Journal of Soft Computing and Information Technology (JSCIT)
Babol Noshirvani University of Technology, Babol, Iran
Journal Homepage: jscit.nit.ac.ir
Volume 8, Number 3, Fall 2019, pp. 10-19
Received: 08/24/2017, Revised: 06/03/2019, Accepted: 06/29/2019

Gravitational Search Algorithm with Nearest-Better Neighborhood for Multimodal Optimization Problems

Mohammad Bagher Dowlatshahi[1], Vali Derhami[2*] and Hossein Nezamabadi-pour[3]
1- Department of Computer Engineering, Faculty of Engineering, Lorestan University, Khoramabad, Iran.
2*- Department of Computer Engineering, Faculty of Engineering, Yazd University, Yazd, Iran.
3- Department of Electrical Engineering, Shahid Bahonar university of Kerman, Kerman, Iran.
[1]dowlatshahi.mb@lu.ac.ir, [2*]vderhami@yazd.ac.ir, and [3]nezam@uk.ac.ir

Corresponding author's address: Vali Derhami, Computer Engineering Department, Faculty of Engineering, Yazd University, Yazd, Iran.

Abstract- Gravitational Search Algorithm (GSA) is a simple and efficient optimization method recently proposed for solving single-objective optimization problems. In this paper, for the first time, the nearest-better neighborhoods are defined in swarm intelligence algorithms and then used in the GSA to solve multi-modal optimization problems. For this purpose, two neighborhoods are defined, called Topological Nearest-Better (TNB) and Distance-based Nearest-Better (DNB), and then these two structures are used separately in the GSA and two different versions of the GSA for multi-modal optimization problems are provided. To investigate the efficiency of the proposed algorithms, an empirical assessment has been performed on several standard multi-modal benchmark functions. The results of these experiments show that the proposed algorithms can achieve good results compared to other multi-modal optimizer algorithms.

Keywords- Gravitational Search Algorithm, Swarm Intelligence, Nearest-better Neighborhood, Multimodal Optimization.

مجله علمی پژوهشی رایانش نرم و فناوری اطلاعات
دانشگاه صنعتی نوشیروانی بابل
صفحه مجله: jscit.nit.ac.ir
جلد ۸، شماره ۳، پاییز ۱۳۹۸، صفحه ۱۹-۱۰
دریافت: ۱۳۹۶/۰۶/۰۲، بازنگری: ۱۳۹۸/۰۳/۱۳، پذیرش: ۱۳۹۸/۰۴/۰۸

الگوریتم جستجوی گرانشی با همسایگی نزدیکترین-بهتر برای حل مسائل بهینه‌سازی چندمُدی

محمدباقر دولتشاهی[۱]، ولی درهمی*[۲]، حسین نظام آبادی پور[۳]
۱- استادیار، دانشکده فنی و مهندسی، گروه مهندسی کامپیوتر، دانشگاه لرستان، خرم آباد، ایران.
*۲- نویسنده مسئول: دانشیار، پردیس فنی و مهندسی، گروه مهندسی کامپیوتر، دانشگاه یزد، یزد، ایران.
۳- استاد، دانشکده فنی و مهندسی، گروه مهندسی برق، دانشگاه شهید باهنر کرمان، کرمان، ایران.
[1]dowlatshahi.mb@lu.ac.ir, [2*]vderhami@yazd.ac.ir, and [3]nezam@uk.ac.ir

* نشانی نویسنده مسئول: ولی درهمی، یزد، صفائیه، دانشگاه یزد، پردیس فنی و مهندسی، گروه مهندسی کامپیوتر.

چکیده- الگوریتم جستجوی گرانشی، یک روش بهینه‌سازی ساده و کارامد است که اخیرا برای حل مسائل بهینه‌سازی تک‌هدفه ارائه شده است. در این مقاله، برای اولین بار ساختار همسایگی نزدیکترین-بهتر در الگوریتم‌های هوش جمعی تعریف شده و سپس در الگوریتم جستجوی گرانشی برای حل مسائل بهینه‌سازی چندمُدی استفاده شده است. برای این منظور، ابتدا دو ساختار همسایگی "نزدیکترین-بهتر توپولوژیکی" و "نزدیکترین-بهتر مبتنی بر فاصله" تعریف‌شده، سپس این دو ساختار به طور مجزا در الگوریتم جستجوی گرانشی استفاده شده و دو نسخه‌ی مختلف از الگوریتم جستجوی گرانشی برای حل مسائل بهینه‌سازی چندمُدی ارائه می‌شود. برای بررسی کارایی الگوریتم‌های پیشنهادی، یک ارزیابی تجربی روی چندین تابع محک چندمُدی استاندارد صورت گرفته است. نتایج این آزمایشات نشان می‌دهد که الگوریتم‌های پیشنهادی می‌توانند نتایج خوبی نسبت به سایر الگوریتم‌های بهینه‌ساز چندمُدی به دست آورند.

واژه‌های کلیدی: الگوریتم جستجوی گرانشی، هوش جمعی، همسایگی نزدیکترین-بهتر، بهینه‌سازی چندمُدی.

۱- مقدمه

یک مسأله بهینه‌سازی، عبارت است از یافتن بهترین راه‌حل از میان همه راه‌حل‌های ممکن. بهینه‌سازی به طور گسترده‌ای در مسائل دنیای واقعی ظاهر می‌شود، برای مثال هدف مهندسان طراحی یک محصول با بهترین عملکرد می‌باشد، بازرگانان دنبال به حداکثر رساندن سود حاصل از معاملات خود هستند، سرمایه‌گذاران سعی می‌کنند ریسک سرمایه‌گذاری را به حداقل برسانند، و غیره [۱].

ساده‌ترین نوع مسائل بهینه‌سازی، مسائل بهینه‌سازی تک‌هدفه-تک‌مُدی هستند که در آنها یک تابع هدف وجود دارد. هدف از حل این نوع مسائل، پیدا کردن یک بردار در فضای تصمیم است که متناظر با نقطه‌ی بهینه‌ی سراسری تابع هدف است [۲]. دسته‌ی دیگری از مسائل بهینه‌سازی وجود دارند که با نام مسائل بهینه‌سازی تک‌هدفه-چندمُدی (یا به اختصار مسائل بهینه‌سازی چندمُدی[۱]) شناخته می‌شوند. همانند مسائل بهینه‌سازی تک‌هدفه-تک‌مُدی، در این مسائل نیز یک تابع هدف وجود دارد و محدودیتی روی تعداد متغیرهای تصمیم وجود ندارد. اما برخلاف مسائل بهینه‌سازی تک‌هدفه-تک‌مُدی، هدف از حل مسائل بهینه‌سازی چندمُدی پیدا کردن فقط یک بردار در فضای تصمیم که متناظر با نقطه‌ی بهینه‌ی سراسری تابع هدف باشد نیست، بلکه هدف پیدا کردن تمام بردارهایی در فضای تصمیم است که متناظر با تمام نقاط بهینه‌ی سراسری و بهینه‌ی محلی تابع هدف هستند [۳].

هر مسأله بهینه‌سازی را می‌توان با دو دسته از الگوریتم‌ها حل کرد [۴]. دسته اول، *الگوریتم‌های دقیق*[۲] هستند که راه‌حل‌های بهینه را برای این مسائل پیدا کرده و بهینگی این راه‌حل‌ها را نیز ضمانت

می‌کنند. دسته دوم، *الگوریتم‌های تقریبی*[۳] هستند. این الگوریتم‌ها معمولاً برای حل نمونه‌های بزرگ مسائل بهینه‌سازی که روش‌های دقیق قادر به حل آنها نیستند، ارائه شده‌اند. بر خلاف الگوریتم‌های دقیق، الگوریتم‌های تقریبی قادرند جواب‌هایی با کیفیت مناسب را برای نمونه‌های بزرگ مسائل بهینه‌سازی در یک مدت زمان معقول تولید کنند. متاسفانه، در این الگوریتم‌ها هیچ تضمینی برای پیدا کردن مقدار بهینه سراسری مسأله داده نمی‌شود. در این کلاس از الگوریتم‌ها دو زیر کلاس عمده قابل تمایز هستند: *الگوریتم‌های تقریب‌زن*[۴] و *الگوریتم‌های ابتکاری*[۵]. الگوریتم‌های تقریب‌زن، برای مقدار انحراف جواب پیدا شده با بهینه سراسری، یک کران ارائه می‌دهند. همچنین، در این الگوریتم‌ها برای بدترین حللت الگوریتم یک کران زمانی ارائه می‌شود. برخلاف الگوریتم‌های تقریب‌زن، اکثر الگوریتم‌های ابتکاری هیچ کرانی برای مقدار انحراف جواب پیدا شده با بهینه سراسری، و حتی حداکثر زمان مورد نیاز الگوریتم ارائه نمی‌دهند. الگوریتم‌های ابتکاری خود به دو خانواده عمده تقسیم می‌شوند: *الگوریتم‌های ابتکاری خاص*[۶] و *الگوریتم‌های فرا ابتکاری*[۷]. الگوریتم‌های ابتکاری خاص (همانطور که از نامشان پیداست)، برای حل یک مسأله یا حتی یک نمونه مسأله خاص طراحی می‌شوند. برعکس، الگوریتم‌های فرا ابتکاری الگوریتم‌هایی همه‌منظوره بوده که می‌توانند تقریباً برای حل هر مسأله بهینه‌سازی استفاده شوند [۴].

الگوریتم‌های فرا ابتکاری، دسته‌ای از روش‌های جستجو هستند که می‌توانند جواب‌های نزدیک به بهینه مسائل بهینه‌سازی پیچیده را با کاهش دادن موثر فضای جستجوی این مسائل و بررسی کارآمد این فضا به دست آورند. این الگوریتم‌ها هیچ تضمینی برای پیدا کردن مقادیر بهینه یا حتی کرانی برای راه‌حل پیدا شده ارائه نمی‌دهند. با این وجود، با توجه به قابلیت‌هایی که این الگوریتم‌ها در عمل از خود نشان داده‌اند، امروزه از این الگوریتم‌ها در حل مسائل بهینه‌سازی مختلف استفاده می‌شود. بر خلاف الگوریتم‌های دقیق، الگوریتم‌های فرا ابتکاری قادرند جواب‌هایی با کیفیت مناسب را در یک مدت زمان معقول برای مسائل بهینه‌سازی سخت با اندازه متوسط پیدا کنند [۴].

به طور کلی الگوریتم‌های فرا ابتکاری را می‌توان به دو گروه تقسیم کرد: *مبتنی بر جمعیت*[۸] در مقابل *مبتنی بر تک‌راه‌حل*[۹]. الگوریتم‌های فرا ابتکاری تک‌راه‌حله در خلال فرآیند جستجو فقط یک راه‌حل را دستکاری می‌کنند، در صورتیکه در الگوریتم‌های فرا ابتکاری مبتنی بر جمعیت در خلال فرآیند جستجو یک جمعیت از راه‌حل‌ها دستکاری می‌شود. این دو دسته از الگوریتم‌های فرا ابتکاری ویژگی‌های مکمل هم بودن را دارا هستند: الگوریتم‌های فرا ابتکاری‌ـتک‌راه‌حله معمولاً بر قابلیت بهره‌گیری تاکید دارند، در صورتیکه الگوریتم‌های فرا ابتکاری مبتنی بر جمعیت معمولاً بر قابلیت کاوش تاکید دارند [۴]. مشهورترین الگوریتم‌های فرا ابتکاری مبتنی بر جمعیت عبارتند از: الگوریتم‌های تکاملی[۱۰] [۵] و الگوریتم‌های هوش جمعی[۱۱] [۶].

الگوریتم‌های هوش جمعی، دسته‌ای از الگوریتم‌های فرا ابتکاری تصادفی[۱۲] مبتنی بر جمعیت هستند که از رفتارهای جمعی موجود در طبیعت الهام گرفته‌اند. مهمترین مولفه در این الگوریتم‌ها، عناصری به نام *ذره*[۱۳] هستند که معمولاً عامل‌هایی ساده و غیرپیچیده می‌باشند [۴]. الگوریتم بهینه‌ساز جمعیت ذرات[۱۴] [۷]، الگوریتم جستجوی گرانشی[۱۵] [۸]، و الگوریتم بهینه‌ساز جمعیت مورچگان[۱۶] [۹] از جمله الگوریتم‌های هوش جمعی‌ای هستند که تا امروز در کاربردهای مختلفی استفاده شده‌اند.

الگوریتم جستجوی گرانشی [۸] یک الگوریتم فرا ابتکاری تصادفی مبتنی بر جمعیت و مبتنی بر تکرار می‌باشد که اخیرا توسط محققان ایرانی با الهام از طبیعت برای حل مسائل بهینه‌سازی پیوسته معرفی شده است [۸]. ایده اصلی این الگوریتم، شبیه‌سازی قانون گرانش نیوتون و قوانین حرکت بر روی یک جمعیت از جرم‌ها در یک فضای پیوسته است. در این مقاله، برای اولین بار ساختار همسایگی "نزدیکترین-بهتر[۱۷]" برای الگوریتم‌های هوش جمعی تعریف شده و در الگوریتم جستجوی گرانشی برای حل مسائل بهینه‌سازی چندمُدی استفاده شده است. برای این منظور، ابتدا دو ساختار همسایگی "نزدیکترین-بهتر توپولوژیکی[۱۸]" و "نزدیکترین-بهتر مبتنی بر فاصله[۱۹]" تعریف شده، سپس این دو ساختار به طور مجزا در الگوریتم جستجوی گرانشی استفاده شده و دو نسخه‌ی مختلف از الگوریتم جستجوی گرانشی برای حل مسائل بهینه‌سازی چندمُدی ارائه می‌شود. برای بررسی کارایی الگوریتم‌های پیشنهادی،ـیک ارزیابی تجربی روی چندین تابع محک چندمُدی استاندارد صورت گرفته است. نتایج این آزمایشات نشان می‌دهد که الگوریتم‌های پیشنهادی می‌توانند نتایج خوبی نسبت به سایر الگوریتم‌های بهینه‌ساز چندمُدی به دست آورند.

ادامه مقاله به این صورت سازماندهی شده است: در بخش ۲ مروری بر الگوریتم جستجوی گرانشی و مشهورترین الگوریتم‌های هوش جمعی برای حل مسائل بهینه‌سازی چندمُدی خواهیم داشت. در بخش ۳ الگوریتم‌های پیشنهادی برای حل مسائل بهینه‌سازی چندمُدی شرح داده خواهند شد. در بخش ۴ نتایج حاصل از پیاده-سازی الگوریتم‌های پیشنهادی در حل برای حل مسائل بهینه‌سازی چندمُدی و همچنین مقایسه این نتایج با سایر الگوریتم های حل

مسائل بهینه‌سازی چندمُدی آورده شده است. در بخش ۵ نیز نتیجه‌گیری آورده شده است.

۲- کارهای مرتبط

در این بخش ابتدا الگوریتم جستجوی گرانشی تشرح می‌شود، و سپس مشهورترین الگوریتم‌های هوش جمعی برای حل مسائل بهینه‌سازی چندمُدی تشریح خواهند شد.

۲-۱- الگوریتم جستجوی گرانشی

الگوریتم جستجوی گرانشی [۸] یک الگوریتم فرا ابتکاری تصادفی مبتنی بر جمعیت و مبتنی بر تکرار می‌باشد که اخیرا توسط محققان ایرانی با الهام از طبیعت برای حل مسائل بهینه‌سازی پیوسته معرفی شده است [۸]. ایده اصلی این الگوریتم، شبیه‌سازی قانون گرانش نیوتون و قوانین حرکت بر روی یک جمعیت از جرم‌ها در یک فضای پیوسته n-بُعدی می‌باشد.

اولین گام از الگوریتم جستجوی گرانشی، تشکیل یک سیستم مصنوعی n-بُعدی با N عامل جستجوگر (یا اصطلاحا "جرم") که به صورت تصادفی مقداردهی شده‌اند، است. بر اساس [۸]، جرم گرانشی عامل i در زمان t (که با نماد $M_i(t)$ نشان داده می‌شود)، پس از محاسبه شایستگی جمعیت فعلی جرم‌ها با استفاده از رابطه‌های (۱) و (۲) محاسبه می‌شود:

$$q_i(t)=\frac{fit_i(t)-worst(t)}{best(t)-worst(t)}\ , \qquad (۱)$$

$$M_i(t)=\frac{q_i(t)}{\sum_{j=1}^{N} q_j(t)}\ , \qquad (۲)$$

که در آن $M_i(t)$ و $fit_i(t)$ به‌ترتیب جرم گرانشی و میزان شایستگی عامل i را در زمان t نشان می‌دهند. همچنین $best(t)$ و $worst(t)$ به ترتیب نماینده بهترین و بدترین جرم‌های جمعیت در زمان t هستند؛ مقدار این دو پارامتر برای مسائل کمینه‌سازی، به ترتیب با استفاده از رابطه‌های (۳) و (۴) محاسبه می‌شود (برای مسائل بیشینه‌سازی جای این دو عوض می‌شود):

$$best(t)=\min_{j\in\{1,\ldots,N\}} fit_j(t), \qquad (۳)$$

$$worst(t)=\max_{j\in\{1,\ldots,N\}} fit_j(t)\ . \qquad (۴)$$

پس از محاسبه جرم گرانشی برای هر عامل، حال می‌توان شتاب، سرعت و موقعیت جدید هر عامل را در فضای جستجو محاسبه کرد. برای محاسبه شتاب عامل i در زمان t، برآیند نیروهای وارد شده از جانب K عامل برتر جمعیت بر روی عامل i بر اساس قانون گرانش محاسبه می‌شود. سپس برای محاسبه سرعت یا میزان جابجایی عامل i در زمان $t+1$، کسری از سرعت این عامل در زمان t با شتاب این عامل در زمان t جمع می‌شود (رابطه (۷)). در نهایت موقعیت جدید عامل i، می تواند بر اساس رابطه (۸) محاسبه شود.

$$F_i^d(t)=\sum_{j\in Kbest,\ j\neq i} rand_j G(t)\frac{M_j(t)M_i(t)}{R_{ij}(t)+\varepsilon}(x_j^d(t)-x_i^d(t))\ , \qquad (۵)$$

$$a_i^d(t)=\frac{F_i^d(t)}{M_i(t)}=\sum_{j\in Kbest,\ j\neq i} rand_j G(t)\frac{M_j(t)}{R_{ij}(t)+\varepsilon}(x_j^d(t)-x_i^d(t))\ , \qquad (۶)$$

$$v_i^d(t+1)=rand_i\times v_i^d(t)+a_i^d(t)\ , \qquad (۷)$$

$$x_i^d(t+1)=x_i^d(t)+v_i^d(t+1)\ . \qquad (۸)$$

شبه کد الگوریتم جستجوی گرانشی در الگوریتم (۱) آمده است.

الگوریتم (۱): شبه کد الگوریتم جستجوی گرانشی

۱. مقداردهی اولیه به پارامترها.
۲. مقداردهی اولیه به عامل‌ها به صورت تصادفی.
۳. تا زمانیکه شرایط پایانی برآورده نشده است، مراحل زیر را انجام بده:
٤. ارزیابی شایستگی عامل‌ها.
٥. محاسبه M برای هر عامل.
٦. به روز رسانی پارامترهای G، K، و $Kbest$.
۷. محاسبه شتاب وارده به هر عامل.
۸. محاسبه سرعت هر عامل.
۹. به روز رسانی موقعیت هر عامل.
۱۰. برگرداندن بهترین راه‌حل پیدا شده.

۲-۲- الگوریتم جستجوی گرانشی و حل مسائل بهینه‌سازی چندمُدی

طبق مطالعات صورت گرفته، تابحال تنها یک مقاله چاپ شده که به طور اختصاصی به ارائه‌ی یک نسخه جدید از الگوریتم جستجوی گرانشی برای حل مسائل بهینه‌سازی چند مُدی پرداخته است، که در آن نیز از ایده‌ی همسایگی نزدیکترین بهتر استفاده نشده است. یزدانی و همکاران در [۱۰] برای اولین بار به معرفی یک نسخه اختصاصی از الگوریتم جستجوی گرانشی به نام NGSA برای حل مسائل بهینه‌سازی چند مُدی پرداختند. تفاوت اصلی NGSA با GSA در این است که در NGSA هر عامل به سمت K عاملی که

کمترین فاصله را در فضای جستجو با آن دارند حرکت می‌کند (در حالیکه در GSA هر عامل به سمت K عامل از جمعیت که بهترین شایستگی را دارند حرکت می‌کند). یزدانی و همکاران [۱۰] با انجام آزمایشاتی نشان دادند که الگوریتم پیشنهادی‌شان در مقایسه با بسیاری از الگوریتم‌های ارائه شده برای حل مسائل چند مُدی، کارایی بهتری دارد.

به طور کلی، برای حل مسائل بهینه‌سازی چند مُدی با استفاده از الگوریتم جستجوی گرانشی بایستی به دو سوال اساسی زیر پاسخ داده شود:

- چگونه اعضای مجموعه Kbest انتخاب شوند که باعث ایجاد تنوع در جمعیت به منظور جلوگیری از همگرایی به یک راه‌حل واحد شوند؟
- چگونه با استفاده از اعضای مجموعه Kbest و موقعیت فعلی یک عامل، سرعت جدید آن عامل محاسبه شود؟

در این مقاله، ما بر روی مورد اول متمرکز خواهیم شد و دو روش جدید برای انتخاب اعضای مجموعه Kbest ارائه خواهیم کرد.

۳-۲- الگوریتم بهینه‌ساز جمعیت ذرات و حل مسائل بهینه-سازی چندمُدی

تابحال محققان زیادی بر روی حل مسائل بهینه‌سازی چندمُدی با استفاده از الگوریتم بهینه‌ساز جمعیت ذرات کار کرده‌اند. در این بخش به بررسی و تشریح برخی از مهمترین کارهای انجام شده در این زمینه خواهیم پرداخت.

بریتس و همکاران [۱۱] الگوریتمی به نام NichePSO را برای حل مسائل بهینه‌سازی چند مُدی معرفی کردند که در آن چندین زیرجمعیت از جمعیت اولیه با توجه به شایستگی ذرات ساخته می-شود. سپس، اوزکان و همکاران [۱۲] عملکرد NichePSO را با استفاده از مکانیزم تپه نوردی بهبود دادند. علاوه بر کارایی نه چندان مناسب، ضعف دیگر این الگوریتم‌ها این است که نیازمند مجموعه‌ای از پارامترهای مرتبط با هر نیچ[۲۰] هستند.

پاسارو و اِستاریتا [۱۳] یک الگوریتم بهینه‌ساز جمعیت ذرات به نام الگوریتم k-means-PSO را برای حل مسائل بهینه‌سازی چند مُدی ارائه دادند که در آن با استفاده از الگوریتم خوشه‌بندی k-means به خوشه‌بندی جمعیت به منظور شناسایی نیچ‌ها می‌پردازد. از طریق حل مجموعه‌ای از توابع آزمون استاندارد، الگوریتم k-means-PSO عملکرد بهتری نسبت به برخی از الگوریتم‌های موجود تا آن زمان را نشان می‌داد. اگر چه این الگوریتم جواب‌های خوبی در مقایسه با کارهای زمان خود به دست آورد، اما نیازمند تنظیم پارامتر تعداد خوشه‌ها و همچنین تعداد مراحل تکرار الگوریتم خوشه‌بندی k-means می‌باشد، که این موضوع استفاده از آن را برای حل مسائل دنیای واقعی با مشکل مواجه می‌سازد.

لی [۱۴] یک الگوریتم بهینه‌ساز جمعیت ذرات به نام الگوریتم RPSO بر اساس توپولوژی حلقه‌ای را برای حل مسائل بهینه‌سازی چند مُدی ارائه داد که مزیت اصلی آن این است که به هیچ پارامتر اضافه‌ای (مثل تعداد نیچ‌ها یا تعداد خوشه‌ها) وابسته نیست. لی نشان داد که الگوریتم RPSO قادر است به خوبی مسائل چندمدی را حل کند. یک نقطه ضعف بزرگ RPSO این است که در توپولوژی حلقه‌ای امکان ایجاد ارتباط بین ذرات موجود در نیچ‌های مختلف نیز می‌شود که این موضوع باعث کُند شدن قدرت همگرایی الگوریتم می‌شود. برای مثال، اگر دو ذره در دو نیچ مختلف قرار گرفته باشند، حرکت آنها به سمت هم باعث ایجاد نوسان[۲۱] در حرکت ذره می‌شود که نوسان بین دو نیچ تاثیر نامطلوبی در کاوش و بهره‌گیری الگوریتم دارد. با این حال، لی نشان داده است که RPSO در برخی مسائل قادر به تشکیل نیچ‌های پایدار است.

کیو و همکاران [۱۵] یک الگوریتم بهینه‌ساز جمعیت ذرات به نام الگوریتم جمعیت ذرات اطلاع‌دهنده‌ی محلی یا به اختصار LIPS را برای حل مسائل بهینه‌سازی چند مُدی ارائه دادند که در آن موقعیت جدید یک ذره با استفاده از موقعیت k تا از نزدیکترین همسایه‌هایش محاسبه می‌شود. در این الگوریتم از فاصله‌ی اقلیدسی برای تعیین نزدیکترین همسایه‌های هر ذره استفاده شده است. کیو و همکاران [۱۵] به مقایسه الگوریتم LIPS با چندین الگوریتم مشهور هوش جمعی برای حل مسائل بهینه‌سازی چند مُدی پرداخته‌اند که نتایج تجربی نشان می‌دهد که LIPS قادر به ارائه عملکرد آماری بهتری خواهد بود. با وجود اینکه الگوریتم LIPS یکی از بهترین الگوریتم-هایی است که تا امروز برای حل مسائل بهینه‌سازی چند مُدی ارائه شده است، اما این الگوریتم دارای یک ضعف اساسی است: برای محاسبه سرعت ذره‌ی i، "شایستگی ذرات همسایه ذره‌ی i" و "فاصله‌ی ذرات همسایه ذره‌ی i" را در نظر نمی‌گیرد، درصورتیکه برای ایجاد یک تعادل مناسب بین همگرایی و تنوع، در نظر گرفتن این دو پارامتر در محاسبه سرعت می‌تواند کمک زیادی کند.

۳- الگوریتم‌های پیشنهادی

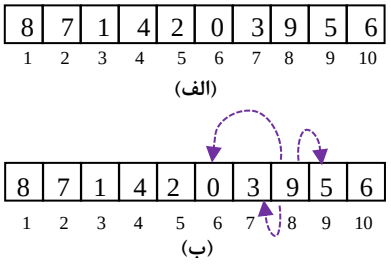
چنانکه گفته شد یکی از مهمترین سوالاتی که در حل یک مساله توسط الگوریتم جستجوی گرانشی بایستی پاسخ داده شود این است

که: چگونه اعضای مجموعه Kbest انتخاب شوند که باعث ایجاد تنوع در جمعیت به منظور جلوگیری از همگرایی به یک راه‌حل واحد شوند؟ برخلاف الگوریتم جستجوی گرانشی استاندارد که در آن K بهترین عضو جمعیت مجموعه Kbest را می‌سازند و مجموعه Kbest برای همه اعضای جمعیت در یک تکرار ثابت است، در دو الگوریتم پیشنهادی مجموعه Kbest برای هر عضو از جمعیت منحصر به فرد است و توسط K همسایه نزدیکترین-بهتر توپولوژیکی آن عضو ساخته می‌شود. در ادامه به تشریح جزئیات دو الگوریتم پیشنهادی پرداخته می‌شود.

۳-۱- الگوریتم جستجوی گرانشی با همسایگی نزدیکترین-بهتر توپولوژیکی[۲۲] (TNB-GSA)

ساختار این الگوریتم دقیقا شبیه به الگوریتم جستجوی گرانشی استاندارد است، به استثنای اینکه مجموعه Kbest برای هر عضو از جمعیت منحصر به فرد است و توسط K همسایه نزدیکترین-بهتر توپولوژیکی آن ساخته می‌شود.

شکل (۱) را در نظر بگیرید. در قسمت (الف)، یک آرایه با ۱۰ عنصر دیده می‌شود که در آن هر عنصر نماینده یک عضو از جمعیت است و مقداری که در هر عنصر نوشته شده است بیانگر مقدار تابع هدف به ازای آن عضو از جمعیت است (فرض کنید تابع هدف یک تابع کمینه‌سازی است، یعنی هر چه مقدار تابع هدف یک عضو کمتر باشد، آن عضو دارای کیفیت بهتری است). فرض کنید قصد پیدا کردن ۳ همسایه نزدیکترین-بهتر توپولوژیکی را برای عضوی از جمعیت (آرایه) با اندیس ۸ و مقدار تابع هدف ۹ را داریم. در این حالت، اعضایی از جمعیت که اولا مقدار تابع هدف کمتری از این عضو را دارند و دوما دارای کمترین فاصله اندیسی با این عضو هستند، مجموعه ۳ همسایه نزدیکترین-بهتر توپولوژیکی را تشکیل می‌دهند. برای عضوی از جمعیت با اندیس ۸، اعضای مجموعه Kbest به‌ترتیب عبارتند از اعضایی با اندیس های ۷، ۹، و ۶. شکل (۱) قسمت (ب) این اعضا از جمعیت را به صورت گرافیکی نشان می‌دهد.



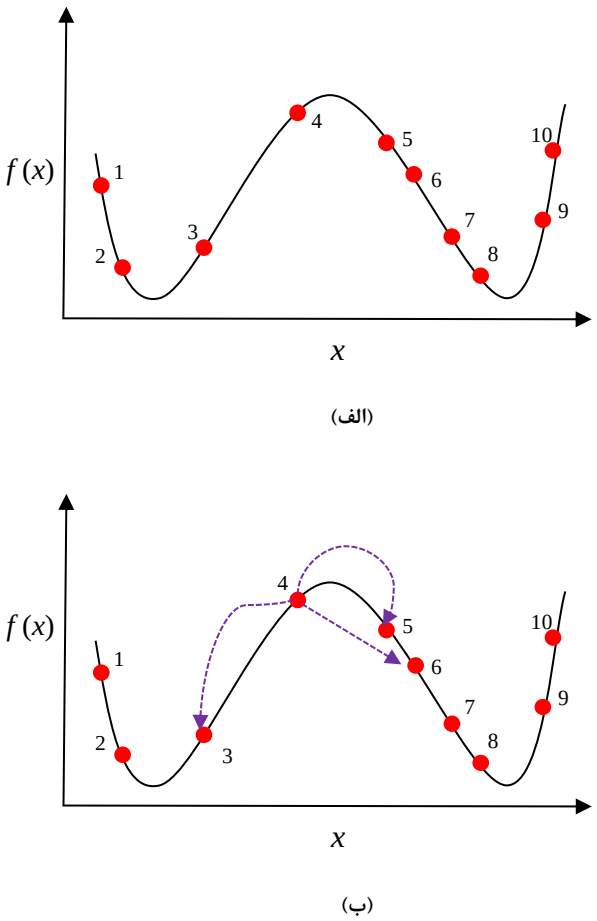
شکل (۱): همسایگی نزدیکترین-بهتر توپولوژیکی برای یک مساله کمینه‌سازی.

۳-۲- الگوریتم جستجوی گرانشی با همسایگی نزدیکترین-بهتر مبتنی بر فاصله[۲۳] (DNB-GSA)

ساختار این الگوریتم نیز دقیقا شبیه به الگوریتم جستجوی گرانشی استاندارد و همچنین الگوریتم TNB-GSA است، به استثنای اینکه مجموعه Kbest برای هر عضو از جمعیت منحصر به فرد است و توسط K همسایه نزدیکترین-بهتر مبتنی بر فاصله آن عضو ساخته می‌شود. در ادامه نحوه ایجاد مجموعه Kbest برای یک عضو از جمعیت توسط الگوریتم DNB-GSA مبتنی بر فاصله شرح داده خواهد شد.

در الگوریتم DNB-GSA، مجموعه Kbest برای یک عضو شامل اعضایی از جمعیت است که اولا مقدار تابع هدف کمتری از این عضو را دارند و دوما دارای کمترین فاصله در فضای تصمیم[۲۴] با این عضو هستند، دقت داشته باشید که تنها فرق بین دو الگوریتم TNB-GSA و DNB-GSA در متری است که از آن برای اندازه‌گیری فاصله استفاده می‌شود، که در الگوریتم اول فاصله در فضای اندیس اعضای جمعیت و در الگوریتم دوم فاصله در فضای تصمیم اندازه‌گیری می‌شود. فضای تصمیم عبارت است از فضایی که در آن متغیرهای تصمیم مساله تعریف می‌شوند.

شکل (۲) را برای تابع تک متغیره $f(x)$ در نظر بگیرید. در قسمت (الف)، متغیر تصمیم x به‌تنهایی فضای تصمیم را شکل می‌دهد (دقت شود که در مسائلی که دارای چندین متغیر تصمیم هستند، تمام متغیرها با هم فضای تصمیم را در فضای دکارتی می‌سازند). در این شکل اعدادی که در کنار هر راه‌حل نوشته شده است، اندیس آن راه‌حل است. فرض کنید قصد پیدا کردن ۳ همسایه نزدیکترین-بهتر مبتنی بر فاصله را برای عضوی از جمعیت با اندیس ۴ داریم. در این حالت، اعضایی از جمعیت که اولا مقدار تابع هدف کمتری را از این عضو دارند و دوما دارای کمترین فاصله اقلیدسی در فضای تصمیم (متغیر x) با این عضو هستند، مجموعه ۳ همسایه نزدیکترین-بهتر مبتنی بر فاصله را تشکیل می‌دهند. برای عضوی از جمعیت با اندیس ۴، اعضای مجموعه Kbest به‌ترتیب عبارتند از اعضایی با اندیس های ۵، ۶، و ۳. شکل (۲) قسمت (ب) این اعضا از جمعیت را به صورت گرافیکی نشان می‌دهد.



شکل (۲): همسایگی نزدیکترین-بهتر مبتنی بر فاصله برای یک مساله کمینه-سازی.

۴- نتایج شبیه‌سازی و مقایسه

در این بخش، به بررسی نتایج شبیه‌سازی و ارزیابی اثربخشی دو الگوریتم پیشنهادی برای حل توابع استاندارد بهینه‌سازی چندمُدی خواهیم پرداخت. در ادامه، ابتدا به معرفی و بررسی ویژگی‌های توابع محک انتخاب‌شده خواهیم پرداخت. سپس، تنظیمات شبیه‌سازی الگوریتم‌های پیشنهادی شرح داده خواهد شد. در نهایت، نتایج به‌دست آمده از اجرای الگوریتم‌های پیشنهادی و سایر الگوریتم‌های مشهور بهینه‌سازی چندمُدی را آورده و به بررسی و تحلیل آن‌ها خواهیم پرداخت. نتایج آزمایشات انجام شده نشان خواهند داد که هر دو الگوریتم‌های پیشنهادی می‌توانند نتایج خوبی نسبت به سایر الگوریتم‌های بهینه‌ساز چندمُدی به دست آورند، به خصوص الگوریتم DNB-GSA که قادر است جواب‌های بسیار خوبی تولید کند.

۴-۱- توصیف توابع محک

در این مقاله، دو مجموعه از توابع محک استاندارد برای ارزیابی عملکرد روش پیشنهادی استفاده می‌شود. جدول (۱) اطلاعات عمومی ۱۵ تابع محک کلاسیک که در مقالات منتشر شده‌ی مختلف استفاده شده است را لیست می‌کند. توجه داشته باشید که این توابع همگی توابع بیشینه‌سازی هستند، یعنی هدف پیداکردن نقاطی از تابع هدف است که به‌ازای آن‌ها مقدار تابع هدف بیشینه می‌شود. جدول (۲) اطلاعات عمومی ۸ تابع محک پیچیده که اخیرا در [۱۶] ارائه شده‌اند را نشان می‌دهد. توجه داشته باشید که این توابع محک، همگی توابع کمینه‌سازی هستند، یعنی هدف پیداکردن نقاطی از تابع هدف است که به‌ازای آن‌ها مقدار تابع هدف کمینه می‌شود. در جدول‌های (۱) و (۲)، تعداد ابعاد و تعداد قله‌های سراسری (یا حتی محلی) توابع محک استفاده‌شده، آمده است. چنانکه دیده می‌شود توابع جدول (۲) برای ابعاد مختلفی قابل تعریف هستند، درصورتیکه توابع جدول (۱) چنین امکانی را به ما نمی‌دهند.

جدول (۱): مشخصات ۱۵ تابع محک کلاسیک

نام تابع محک	ابعاد	تعداد قله‌های سراسری
F1: Equal maxima [15]	۱	۵
F2: Decreasing maxima [15]	۱	۱
F3: Uneven maxima [15]	۱	۵
F4: Uneven decreasing maxima [15]	۱	۱
F5: Himmelblau’s function [15]	۲	۴
F6: Six-hump camel back [15]	۲	۲
F7: Shekel’s foxholes [15]	۲	۱
F8: Inverted Shubert function [15]	۲	۱۸
F9: Waves [15]	۲	۱۰
F10: Sphere	۱۰	۱
F11: Branin RCOS [15]	۲	۳
F12: Ackley [15]	۲	۱
F13: Michalewicz [15]	۲	۱
F14: Ursem F1 [15]	۲	۱
F15: Ursem F3 [15]	۲	۱

جدول (۲): مشخصات ۸ تابع محک پیچیده [۱۶]

نام تابع محک	ابعاد	تعداد قله‌های سراسری/ محلی
F16: Shifted and rotated expanded two-peak trap	۵	۱۵/۱
	۱۰	۵۵/۱
	۲۰	۲۱۰/۱
F17: Shifted and rotated expanded Five-Uneven-Peak Trap	۲	۲۱/۴
	۵	۰/۳۲
	۸	۰/۲۵۶
F18: Shifted and rotated Expanded Equal Minima	۲	۰/۲۵
	۳	۰/۱۲۵
	۴	۰/۶۲۵
F19: Shifted and rotated Expanded Decreasing Minima	۵	۱۵/۱
	۱۰	۵۵/۱
	۲۰	۲۱۰/۱
F20: Shifted and rotated expanded uneven minima	۲	۰/۲۵
	۳	۰/۱۲۵
	۴	۰/۶۲۵
F21: Shifted and rotated expanded Himmelblau's function	۴	۰/۱۶
	۶	۰/۶۴
	۸	۰/۲۵۶
F22: Shifted and rotated expanded six-hump camel back	۶	۰/۸
	۱۰	۰/۳۲
	۱۶	۰/۲۵۶
F23: Shifted and rotated modified Vincent function	۲	۰/۳۶
	۳	۰/۲۱۶
	۴	۰/۱۲۹۶

۲-۴- تنظیمات شبیه‌سازی

به‌دلیل این‌که پیداکردن راه‌حل‌های بهینه در توابع هدف پیوسته کار بسیار دشواری است، در تمام تحقیقات قبلی انجام‌شده، معمولا یک سطح دقت (ε) در نظر گرفته می‌شود که نشان می‌دهد راه‌حل‌های پیداشده توسط الگوریتم چقدر به راه‌حل‌های بهینه‌ی واقعی نزدیک هستند. همچنین، برای اطمینان از این‌که بهینه‌های پیدا‌شده با هم متفاوت هستند، یک پارامتر شعاع نیچینگ (r) در نظر گرفته می‌شود. به‌عبارت دیگر، یک راه‌حل پیداشده توسط الگوریتم به‌عنوان یک راه‌حل بهینه گزارش می‌شود اگر همزمان دو شرط زیر برقرار باشد: (۱) اختلاف بین مقدار هدف آن راه‌حل با مقدار هدف یکی از راه‌حل‌های بهینه‌ی واقعی کمتر از اپسیلون (ε) باشد، و (۲) فاصله اقلیدسی بین آن راه‌حل با یکی از راه‌حل‌های بهینه‌ی واقعی کمتر از r باشد.

در جدول (۳)، سطح دقت (ε)، شعاع نیچینگ (r)، اندازه جمعیت، و حداکثر تعداد ارزیابی تابع شایستگی برای هر یک از توابع محک فهرست شده است (این تنظیمات برای تمام الگوریتم‌های مورد مقایسه نیز معتبر است). به طور کلی، اندازه جمعیت و حداکثر تعداد ارزیابی تابع شایستگی به تعداد نقاط بهینه‌ی هر تابع وابسته هستند، به‌طوریکه تعداد زیاد نقاط بهینه نیازمند اندازه جمعیت بزرگتر و تعداد ارزیابی شایستگی بیشتری است. توجه داشته باشید که اندازه‌گیری عملکرد هر یک از الگوریتم‌ها به سطح دقت مشخص‌شده و شعاع نیچینگ وابسته است.

جدول (۳): تنظیمات شبیه‌سازی توابع محک

شماره تابع	تعداد ارزیابی تابع شایستگی	اندازه جمعیت	r	ε
F1	۱۰۰۰۰	۵۰	۰/۰۱	۰/۰۰۰۰۰۱
F2	۱۰۰۰۰	۵۰	۰/۰۱	۰/۰۰۰۰۰۱
F3	۱۰۰۰۰	۵۰	۰/۰۱	۰/۰۰۰۰۰۱
F4	۱۰۰۰۰	۵۰	۰/۰۱	۰/۰۰۰۰۰۱
F5	۱۰۰۰۰	۵۰	۰/۵	۰/۰۰۰۵
F6	۱۰۰۰۰	۵۰	۰/۵	۰/۰۰۰۰۰۱
F7	۱۰۰۰۰	۵۰	۰/۵	۰/۰۰۰۰۱
F8	۱۰۰۰۰۰	۲۵۰	۰/۵	۰/۰۵
F9	۳۰۰۰۰	۲۰۰	۰/۲	۰/۰۰۱
F10	۱۲۵۰۰	۵۰	۰/۲	۰/۰۱
F11	۲۰۰۰۰	۲۰۰	۰/۵	۰/۰۰۱
F12	۱۰۰۰۰	۱۰۰	۰/۵	۰/۰۱
F13	۱۰۰۰۰	۱۰۰	۰/۵	۰/۰۰۰۱
F14	۲۰۰۰۰	۱۰۰	۰/۵	۰/۰۰۰۰۰۱
F15	$۲۰۰۰ * D * \sqrt{q}$	$۵۰۰ * Round(\sqrt{D})$	۰/۱*D	۰/۱
F16-F23				

۳-۴- نتایج و مقایسه‌ها

در این بخش، نتایج تجربی الگوریتم‌های پیشنهادی بر روی توابع محک و همچنین نتایج سایر الگوریتم‌های بهینه‌سازی چندمُدی ارائه خواهد شد. جدول (۴) نتایج حاصل از شبیه‌سازی الگوریتم‌های پیشنهادی و ۱۰ الگوریتم فرا ابتکاری بهینه‌ساز چندمُدی شامل الگوریتم‌های LIPS [۱۵]، r2PSO [۱۴]، r2PSO-LHC [۱۴]، SPSO [۱۷]، FER-PSO [۱۸]، SDE [۱۹]، CDE [۲۰]، TriDEMO [۲۱]، MMODE [۲۲]، و RCSA [۲۳] را برای توابع کلاسیک F1-F15 نشان می‌دهد.

برای ارزیابی یک تابع محک توسط یک الگوریتم، از معیار نرخ موفقیت[۲۵] استفاده شده است. این معیار نشان می‌دهد که الگوریتم مورد نظر قادر است در چند درصد از اجراهای مستقلش توانسته است تمام نقاط بهینه‌ی تابع محک مورد بررسی را پیدا کند. مقدار ۱ برای نرخ موفقیت به این معناست که الگوریتم قادر به پیداکردن تمام بهینه‌ها در تمام اجراهای مستقلش است، و مقدار صفر برای نرخ موفقیت نشان می‌دهد که الگوریتم قادر نبوده است در هیچ یک

از اجراهای مستقلش تمام نقاط بهینه‌ی تابع را پیدا کند. علاوه بر معیار نرخ موفقیت، رتبه‌ی هر الگوریتم در حل هر تابع محک نیز در درون پرانتز آمده است. همچنین، رتبه‌ی کلی هر الگوریتم در ردیف آخر جدول (۴) گزارش شده است. همان‌طور که از نتایج رتبه کلی الگوریتم‌ها دیده می‌شود، الگوریتم DNB-GSA بهترین کارایی را نسبت به سایر الگوریتم‌های فرا ابتکاری بهینه‌سازی چندمُدی مورد مقایسه داشته است. نکته جالب اینجاست که رتبه دوم این مقایسه را نیز الگوریتم TNB-GSA به دست آورده است. این موضوع به خوبی نشان می‌دهد که ساختار همسایگی نزدیکترین-بهتر قادر است یک کنترل خوب بین دو پارامتر همگرایی و تنوع در الگوریتم جستجوی گرانشی به منظور پیدا کردن راه‌حل‌های بهینه مختلف ایجاد کند.

جدول (۵) نتایج حاصل از پیاده‌سازی الگوریتم‌های پیشنهادی و الگوریتم LIPS را توسط معیار متوسط تعداد بهینه‌های پیداشده برای توابع محک F16-F23 نشان می‌دهد. باید دقت داشت که توابع محک چندمُدی F16-F23 بسیار پیچیده‌تر از توابع F1-F15 هستند، به‌طوری‌که برای این توابع محک هیچ‌کدام از الگوریتم‌ها قادر به تولید نرخ موفقیت غیر صفر نیستند. به‌عبارت دیگر، معیار اندازه-گیری نرخ موفقیت معیار مناسبی برای بررسی عملکرد الگوریتم‌های چندمُدی فعلی بر روی این توابع محک نیست، چراکه برای تمام الگوریتم‌ها مقدار صفر را گزارش می‌کند و در این حالت کارایی الگوریتم‌ها به هیچ وجه قابل مقایسه نیست. در این حالت، یک معیار اندازه‌گیری بهتر معیار متوسط تعداد بهینه‌های پیداشده توسط هر یک از الگوریتم‌ها است. این معیار میانگین تعداد بهینه‌هایی که یک الگوریتم در اجراهای مختلفش روی یک تابع محک پیدا می‌کند را گزارش می‌کند. همان‌طور که در جدول (۵) دیده می‌شود، الگوریتم DNB-GSA کارایی بهتری نسبت به سایر الگوریتم‌ها دارد، به-طوری‌که در اکثر توابع محک این الگوریتم توانسته است مقدار متوسط تعداد بهینه‌های بزرگتری را تولید کند.

نتایج به دست آمده از دو جدول (۴) و (۵) به خوبی تایید می‌کنند که ساختار همسایگی نزدیکترین-بهتر بخصوص زمانی که همراه با فاصله اقلیدسی در فضای تصمیم استفاده می‌شود کارایی بسیار خوبی در مقایسه با سایر الگوریتم‌های فرا ابتکاری در حل مسائل بهینه‌سازی چند مُدی از خود نشان می‌دهد. لازم به ذکر است که دلیل اصلی قدرت الگوریتم DNB-GSA در حل مسائل پیچیده، کنترل مناسبی است که این الگوریتم قادر است بین دو پارامتر همگرایی و تنوع راه‌حل‌ها ایجاد کند. در کنار این واقعیت که الگوریتم TNB-GSA از نظر کیفیت جواب‌های پیدا شده دارای کارایی کمتری نسبت به الگوریتم DNB-GSA است، باید به این نکته نیز اشاره شود که الگوریتم TNB-GSA قادر است با صرف زمان کمتری (با صرف پیچیدگی زمانی $O(K\times N^2)$ در هر تکرار) به چنین کیفیتی برای جواب‌ها دست یابد (در صورتیکه الگوریتم DNB-GSA با صرف پیچیدگی زمانی $O(K\times n\times N^2)$ در هر تکرار محاسباتش را انجام می‌دهد). بنابراین، زمانی‌که میزان مصرف زمان توسط یک الگوریتم از اهمیت بالایی برخوردار باشد، چون پیچیدگی زمانی الگوریتم TNB-GSA برتری محسوسی نسبت به سایر الگوریتم‌ها دارد، استفاده از این الگوریتم‌ها می‌تواند در اولویت باشد.

جدول (۴): مقایسه الگوریتم پیشنهادی با ۱۰ الگوریتم فرا ابتکاری دیگر بر اساس معیار نرخ موفقیت

شماره تابع	RCSA	MMODE	TriDEMO	CDE	SDE	FERPSO	SPSO	r2pso-lhc	r2pso	LIPS	TNB-GSA	DNB-GSA
F1	۱ (۱)	۰/۹۲ (۶)	۰/۹۲ (۶)	۰/۲۸ (۱۲)	۰/۷۲ (۱۱)	۰/۸۴ (۱۰)	۰/۸۸ (۹)	۱ (۱)	۰/۹۲ (۶)	۱ (۱)	۱ (۱)	۱ (۱)
F2	۰/۹۲ (۷)	۰/۹۶ (۴)	۰/۹۶ (۴)	۰/۴۸ (۹)	۰ (۱۰)	۰ (۱۰)	۱ (۱)	۰/۶۴ (۸)	۰ (۱۰)	۰/۹۶ (۴)	۱ (۱)	۱ (۱)
F3	۱ (۱)	۱ (۱)	۱ (۱)	۰/۲۸ (۱۲)	۰/۶۰ (۱۱)	۱ (۱)	۰/۹۲ (۸)	۰/۹۲ (۸)	۰/۸۸ (۱۰)	۱ (۱)	۱ (۱)	۱ (۱)
F4	۱ (۱)	۱ (۱)	۱ (۱)	۱ (۱)	۱ (۱)	۱ (۱)	۱ (۱)	۱ (۱)	۱ (۱)	۱ (۱)	۱ (۱)	۱ (۱)
F5	۰/۸۸ (۵)	۰/۲۴ (۱۰)	۰/۹۶ (۴)	۰ (۱۱)	۰/۷۲ (۶)	۰/۷۲ (۶)	۰ (۱۱)	۰/۲۸ (۸)	۰/۲۸ (۸)	۱ (۱)	۱ (۱)	۱ (۱)
F6	۱ (۱)	۰/۵۲ (۱۰)	۰/۸۸ (۷)	۰ (۱۱)	۱ (۱)	۰/۹۶ (۶)	۰ (۱۱)	۰/۵۶ (۸)	۰/۵۶ (۸)	۱ (۱)	۱ (۱)	۱ (۱)
F7	۱ (۱)	۰/۷۶ (۷)	۰/۵۲ (۹)	۰ (۱۰)	۰ (۱۰)	۰ (۱۰)	۰/۹۲ (۵)	۰/۸۴ (۶)	۰/۶۰ (۸)	۱ (۱)	۱ (۱)	۱ (۱)
F8	۱ (۱)	۰/۹۲ (۳)	۰/۸۴ (۵)	۰/۷۲ (۷)	۰ (۱۱)	۰/۵۲ (۸)	۰ (۱۱)	۰/۰۴ (۹)	۰/۰۴ (۹)	۰/۸۴ (۵)	۰/۸۸ (۴)	۰/۹۶ (۱)
F9	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)	۰ (۱)
F10	۱ (۱)	۱ (۱)	۱ (۱)	۱ (۱)	۰ (۸)	۰ (۸)	۰ (۸)	۰ (۸)	۰ (۸)	۱ (۱)	۱ (۱)	۱ (۱)
F11	۱ (۱)	۰/۸۰ (۷)	۰/۸۰ (۷)	۰/۰۸ (۱۲)	۱ (۱)	۱ (۱)	۰/۶۴ (۱۱)	۰/۷۶ (۱۰)	۰/۸۰ (۷)	۱ (۱)	۱ (۱)	۱ (۱)
F12	۱ (۱)	۰/۷۲ (۸)	۰/۹۶ (۶)	۰ (۱۲)	۰/۹۶ (۶)	۱ (۱)	۰/۰۸ (۱۰)	۰/۵۶ (۱۱)	۰/۷۲ (۸)	۱ (۱)	۱ (۱)	۱ (۱)
F13	۰/۴۴ (۱۰)	۰/۹۶ (۷)	۱ (۱)	۱ (۱)	۱ (۱)	۰/۰۴ (۱۱)	۰ (۱۲)	۰/۹۲ (۹)	۰/۹۶ (۷)	۱ (۱)	۱ (۱)	۱ (۱)
F14	۱ (۱)	۰/۶۰ (۷)	۰/۸۸ (۵)	۰ (۹)	۰ (۹)	۰ (۹)	۰ (۹)	۰/۷۶ (۶)	۰/۱۲ (۸)	۱ (۱)	۱ (۱)	۱ (۱)
F15	۰/۷۲ (۵)	۰/۶۴ (۶)	۰/۷۶ (۴)	۰ (۷)	۰ (۷)	۰ (۷)	۰ (۷)	۰ (۷)	۰ (۷)	۱ (۱)	۱ (۱)	۱ (۱)
میانگین رتبه	۲/۵۳	۵/۲۶	۴/۱۳	۷/۷۳	۶/۲۶	۶/۹۳	۷/۶۶	۶/۷۳	۷/۰۶	۱/۴۶	۱/۲۰	**۱/۰۰**

جدول (۵): مقایسه الگوریتم پیشنهادی با الگوریتم LIPS توسط معیار متوسط تعداد بهینه‌های پیداشده

شماره تابع	ابعاد	LIPS				TNB-GSA					DNB-GSA				
		بهترین	بدترین	میانگین	انحراف معیار	بهترین	بدترین	میانگین	انحراف معیار		بهترین	بدترین	میانگین	انحراف معیار	
F16	۵	۰	۰	۰	۰	۰	۰	۰	۰		۴	۰	۲/۱۰	۰/۹۲	
	۱۰	۰	۰	۰	۰	۰	۰	۰	۰		۰	۰	۰	۰	
	۲۰	۰	۰	۰	۰	۰	۰	۰	۰		۰	۰	۰	۰	
F17	۲	۴	۳	۴	۰/۴۷	۴	۴	۴	۰		۶	۴	۴/۹۰	۰/۷۴	
	۵	۰	۰	۰	۰	۰	۰	۰	۰		۲	۰	۱/۰۱	۰/۶۸	
	۸	۰	۰	۰	۰	۰	۰	۰	۰		۱۱	۶	۸/۷۰	۱/۹۳	
F18	۲	۲۳	۱۹	۲۱/۲۰	۱/۲۳	۲۵	۲۱	۲۳/۵۰	۱/۴۱		۲۵	۲۲	۲۳/۸۰	۱/۳۵	
	۳	۳۰	۱۶	۲۲/۸۰	۴/۷۳	۶۹	۶۶	۶۵/۵۰	۲/۲۶		۶۹	۶۴	۶۶/۳۰	۲/۴۵	
	۴	۲۵	۱۹	۲۳/۱۰	۱/۶۶	۸۷	۷۹	۸۴/۴۰	۴/۷۲		۹۱	۸۲	۸۶/۱۰	۴/۸۰	
F19	۵	۰	۰	۰	۰	۲	۰	۰/۶۰	۰/۴۵		۲	۰	۰/۹۰	۰/۵۱	
	۱۰	۰	۰	۰	۰	۰	۰	۰	۰		۰	۰	۰	۰	
	۲۰	۰	۰	۰	۰	۰	۰	۰	۰		۰	۰	۰	۰	
F20	۲	۲۵	۲۰	۲۱/۷۰	۱/۷۰	۲۵	۲۱	۲۱/۹۰	۱/۳۲		۲۵	۲۳	۲۴/۱۰	۱/۰۴	
	۳	۳۱	۲۱	۲۵/۹۰	۲/۸۵	۷۵	۶۷	۷۱/۱۰	۵/۹۲		۷۸	۶۸	۷۲/۹۰	۵/۸۶	
	۴	۳۱	۱۶	۲۳/۸۰	۵/۴۹	۷۹	۶۳	۷۲/۳۰	۶/۴۲		۸۱	۶۲	۷۱/۵۰	۶/۷۳	
F21	۴	۲	۰	۰/۵۰	۰/۷۱	۹	۵	۷/۱۰	۱/۱۳		۱۵	۹	۱۱/۴۰	۱/۸۵	
	۶	۴۵	۳۸	۴۰/۱۰	۱/۹۷	۸	۲	۴/۷۰	۲/۰۸		۵۲	۴۳	۴۷/۲۰	۲/۲۳	
	۸	۷۰	۶۲	۶۵/۷۰	۲/۹۱	۸	۴	۶/۴۰	۲/۲۲		۹۸	۷۹	۸۸/۴۰	۴/۴۱	
F22	۶	۰	۰	۰	۰	۴	۱	۲/۴۰	۱/۲۴		۵	۰	۲/۶۰	۱/۴۸	
	۱۰	۱	۰	۰	۰/۳۲	۱۰	۲	۵/۷۰	۲/۷۳		۱۱	۳	۶/۰۰	۲/۹۵	
	۱۶	۱	۰	۰/۱۰	۰/۳۲	۵	۱	۲/۹۰	۱/۴۸		۳۲	۱۹	۲۵/۳۰	۳/۱۲	
F23	۲	۳۲	۲۶	۲۹/۱۰	۱/۹۱	۲۹	۲۴	۲۶/۱۰	۲/۰۳		۳۶	۳۰	۳۲/۹۰	۲/۲۶	
	۳	۸۱	۶۸	۷۴/۶۰	۴/۴۸	۴۵	۳۶	۴۱/۳۰	۳/۲۹		۹۴	۸۲	۸۸/۴۰	۴/۳۹	
	۴	۱۰۲	۸۰	۹۳/۲۰	۶/۰۵	۴۹	۴۰	۴۴/۸۰	۳/۷۸		۱۳۹	۱۲۰	۱۲۷/۹۰	۵/۸۳	

۵- نتیجه‌گیری و کارهای آینده

در این مقاله، برای اولین بار ساختار همسایگی نزدیکترین-بهتر برای الگوریتم‌های هوش جمعی تعریف و سپس در الگوریتم جستجوی گرانشی برای حل مسائل بهینه‌سازی چندمُدی استفاده شد. برای این منظور، ابتدا دو ساختار همسایگی "نزدیکترین-بهتر توپولوژیکی" و "نزدیکترین-بهتر مبتنی بر فاصله" تعریف شد، سپس این دو ساختار به طور مجزا در الگوریتم جستجوی گرانشی استفاده شدند و دو نسخه‌ی مختلف از الگوریتم جستجوی گرانشی به نام‌های TNB-GSA و DNB-GSA برای حل مسائل بهینه‌سازی چندمُدی ارائه شد. برای بررسی کارایی الگوریتم‌های پیشنهادی، یک ارزیابی تجربی روی چندین تابع محک چندمُدی استاندارد صورت گرفت که نتایج این آزمایشات نشان دادند که هر دو الگوریتم‌های پیشنهادی می‌توانند نتایج خوبی نسبت به سایر الگوریتم‌های بهینه‌ساز چندمُدی به دست آورند. اما در مقایسه بین دو الگوریتم پیشنهادی، مشخص شد که الگوریتم DNB-GSA در کل کارایی بهتری نسبت به الگوریتم TNB-GSA دارد.

برای تحقیقات آینده، الگوریتم‌های پیشنهادی می‌توانند بر روی مسائل بهینه‌سازی چندمُدی دنیای واقعی مانند انتخاب ویژگی، خوشه‌بندی داده‌ها، و غیره مورد بررسی و تحلیل قرار گیرند. همچنین، اثر بخشی الگوریتم‌های پیشنهادی در حل مسائل بهینه-سازی چندمُدی پویا می‌تواند مورد بررسی قرار گیرد. در نهایت، می-توان تعمیم‌های جدیدی از همسایگی نزدیکترین-بهتر برای به‌روز-رسانی موقعیت ذرات در فضای جستجو تعریف کرد به‌نحوی‌که سایر پارامترهای موثر برای تنظیم تنوع و همگرایی را نیز در خود داشته باشند.

مراجع

[1] J. Nocedal and S. Wright, *Numerical optimization*, Springer Science & Business Media, 2006.

[2] J.J. Liang, B.Y. Qu, P.N. Suganthan and Q. Chen, "Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization." Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 2014.

[3] B.Y. Qu, J.J. Liang, Z.Y. Wang, Q. Chen and P.N. Suganthan, "Novel benchmark functions for continuous multimodal optimization with comparative results," Swarm and Evolutionary Computation, vol. 26, pp. 23-34, 2016.

[4] E.G. Talbi, *Metaheuristics: from design to implementation*, John Wiley & Sons, 2009.

[5] K. Miettinen, P. Neittanmaki, M.M. Makela and J. Periaux, *Evolutionary algorithms in engineering and computer science*, John Wiley and Sons, Ltd, New York, 1999.

[6] J. Kennedy, R.C. Eberhart and Y. Shi, *Swarm intelligence*. Morgan Kaufmann, 2001.

[7] R.C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," In Proceedings of the Sixth International Symposium on Micro Machine and Human Science 1995 Oct 4 (pp. 39-43). IEEE.

[8] E. Rashedi, H. Nezamabadi-Pour and S. Saryazdi, "GSA: a gravitational search algorithm," Information sciences, vol. 179, no. 13, pp. 2232-2248.

[9] M. Dorigo, *Optimization, learning and natural algorithms*, Ph.D. Thesis, Politecnico di Milano, Italy, 1992.

[10] S. Yazdani, H. Nezamabadi-pour and S. Kamyab, "A gravitational search algorithm for multimodal optimization," Swarm and Evolutionary Computation, vol. 14, pp. 1-14, 2014.

[11] R. Brits, A.P. Engelbrecht and F. Van den Bergh, "A niching particle swarm optimizer," In Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning 2002 Nov 18 (Vol. 2, pp. 692-696). Singapore: Orchid Country Club.

[12] E. Özcan and M. Yılmaz, "Particle swarms for multimodal optimization," In International Conference on Adaptive and Natural Computing Algorithms 2007 Apr 11 (pp. 366-375). Springer, Berlin, Heidelberg.

[13] A. Passaro and A. Starita, "Particle swarm optimization for multimodal functions: a clustering approach," Journal of Artificial Evolution and Applications, vol. 2008, 2008.

[14] X. Li, "Niching without niching parameters: particle swarm optimization using a ring topology," IEEE Transactions on Evolutionary Computation, vol. 14(1), pp. 150-169, 2010.

[15] B.Y. Qu, P.N. Suganthan and S. Das, "A distance-based locally informed particle swarm model for multimodal optimization," IEEE Transactions on Evolutionary Computation, vol. 17(3), pp. 387-402, 2013.

[16] B.Y. Qu, J.J. Liang, Z.Y. Wang, Q. Chen and P.N. Suganthan, "Novel benchmark functions for continuous multimodal optimization with comparative results," Swarm and Evolutionary Computation, vol. 26, pp. 23-34, 2016.

[17] X. Li, "Adaptively choosing neighborhood bests using species in a particle swarm optimizer for multimodal function optimization," In Genetic and Evolutionary Computation Conference 2004 Jun 26 (pp. 105-116). Springer, Berlin, Heidelberg.

[18] X. Li, "A multimodal particle swarm optimizer based on fitness Euclidean-distance ration," In Proceedings of the 9th annual conference on Genetic and evolutionary computation 2007 Jul 7 (pp. 78-85). ACM.

[19] X. Li, "Efficient differential evolution using speciation for multimodal function optimization," In Proceedings of the 7th annual conference on Genetic and evolutionary computation 2005 Jun 25 (pp. 873-880). ACM.

[20] R. Thomsen, "Multimodal optimization using crowding-based differential evolution," In Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753) 2004 Jun 19 (Vol. 2, pp. 1382-1389). IEEE.

[21] W. J. Yu, J. Y. Ji, Y. J. Gong, Q. Yang, J. Zhang, "A tri-objective differential evolution approach for multimodal optimization," Information Sciences, vol. 423, pp. 1-23, 2018.

[22] J. Liang, W. Xu, C. Yue, K. Yu, H. Song, O. D. Crisalle, B. Qu, "Multimodal multiobjective optimization with differential evolution," Swarm and evolutionary computation, vol. 44, pp. 1028-1059, 2019.

[23] K. Thirugnanasambandam, S. Prakash, V. Subramanian, S. Pothula, V. Thirumal, "Reinforced cuckoo search algorithm-based multimodal optimization," Applied Intelligence, pp.1-25, 2019.

[1] Multimodal Optimization problems
[2] Exact algorithms
[3] Approximate algorithms
[4] Approximation algorithms
[5] Heuristic algorithms
[6] Specific heuristic algorithms
[7] Metaheuristics
[8] Population-based
[9] Single-solution based
[10] Evolutionary Algorithms (EAs)
[11] Stochastic
[12] Swarm Intelligence (SI) algorithms
[13] Particle
[14] Particle Swarm Optimization (PSO)
[15] Gravitational Search Algorithm (GSA)
[16] Ant Colony Optimization (ACO)
[17] Nearest-Better Neighborhood
[18] Topological Nearest-Better Neighborhood
[19] Distance-based Nearest-Better Neighborhood
[20] Niche
[21] Oscillation
[22] Topological Nearest-Better GSA (TNB-GSA)
[23] Distance-based Nearest-Better GSA (DNB-GSA)
[24] Decision Space
[25] Success rate