhan, V. Kumar, "A review on genetic algorithm: past, present, and future", *Multimedia Tools and Applications*, vol. 80, no. 5, Oct. 2020, doi: https://doi.org/10.1007/s11042-020-10139-6.

[4] T. Chen and M. Li, "The weights can be harmful: pareto search versus weighted search in multi-objective search-based software engineering", *ACM Transactions on Software Engineering and Methodology*, Apr. 2022, doi: https://doi.org/10.1145/3514233.

[5] A. J. Bagnall, V. J. Rayward-Smith, I. M. Whittley, "The next release problem", *Information and Software Technology*, vol. 43, no. 14, pp. 883–890, Dec. 2001, doi: https://doi.org/10.1016/s0950-5849(01)00194-x.

[6] B. Liu, G. Zhou, Y. Zhou, Q. Luo, Y. Wei, "Quantum-inspired multi-objective african vultures optimization algorithm with hierarchical structure for software requirement", *Cluster Computing*, vol. 27, no. 8, pp. 11317–11345, May 2024, doi: https://doi.org/10.1007/s10586-024-04503-6.

[7] José del Sagrado, A. Sierra, I. María, "An estimation of distribution algorithm based on interactions between requirements to solve the bi-objective Next Release Problem", *Journal of Systems and Software*, vol. 199, p. 111632, May 2023, doi: https://doi.org/10.1016/j.jss.2023.111632.

[8] Seyedali Mirjalili, Jin Song Dong, A. Lewis, "Nature-inspired optimizers", Springer, 2019.

[9] I. Rahimi, A. H. Gandomi, M. R. Nikoo, F. Chen, "A comparative study on evolutionary multi-objective algorithms for next release problem", *Applied Soft Computing*, vol. 144, p. 110472, May 2023, doi: https://doi.org/10.1016/j.asoc.2023.110472.

[10] C. A. Coello Coello, S. González Brambila, J. Figueroa Gamboa, M. G. Castillo Tapia, R. Hernández Gómez, "Evolutionary multiobjective optimization: open research areas and some challenges lying ahead", *Complex & Intelligent Systems*, vol. 6, no. 2, pp. 221–236, Jun. 2019, doi: https://doi.org/10.1007/s40747-019-0113-4.

[11] J. Dong, W. Gong, F. Ming, L. Wang, "A two-stage evolutionary algorithm based on three indicators for constrained multi-objective optimization", *Expert Systems with Applications*, vol. 195, p. 116499, Jun. 2022, doi: https://doi.org/10.1016/j.eswa.2022.116499.

[12] Y. Zhang, M. Harman, S. A. Mansouri, "The multi-objective next release problem", *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pp. 1129-1137, London, England, 2007, doi: https://doi.org/10.1145/1276958.1277179.

[13] J. del Sagrado, I. M. del Águila, F. J. Orellana, "Multi-objective ant colony optimization for requirements selection", *Empir. Softw. Eng.*, vol. 20, no. 3, pp. 577–610, 2015.

[14] J. M. Chaves-González, M. A. Pérez-Toledano, A. Navasa, "Teaching learning based optimization with pareto tournament for the multiobjective software requirements selection", *Engineering Applications of Artificial Intelligence*, vol. 43, pp. 89–101, Aug. 2015, doi: https://doi.org/10.1016/j.engappai.2015.04.002.

[15] S. P. Adam, S.-A. N. Alexandropoulos, P. M. Pardalos, M. N. Vrahatis, "No free lunch theorem: a review", in *Springer Optimization and Its Applications*, Cham: Springer International Publishing, 2019, pp. 57–82. doi: http://dx.doi.org/10.1007/978-3-030-12767-1_5.

[16] X. Cai, M. Hu, D. Gong, Y. N. Guo, Y. Zhang, Z. Fan, Y. Huang, "A decomposition-based coevolutionary multiobjective local search for combinatorial multiobjective optimization", *Swarm and Evolutionary Computation*, vol. 49, pp. 178–193, Sep. 2019, doi: 10.1016/j.swevo.2019.05.007.

[17] W. H. Dukhan, M. H. Mohamed, A. A. Amer, E. A. Zanaty, O. Reyad, "Software requirement selection using a combined multi-objective optimisation technique", *IET Software*, vol. 16, no. 6, pp. 558–575, Aug. 2022, doi: 10.1049/sfw2.12070.

[18] L. Li, "Exact analysis for next release problem", *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pp. 438-443, Beijing, China, 2016, doi: 10.1109/RE.2016.7.

[19] P. Singh, M. S. Bhamrah and J. Kaur, "Jaya algorithm using weighted sum approach for the multi-objective next release problem", *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, pp. 273-277, Jalandhar, India, 2018, doi: 10.1109/ICSCCC.2018.8703315.

[20] X. Cai and O. Wei, "A hybrid of decomposition and domination based evolutionary algorithm for multi-objective software next release problem", *2013 10th IEEE International Conference on Control and Automation (ICCA)*, pp. 412-417, Hangzhou, China, Jun. 2013, doi: 10.1109/ICCA.2013.6565143.
[21] M. Á. Domínguez-Ríos, F. Chicano, E. Alba, I. del Águila, J. Del Sagrado, "Efficient anytime algorithms to solve the bi-objective next release problem", *Journal of Systems and Software*, vol. 156, pp. 217–231, Oct. 2019, doi: https://doi.org/10.1016/j.jss.2019.06.097.
[22] D. Greer and G. Ruhe, "Software release planning: an evolutionary and iterative approach", *Information and Software Technology*, vol. 46, no. 4, pp. 243–253, Mar. 2004, doi: https://doi.org/10.1016/j.infsof.2003.07.002.
[23] J. J. Durillo, Y. Zhang, E. Alba, M. Harman, A. J. Nebro, "A study of the bi-objective next release problem", *Empirical Software Engineering*, vol. 16, no. 1, pp. 29–60, Dec. 2010, doi: 10.1007/s10664-010-9147-3.
[24] J. del Sagrado, I. M. del Águila and F. J. Orellana, "Ant colony optimization for the next release problem: a comparative study", *2nd International Symposium on Search Based Software Engineering*, pp. 67-76, Benevento, Italy, 2010, doi: 10.1109/SSBSE.2010.18.
[25] J. M. Chaves-González and M. A. Pérez-Toledano, "Differential evolution with pareto tournament for the multi-objective next release problem", *Applied Mathematics and Computation*, vol. 252, pp. 1–13, Dec. 2014, doi: https://doi.org/10.1016/j.amc.2014.11.093.
[26] N. Ranjith and A. Marimuthu, "A multi objective teacher-learning-artificial bee colony (MOTLABC) optimization for software requirements selection", *Indian Journal of Science and Technology*, vol. 9, no. 34, Sep. 2016, doi: https://doi.org/10.17485/ijst/2016/v9i34/95638.
[27] A. Hamdy and A. A. Mohamed, "Greedy binary particle swarm optimization for multi-objective constrained next release problem", *International Journal of Machine Learning and Computing*, vol. 9, no. 5, pp. 561–568, Oct. 2019, doi: https://doi.org/10.18178/ijmlc.2019.9.5.840.
[28] H. Alrezaamiri, A. Ebrahimnejad, H. Motameni, "Parallel multi-objective artificial bee colony algorithm for software requirement optimization", *Requirements Engineering*, Jan. 2020, doi: https://doi.org/10.1007/s00766-020-00328-y.
[29] Y.T. Qi, F. Liu, J.L. Liu, Y. Ren, L.C. Jiao, "Hybrid Immune algorithm with EDA for multi-objective optimization", *Journal of Software*, vol. 24, no. 10, pp. 2251–2266, Jan. 2014, doi: https://doi.org/10.3724/sp.j.1001.2013.04341.
[30] I. María and José del Sagrado, "Salience-based stakeholder selection to maintain stakeholder coverage in solving the next release problem", *Information & Software Technology*, vol. 160, p. 107231, Aug. 2023, doi: https://doi.org/10.1016/j.infsof.2023.107231.
[31] M. Ghasemi, Karamollah Bagherifard, H. Parvin, Samad Nejatian, "Novel multi objective evolutionary framework for solving next release problem", *Journal of Intelligent and Fuzzy Systems*, vol. 44, no. 2, pp. 3315–3339, Jan. 2023, doi: https://doi.org/10.3233/jifs-200223.
[32] M. Ghasemi, K. Bagherifard, H. Parvin, S. Nejatian, K.-H. Pho, "Multi-objective whale optimization algorithm and multi-objective grey wolf optimizer for solving next release problem with developing fairness and uncertainty quality indicators", *Applied Intelligence*, vol. 51, no. 8, pp. 5358–5387, Jan. 2021, doi: https://doi.org/10.1007/s10489-020-02018-2.
[33] K. V. Divya and R. J. Anandhi, "Fuzzy branch and bound optimization technique for solving multi-objective next-release problem", *SN Computer Science*, vol. 5, no. 7, Sep. 2024, doi: 10.1007/s42979-024-03231-3.
[34] S. Verma, M. Pant, V. Snasel, "A comprehensive review on nsga-ii for multi-objective combinatorial optimization problems", *IEEE Access*, vol. 9, pp. 57757–57791, 2021, doi: 10.1109/access.2021.3070634.
[35] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II", *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[36] D. Simon, "Biogeography-based optimization", *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, Dec. 2008, doi: https://doi.org/10.1109/TEVC.2008.919004.
[37] X. Li and M. Yin, "Multiobjective binary biogeography based optimization for feature selection using gene expression data", *IEEE Transactions on NanoBioscience*, vol. 12, no. 4, pp. 343–353, Dec. 2013, doi: https://doi.org/10.1109/tnb.2013.2294716.
[38] J. Li, X. Guo, Y. Yang, Q. Zhang, "A hybrid algorithm for multi-objective optimization—combining a biogeography-based optimization and symbiotic organisms search", *Symmetry*, vol. 15, no. 8, p. 1481, Jul. 2023, doi: 10.3390/sym15081481.
[39] P. Civicioglu, "Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm", *Computers & amp; Geosciences*, vol. 46, pp. 229–247, Sep. 2012, doi: 10.1016/j.cageo.2011.12.011.
[40] C. Coello, G. B. Lamont, D. A. van Veldhuizen, "Evolutionary algorithms for solving multi-objective problems", 5th Edition. Springer Science & Business Media, 2007.

[41] J. M. Chaves-González, M. A. Pérez-Toledano, A. Navasa, "Software requirement optimization using a multiobjective swarm intelligence evolutionary algorithm", *Knowledge-Based Systems*, vol. 83, pp. 105–115, Jul. 2015, doi: 10.1016/j.knosys.2015.03.012.

زیرنویس‌ها:

1. Search Based Software Engineering
2. Next Release Problem
3. Pareto optimal solutions’set
4. nonDominated solutions
5. No-Free Lunch
6. Biogeography Based Optimization
7. Brownian-like random-walk
8. Hyper Volume
9. nonDominated sorting
10. Crowding distance
11. Estimation Of Distribution Algorithm
12. Fuzzy Branch and Bound Optimization
13. Habitat Suitability Index
14. Suitability Index Variable
15. Number of Fitness Evaluations