جهش جهت‌دار انطباقی برای کد واقعی الگوریتم ژنتیک

ترجمه: **ناصر باقری محمودآبادی**

چکیده:

عملگر جهش جهت‌دار انطباقی یک ایده‌ی جدید، ساده و کارآمد روی کد واقعی الگوریتم ژنتیک است و برای حل مسائل بهینه‌سازی با توابع پیچیده مورد استفاده قرار می‌گیرد. عملگر ADM برای بالابردن توانایی الگوریتم ژنتیک در جستجوی بهینه‌ی محلی و بالا بردن سرعت همگرایی با یکپارچه‌سازی استراتژی جستجوی محلی و استراتژی جستجوی تصادفی تطبیقی(توافقی) می‌باشد.

با استفاده از ۴۱ تابع تست بهینه‌سازی سراسری، عملکرد الگوریتم جدید را با پنج عملگر جهش معمولی مقایسه کرده و پس از آن با شش مورد از الگوریتم‌های ژنتیکی معروف که در گزارش آمده، مقایسه می‌شوند.

نتایج نشان می‌دهد که الگوریتم پیشنهادی ADM-RCGA سریع، دقیق و قابل اطمینان است و بهتر از همه الگوریتم‌های ژنتیک مطرح شده در این مقاله است.

۱- مقدمه:

بسیاری از برنامه‌های زندگی واقعی می‌تواند به مسائل بهینه‌سازی غیرخطی مدل شوند و در اغلب موارد به دنبال جواب بهینه‌ی سراسری مطلوب هستیم. یک نوع از مسأله بهینه‌سازی خطی به شرح می‌باشد.

$$\frac{Maximize}{Minimize}\ f(\boldsymbol{x} = x_1, x_1, \dots, x_1)\ \ subject\ to\ \boldsymbol{x} \in \Omega \qquad (1)$$

که x برداری با متغیرهای پیوسته در فضای جستجوی $\Omega \subseteq R^N$ و $f(x)$ یک تابع پیوسته حقیقی N متغیره می‌باشد و دامنه‌ی Ω درکران بالا و پایین هر یک از ابعاد تعریف شده است. برای پیدا کردن جواب بهینه‌ی سراسری x^* که متناظر با مقدار بهینه‌ی سراسری تابع $f(x^*)$ است، دو تکنیک برای حل مسائل بهینه‌سازی غیرخطی می‌توان یافت: یعنی بهینه‌سازی مبتنی برگرادیان[۱] و الگوریتم‌های بهینه‌سازی تکاملی[۲] [۱،۲]

همه بهینه‌سازی مبتنی برگرادیان (بهینه سازی قطعی نیز نامیده می‌شود) الگوریتم‌های نقطه به نقطه هستند و از این رو تکنیک‌های بهینه‌سازی محلی در طبیعت هستند. تکنیک‌های بهینه‌سازی مبتنی برگرادیان، مراحل جستجو را با یک جواب حدس اولیه شروع می‌کنند. اگر این جواب حدسی به اندازه کافی به جواب بهینه‌ی سراسری نزدیک نباشد، تکنیک‌های بهینه‌سازی مبتنی برگرادیان به احتمال زیاد در جواب بهینه محلی به دام می‌افتند. در عمل، هنگامی که برای بهینه‌سازی تلاش می‌کنیم به صورت خودکار، پیدا کردن یک جواب شروع مناسب، مشکل اصلی است.

بهینه‌سازی مبتنی برگرادیان با حدود ۲۰ متغیر معمولا غیرعملی است.[۲] زیرا هنگامی‌که تعداد متغیرها را افزایش می‌دهیم، به نوعی تعداد ارزیابی‌ها را افزایش داده‌ایم. بسیاری از آنها برای حل یک کلاس خاص از مسائل بهینه‌سازی با تعدادکمی متغیر طراحی شده‌اند.

به عبارت دیگر، تمام الگوریتم‌های بهینه‌سازی تکاملی با مجموعه‌های تصادفی از جواب‌های بالقوه که آنها براساس الگوریتم جستجو تصادفی و روش‌های بهینه‌سازی سراسری هستند، کار می‌کنند. الگوریتم بهینه‌سازی تکاملی به طور کلی در مقایسه با الگوریتم‌های بهینه‌سازی مبتنی برگرادیان در حل مسائل بهینه‌سازی با ابعاد بالا به خوبی قابل قیاس هستند.

الگوریتم‌های تکاملی از سه روش مبتنی بر جمعیت ابتکاری[۳] تشکیل شده است: الگوریتم‌های ژنتیکی، برنامه‌نویسی تکاملی و استراتژی-های تکاملی، که شاید الگوریتم‌های ژنتیک محبوب‌ترین الگوریتم تکاملی باشند. در پیاده‌سازی الگوریتم‌های ژنتیک متداول[۴،۵] متغیرهای تصمیم به عنوان رشته‌هایی دودویی کدگذاری شده هستند مانند الگوریتم ژنتیک دودویی[۴]

عملکرد الگوریتم ژنتیک دودویی در حل مسائل کوچک و متوسط با دقت کم، رضایت‌بخش بوده است. اما الگوریتم ژنتیک دودویی برای مسائل با ابعاد بالا که دنبال پاسخ با دقت زیادند، مستلزم زمان زیاد و حافظه [۶] می‌باشد.

برای بهبود این اشکالاتی در هنگام استفاده از الگوریتم ژنتیک دودویی در مسائل عددی با چند بعد و دقت بالا، متغیرهای تصمیم را می‌توان به عنوان اعداد حقیقی کدگذاری‌شده فرض کرد، بنام کد واقعی الگوریتم ژنتیک[۵]، که این تبدیل به طور فزآینده‌ای متداول است

---

[1] Gradient-Based Optimizers
[2] Evolutionary algorithm optimizers
[3] Heuristic
[4] Binary Coded Genetic Algorithm (BCGA)
[5] Real-Coded Genetic Algorithm (RCGA)

[۱،۷]. برتری کد واقعی الگوریتم ژنتیک (پیوسته) نسبت به الگوریتم ژنتیک دودویی برای مسائل بهینه‌سازی پیوسته [۸] و استخراج داده‌های پزشکی می‌باشد [۹].

عملکرد الگوریتم‌های ژنتیک متکی به عملگرهای جستجویی است که برای هدایت سیستم به سمت بهینه‌ی سراسری طراحی شده‌اند. یکی از مشکلات الگوریتم‌های ژنتیک همگرایی زودرس است. برای کاهش و یا حتی جلوگیری از به دام افتادن در بهینه محلی، عملگر جهش یک مکانیسم برای کشف جواب‌های جدید و حفظ تنوع جمعیت در جستجوی الگوریتم‌های ژنتیک فراهم می‌کند، اما این کار با کم‌کردن هزینه سرعت در فرآیند اطلاع[۱] همراه است.

به طور معمول در الگوریتم‌های ژنتیک، تلاش نسبتاً کمتری برای طراحی یک عملگر جهش جدید در کد واقعی الگوریتم ژنتیک شده است[۱]. اندازه گام و جهت جستجو عوامل اصلی در تعیین عملکرد عملگر جهش [۱۰] می‌باشد.

در بررسی کنونی به دنبال ارائه یک ایده‌ی جدید، ساده و کارآمد روی کد واقعی الگوریتم ژنتیک براساس عملگر جهش جهت‌دار انطباقی (ADM) که عملکرد آن را در مجموعه‌ای از مسائل بهینه‌سازی با توابع پیچیده نشان می‌دهد، هستیم.

ادامه این مقاله به شرح زیر است: بخش ۲ به بررسی مختصری از عملگر جهش درکد واقعی الگوریتم ژنتیک می‌پردازد. در بخش ۳ شرح مفصلی از روش پیشنهادی ارائه می‌کنیم. مجموعه‌ای از مسائل تست (محک)، مقایسه الگوریتم‌ها و نتایج آزمایشات در بخش ۴ گزارش شده است. در نهایت، بخش ۵ تعدادی از نتایج از بررسی کنونی ارائه شده است.

۲- نقد و بررسی عملگر جهش

به طورکلی، کد واقعی الگوریتم ژنتیک معمولی شامل سه عملگر اصلی انتخاب[۲]، برش[۳] و جهش[۴] به دنبال تکامل میزان شایستگی[۵] از جمعیت است که بعد از طی چندین نسل به سوی جواب بهینه‌ی سراسری همگرا می‌شود. این روش می‌تواند به عنوان یک فرآیند تکاملی مشاهده شود. عملیات جهش برای تغییر ژن فرزندان استفاده می‌شود. جهش یک عملگر کلیدی برای افزایش تنوع جمعیت است، از این رو الگوریتم‌های ژنتیک را به کشف مناطق امیدوارکننده[۶] از فضای جستجو [۱۰] قادر می‌سازد. برای جهش‌های متداول، می‌توان جهش تصادفی[۷]، جهش یکنواخت[۸]، جهش غیریکنواخت[۹]، جهش چند جمله‌ای[۱۰]، و جهش گوسی[۱۱] را نام برد.[۱،۱۱]

به تازگی تلاش‌های تحقیقاتی زیادی به منظور بهبود عملکرد الگوریتم‌های ژنتیک با استفاده از تکنیک‌های مختلف جهش انجام شده است. مانند زیر مفهوم جهش ایجاد شده در سیستم های بیولوژیکی، Bhandari و همکارانش [۱۲]. برای اولین بار روش جهش هدایت شده (جهت‌دار) را برای بهبود الگوریتم‌های ژنتیک دودویی مورد استفاده قرار دادند. براساس شیب و یا برون‌یابی، جهش هدایت‌شده قطعی با معرفی یک نقطه جدید در جمعیت که با اطلاعات به دست آمده در نسل‌های قبلی هدایت می‌شد.

Zhou و Li [۱۳] یک تکنیک تنوع (دگرگونی) هدایت‌شده برای عملگر جهش برای تنظیم برخی از موجودات با استفاده از اطلاعات بازخورد از جمعیت فعلی پیشنهاد داده‌اند.

Berry و Vamplew [۱۴] یک تکنیک co-evolutionary (تغییر یک شی بیولوژیکی موجب شده توسط تغییر از یک شیء مرتبط) که در آن به هر مولفه بردار جواب، یک بیت اضافی برای تعیین جهت جهش با استفاده از اطلاعات بازخورد از جمعیت فعلی اضافه شده است، پیشنهاد داده‌اند.

Temby و همکاران [۱۵] معرفی یک جهش هدایت شده براساس (مقدار) حرکت، که در آن هر مولفه از موجودات به یک جهش گوسی استاندارد و (مقدار) حرکت فعلی در جهش در هر جزء وابسته اند، پیشنهاد داده‌اند.

Korejo و همکاران [۱۰] یک عملگر جهش هدایت شده برای بهبود روش (دگرگونی) هدایت شده [۱۳] ، که در آن اطلاعات آماری در

---

[1] learning process
[2] selection
[3] crossover
[4] mutation
[5] fitness
[6] promising
[7] random mutation (RM)
[8] uniform mutation
[9] non-uniform mutation (NUM)
[10] polynomial mutation (PLM),
[11] Gaussian mutation

مورد میزان شایستگی و توزیع (پخش) موجودات با فواصل بالا که هر یک از ابعاد با توجه به جمعیت فعلی محاسبه شده و برای هدایت جهش، از یک موجود در فاصله‌ی همسایه آن که بهترین نتیجه آماری در هر بعد دارد، را پیشنهاد داده‌اند.

Srinivas و Patnaik [۱۶] یک الگوریتم ژنتیک دودویی تطبیقی برای بهینه‌سازی عملکرد توابع چندگانه تعریف کرده‌اند. در این الگوریتم ژنتیک تطبیقی، احتمال برش و جهش بسته به میزان شایستگی جواب‌ها، قابل تغییر و مرتبط هستند و جواب‌های با میزان شایستگی بالا محافظت‌شده درحالی‌که جواب‌های با میزان شایستگی متوسط کاملا مختل می‌شوند.

با توجه به اطلاعات از تحولات جمعیت در اثر تغییرات سطح میزان شایستگی، Chen و Liao [۱۷] عملگر جهش انطباقی با تنظیم مناسب سیاست‌های جستجو در کد واقعی الگوریتم‌های ژنتیک با استفاده از شبیه‌سازی گرادیان و یا ضد گرادیان جهت پیشنهاد داده‌اند.

Liao و Tseng [۱۸] دو استراتژی تطبیقی به منظور بهبود بهره‌وری تکاملی الگوریتم‌های ژنتیک پیشنهاد داده‌اند. یک استراتژی که به طور تصادفی می‌تواند عملگر برش رایج را جایگزین عملگر برش دیگر در هر زمان شود.

استراتژی دیگر این است که یک اصلاح تطبیقی از نرخ برش و جهش به منظور افزایش سطح تنوع ژنتیکی و راهنمای سیستم به سمت بهینه سراسری در صورتی که سیستم به دام بهینه محلی نیافتد، انجام می‌دهند.

روش state-of-the-art برای جهش تطبیقی، یک استراتژی تکاملی تطبیقی روی ماتریس کواریانس می‌باشد. (CMA-ES) [۱۹] CMA-ES بهتر از بسیاری الگوریتم‌های بهینه‌سازی پارامتری دیگر بوده و به طور مشهود در مسابقه الگوریتم CEC ۲۰۰۵، و توسط کارشناسان [۲۰] توصیه می شود.

Ling و Leung [۷] جهش موجک، که در قضیه موجک است، Deep و Thakur [۱] عملگر جهش قدرت (PM) را برای کد واقعی الگوریتم‌های ژنتیک براساس توزیع قدرت، پیشنهاد داده‌اند.

با استفاده از ایده‌ی بیولوژیکی و ریاضی که چارچوب کلی کد واقعی الگوریتم‌های ژنتیک است، Vafaee و Nelson [۲۱] یک روش جهش تطبیقی براساس فراوانی ژن بهترین کروموزوم، پیشنهاد داده‌اند.

۳- روش‌شناسی

۳-۱ ADM پیشنهاد شده :

هدف از این پژوهش معرفی یک عامل جهش جدید به نام ADM و ارزیابی عملکرد آن در برابر دیگر عملگرهای جهش موجود معرفی شده می‌باشد. عملگر ADM برای جلوگیری از همگرایی هر کروموزم ناشی از عمل برش و همچنین جستجوی غیر سیستماتیک یک سیستم ناشی از RM طراحی شده بود. عامل ADM یک جواب جدید را در جمعیت معرفی می‌کند. نقطه جستجو جدید که توسط جواب‌های به دست آمده از قبل که براساس جهت تطبیقی از گرادیان راهنمایی شده است، نام‌گذاری شده‌اند و جهت گرادیان از تکامل مقدار شایستگی هر موجود نشات گرفته شده است.

تعریف $\Delta f(t)$ و $\Delta f(t-1)$: تغییرات مقدار شایستگی برای هر کروموزوم x در سه نسل متوالی $(t-2, t-1, t)$

$$\Delta f(t) = f\big(\boldsymbol{x}(t)\big) - f\big(\mathbf{x}(\mathrm{t}-1)\big) \qquad (2)$$

$$\Delta f(t-1) = f\big(\boldsymbol{x}(t-1)\big) - f\big(\mathbf{x}(\mathrm{t}-2)\big) \qquad (3)$$

که $x = \{x_1, x_2, \dots, x_k, \dots, x_N\}$ یک کروموزوم است، $f(x(t))$ مقدار شایستگی کروموزوم x در نسل t است.

تعریف $\Delta x_k(t)$ و $\Delta x_k(t-1)$: تغییرات k بعدی ژن برای کروموزوم x در سه نسل متوالی $(t-2, t-1, t)$ بصورت زیر تعریف می‌شود :

$$\Delta x_k(t) = x_k(t) - x_k(t-1) \qquad (4)$$

$$\Delta x_k(t-1) = x_k(t-1) - x_k(t-2) \qquad (5)$$

از ترکیب فرمول‌های ۲ با ۵ ، جواب جدید برای x_k بدین شکل است :

$$x_k(t+1) = x_k(t) + (\Delta f(t), \Delta f(t-1), \Delta x_k(t), \Delta x_k(t-1), x_k(t), x_k^{UB}, x_k^{LB}).p_m \qquad (6)$$

که x_k^{UB} کران بالا و x_k^{LB} کران پایین برای x_k هستند و p_m احتمال جهش انطباقی می‌باشد. p_m باعث می‌شود کروموزوم بد دستخوش تغییرات قابل توجهی در جمعیت شود و می‌توان آن را به صورت زیر بیان کرد:

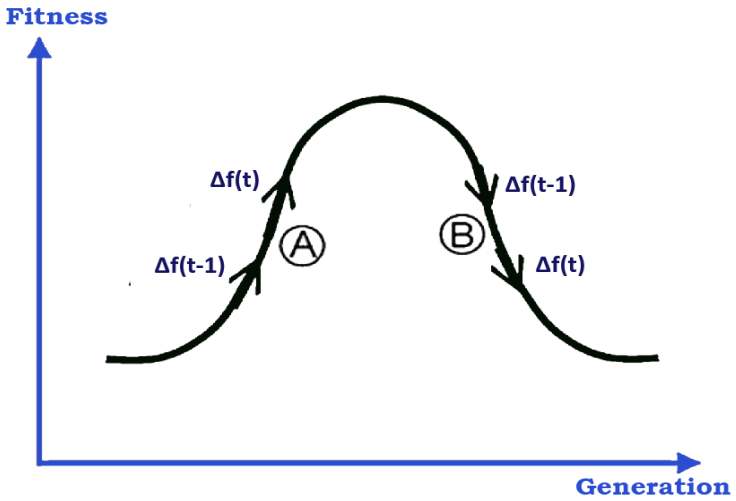
$$p_m = \begin{cases} 0.5 \times \frac{f_{max}(t) - f(x(t))}{f_{max}(t) - f(t)} & if\ f(x(t)) \geq \bar{f}(t) \\ 0.5 & if\ f(x(t)) < \bar{f}(t) \end{cases} \qquad (7)$$

که $f_{max}(t)$ بیشینه مقدار شایستگی جمعیت و $\bar{f}(t)$ میانگین مقدار شایستگی جمعیت می‌باشند. در معادله ۶ تابع $g(\cdot)$ می‌تواند به عنوان یک تابع سرعت‌دهنده باشد که جهش جهت‌دار را کنترل می‌کند. در این مقاله چهار استراتژی هدایت‌شده براساس ۹روند تکامل متفاوت برای هر کروموزوم بنا شده است. به نام‌های « جهش جهت‌دار مقیاس کوچک[1] »، « جهش تصادفی مقیاس کوچک[2] »، « جهش تصادفی مقیاس متوسط[3] »، « جهش تصادفی مقیاس بزرگ[4] ».

۹ روند تکامل متفاوت برای هر کروموزوم را می‌توان در سه حالت زیر دسته‌بندی کرد :

حالت ۱) $\Delta f(t-1).\Delta f(t) > 0$

طبق شکل ۱ هرکروموزوم در جمعیت همگرایی یا واگرایی میزان شایستگی خود را نسبت به بیشینه سراسری حفظ می‌کند. برای روند همگرایی مداوم نشان داده شده در مورد A ما برای سرعت بخشیدن به بهبود مقدار شایستگی، مقدار جهش را افزایش می‌دهیم. در مقابل مورد A، برای سرکوب سیر تکاملی واگرای سیستم در مورد B، جواب‌ها را در جهت مخالف حرکت می‌دهیم.



شکل ۱: روند تکامل میزان شایستگی در حالت $\Delta f(t-1).\Delta f(t) > 0$

[1] directional small-scale mutation
[2] random small-scale mutation
[3] random medium-scale mutation
[4] random large-scale mutation

جدول ۱ سه استراتژی ADM برای مورد $\Delta f(t-1).\Delta f(t)>0$ به طور خلاصه بیان می‌کند.

جدول ۱: استراتژی جهش برای حالت $\Delta f(t-1).\Delta f(t)>0$

استراتژی جهش	حالت ۱) $\Delta f(t-1).\Delta f(t)>0$
Directional small scale	$\Delta x_k(t-1).\Delta x_k(t)>0$
Random small scale	$\Delta x_k(t-1).\Delta x_k(t)<0$
Random medium scale	$\Delta x_k(t-1).\Delta x_k(t)=0$

برای مورد $\Delta x_k(t-1).\Delta x_k(t)>0$ جواب جدید $x_k\ (t+1)$ برای ژن k بعدی در کروموزوم x طراحی شده تا هم جهت یا در جهت مخالف جواب اصلی $x_k\ (t)$ امکان همگرایی سیستم را بهبود می‌بخشد. نقطه جدید $x_k\ (t+1)$ بدین صورت تشکیل می‌شود :

$$x_k(t+1)=x_k(t)+sign\big(\Delta f(t)\big).\Delta x_k(t).p_m \qquad (8)$$

$$sign(z)=\begin{cases}1 & \text{if } z>0\\ -1 & \text{if } z<0\\ 0 & \text{if } z=0\end{cases}$$

برای مورد $\Delta x_k(t-1).\Delta x_k(t)>0$ بدین معنی است که میزان شایستگی کروموزوم همگراست (مورد A) یا واگراست (مورد B) حتی اگر ژن‌های k بعدی در طول این سه نسل در جهت‌های مخالف تغییرکنند، ما مقدار کمی جهش را در جهت تصادفی استفاده کرده‌ایم. زیرا محاسبه تاثیرات بین ژن k بعدی و میزان شایستگی متناظر با آن سخت است. جواب جدید $x_k\ (t+1)$ را می‌توان این‌گونه تعریف کرد :

$$x_k(t+1)=x_k(t)+|\Delta x_k(t)|.r_s.p_m \qquad (9)$$

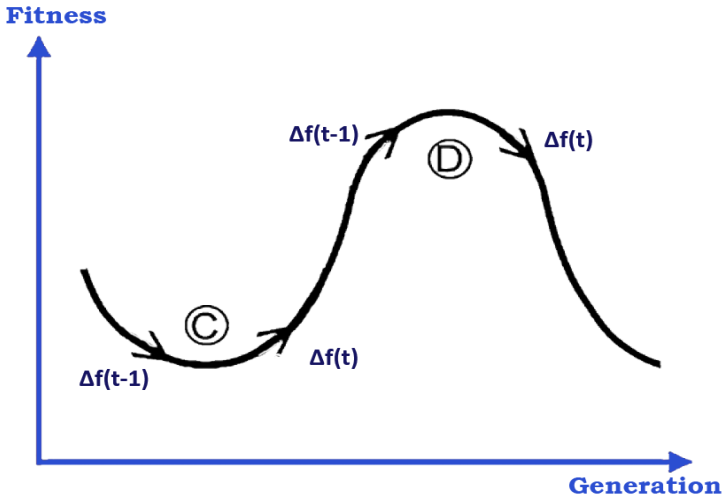
که r_s یک توزیع یکنواخت تصادفی بین ۱ و ۱- است .

به علاوه هنگامی‌که $\Delta x_k(t-1).\Delta x_k(t)=0$ تغییرات ژن k بعدی حداقل یکبار در سه نسل اخیر صفر می‌باشد. ما مقدار متوسط از جهش را برای کاهش و یا حتی جلوگیری از به دام انداختن بهینه‌ی محلی برای این جواب در جهت تصادفی به کار گرفته‌ایم. جواب جدید $x_k\ (t+1)$ را می‌توان این‌گونه تعریف کرد :

$$x_k(t+1)=x_k(t)+x_k(t).r_s.p_m \qquad (10)$$

حالت ۲) $\Delta f(t-1).\Delta f(t)<0$

شکل ۲ نشان می‌دهد که سیستم به احتمال زیاد به سمت همگرایی (مورد C) و به سمت واگرایی (مورد D) در حرکت است. هنگامی‌که $\Delta f(t-1).\Delta f(t)<0$ این بدین معنی است که جواب کروموزوم یا در روی قله و یا در قعر (نمودار) می‌باشد.



شکل ۲: روند تکامل میزان شایستگی در حالت $\Delta f(t-1).\Delta f(t)<0$

جدول ۲ سه استراتژی ADM برای مورد $\Delta f(t-1).\Delta f(t)<0$ به طور خلاصه بیان می‌کند.

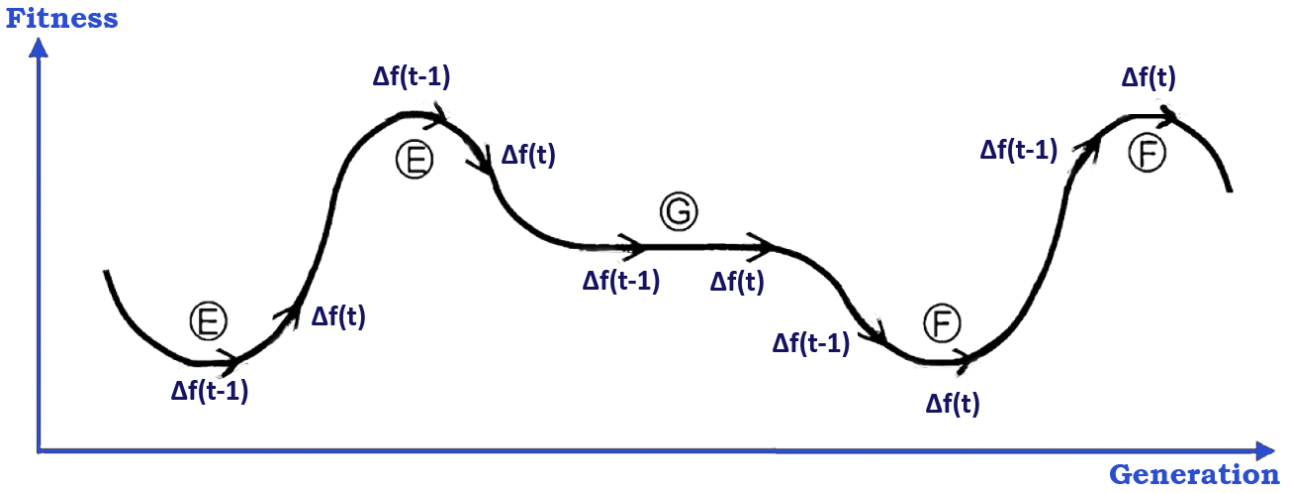
جدول ۲ : استراتژی جهش برای حالت $\Delta f(t-1).\Delta f(t)<0$

استراتژی جهش	حالت ۲) $\Delta f(t-1).\Delta f(t)<0$
Directional small scale	$f(x(t)) \geq \bar{f}(t)$
Random small scale	$f(x(t)) < \bar{f}(t)\ and\ \Delta x_k(t-1).\Delta x_k(t)<0$
Random medium scale	$f(x(t)) < \bar{f}(t)\ and\ \Delta x_k(t-1).\Delta x_k(t)=0$

برای مورد C ، $\Delta f(t-1)>0$ و $\Delta f(t)<0$ احتمال دارد سیستم همگرا شود. برای آزمایش این‌که آیا جهت جستجو قادر به بهبود مقدار شایستگی هست یا نه، مقدار کمی از جهت جهش برای کروموزوم‌هایی که از میانگین مقدار شایستگی جمعیت بهترند مورد استفاده قرار می‌گیرد. برای اعمال تنوع سیستم و بهبود مقدار شایستگی سیستم هنگامی‌که $f(x(t)) < \bar{f}(t)$، استراتژی جهش تصادفی مقیاس کوچک (معادله ۹) هنگامی‌که $\Delta x_k(t-1).\Delta x_k(t)<0$ و جهش تصادفی مقیاس متوسط (معادله ۱۰) هنگامی‌که $\Delta x_k(t-1).\Delta x_k(t)=0$ اعمال می‌شود.

حالت ۳) $\Delta f(t-1).\Delta f(t)=0$

شکل ۳ سه روند تکاملی را نشان می‌دهد (موارد E و F و G)



شکل ۳: روند تکامل میزان شایستگی در حالت $\Delta f(t-1).\Delta f(t)=0$

برای حالت $\Delta f(t-1).\Delta f(t)=0$ استراتژی های متناظر ADM در جدول ۳ نمایش داده شده‌اند.

جدول ۳: استراتژی جهش برای حالت $\Delta f(t-1).\Delta f(t)=0$

استراتژی جهش	حالت ۳) $\Delta f(t-1).\Delta f(t)=0$
Directional small scale	$f(x(t)) \geq \bar{f}(t)$
Random small scale	$f(x(t)) < \bar{f}(t)\ and\ \Delta x_k(t-1).\Delta x_k(t)<0$
Random Large scale	$f(x(t)) < \bar{f}(t)\ and\ \Delta x_k(t-1).\Delta x_k(t)=0$

برای مورد G، $\Delta f(t-1)=0$ و $\Delta f(t)=0$ هیچ تغییری در میزان شایستگی رخ نمی‌دهد که نتیجه آن تنوع بسیارکم سیستم است. ما مقدار زیادی جهش را برای کمک به موجود اصلی برای فرار از این منطقه هموار و جلوگیری از به دام انداختن در بهینه‌ی محلی انجام می‌دهیم. جهش مقیاس بزرگ به صورت زیر تعریف می‌شود :

$$x_k(t+1) = \begin{cases} x_k(t) + \left(x_k^{UB} - x_k(t) \right).r_s.p_m & if\ \ r < 0.5 \\ x_k(t) + (x_k(t) - x_k^{LB}).r_s.p_m & if\ \ r \geq 0.5 \end{cases} \quad (11)$$

که r یک عدد تصادفی با توزیع یکنواخت بین ۰ و ۱ است .

برای موردE، $\Delta f(t-1) = 0$ و $\Delta f(t) \neq 0$ از استراتژی جهش جهت‌دار مقیاس کوچک (معادله ۸) برای بهبود همگرایی سیستم استفاده می‌شود. در مقابل استراتژی جهش تصادفی مقیاس کوچک (معادله ۹) برای تنوع سیستم در مورد $\Delta f(t-1) \neq 0$ و $\Delta f(t) = 0$ (موردF) به کار می‌رود. (معادله ۱۰)

۳-۲ ADM-RCGA پیشنهادی:

در این قسمت روند تکامل کد واقعی ژنتیک بر پایه عامل جهش پیشنهادی ADM برای بهبود تابع توضیح داده می‌شود. الگوریتم ژنتیک به طور ساده یک تکنیک جستجوی تکرارشونده بر پایه جمعیت است که روی مفهوم احتمال کار می‌کند. عوامل ژنتیکی از جمله انتخاب، برش و جهش، هسته‌های اصلی الگوریتم‌های ژنتیک هستند. کار اصلی آنها تولید ترکیبات جدید براساس پارامترها در راستای تابع شایستگی (ترکیب پارامترها) برای دستیابی به هدف تکامل می‌باشد.

اول، یک جمعیت اولیه از کروموزوم‌ها، که در آن هر ژن توسط یک عدد حقیقی p مشخص شده که به طور تصادفی از کران‌های پایینی و بالایی از هر متغیر تصمیم ایجاد شده است.

دوم، هرکروموزوم توسط یک تابع شایستگی تعریف می‌شود؛ در این فرآیند، میزان تابع شایستگی بالاتر نشان‌دهنده کروموزوم بهتر است.

سوم برای بهبود عملکرد همگرایی و جلوگیری از کاهش سطح تنوع ژنتیکی در جمعیت در طول روند تکامل یک روش مقیاس‌بندی[1] مورد نیاز است و به موجب آن به ویژه رشته‌های خوب را می‌توان در مراحل اولیه از جمعیت غربال کرد در حالی که مقداری از فشار انتخاب هنوز هم در مراحل نهایی حفظ می‌شود. [۵] در این مقاله یک مکانیزم مقیاس‌بندی خطی اعمال شده است.

چهارم، برخی از کروموزوم انتخاب می‌شوند تا تحت عمل ژنتیکی برای انتخاب روش نمونه‌گیری تصادفی جهانی[2] قرار گیرند[۲۲]. انتخاب یک پروسه که فشار بر روی یک جمعیت در شیوه‌ای مشابه انتخاب طبیعی اعمال شده در سیستم‌های بیولوژیکی است. کروموزوم‌های با میزان شایستگی بالا برای تولیدمثل انتخاب می‌شوند درحالی که کروموزوم‌های ضعیف ممکن است اصلا انتخاب نشوند. پنجم، عملیات ژنتیکی برش انجام می‌شود. برش اجازه می‌دهد تا اطلاعات در یک روش مشابه ارگانیسم طبیعی که تحت تکثیر جنسی انجام می‌شود رد و بدل شوند. برش توسط جابجایی بخش مربوطه یک نماینده ژنتیکی از پدر و مادر انجام می‌شود و جستجو برای جواب‌های جدید در جهات دور از دسترس را گسترش می‌دهد. عملگر برش تنها با چند احتمال رخ می‌دهند که به آنها نرخ برش (p_c) می‌گویند. یک برش ترکیبی متغیر $(BLX-\alpha)$ [۲۳] گنجانیده شده در پوشش برش در این مقاله به اجرا درآمد است.

ششم، عملیات جهش بعد از عمل برش انجام می‌شود. جهش برای انتخاب تصادفی یک عضو از جمعیت و تغییر یک جنبه تصادفی در رشته نمایش آن استفاده می‌شود.

با اینکه انتخاب و جهش رشته‌های جدید را بوجود می‌آورند، اما هیچ اطلاعات جدیدی را در سطح ژن به جمعیت معرفی نمی‌کنند. روند جهش اطمینان حاصل می‌کند که احتمال دسترسی هر نقطه در فضای جستجو هیچ‌وقت صفر نباشد. جهش به همراه یک میزان از احتمال رخ می‌دهد که به آن نرخ جهش می‌گویند (p_m). در این مقاله ما یک عامل جهش به نام ADM را پیشنهاد کرده‌ایم.

هفتم یک استراتژی نخبه‌گرایی بعد از عملگر جهش اعمال می‌شود. یک استراتژی نخبه‌گرایی امکان دارد سرعت را افزایش دهد تا یک موجود فوق العاده بر جمعیت غالب شود.

برای بسیاری از کاربردها سرعت جستجو می‌تواند با از دست ندادن بهترین یا نخبه‌ترین عضو بین نسل‌ها بسیار افزایش یابد[۵]. این مقاله یک مکانیزم چند نخبگی براساس احتمال (p_e) را ارائه می‌کند. روندی که نه تنها بهترین موجودات را در جمعیت نگه می‌دارد، بلکه به بقیه موجودات نیز اجازه بقاء می‌دهد.

بعد از اینکه روند انتخاب، برش، جهش و نخبه‌گرایی روی جمعیت اولیه اعمال شدند یک جمعیت جدید در مرحله جابجایی شکل می‌گیرد. بعد جابجایی جمعیت جدید براساس شایستگی، تکامل پیدا می‌کند. این روند انتخاب، برش، جهش، نخبه‌گرایی و جابجایی ادامه

---

[1] scaling method

[2] stochastic universal sampling (SUS)

پیدا می‌کند تا اینکه به یک مقدار ثابت نسل یا اینکه به یک معیار همگرایی برسد.

شکل ۴ چرخه‌ی کاری ADM-RCGA پیشنهادی را نشان می‌دهد. در این مقاله این روند تکامل با استفاده از یک چارچوب برنامه‌نویسی Microsoft.NET framework نوشته شده‌اند.

```
Procedure ADM − RCGA
{
    t = 0;
    Randomly generate initial population P(t);
    repeat {
        Evaluate P(t) to obtain its fitness;
        Scale the fitness values of P(t);
        while (not done)
            {
                Select two parents p1 and p2 from P(t) based on their fitness;
                Perform crossover for p1, p2 to produce c1 and c2 based on probability pc;
                Mutate c1 or c2 using ADM based on probability pm;
                Put c1 and c2 into P(t + 1);
            }
        Put the elite members of P(t) into P(t + 1) based on probability pe;
        Generate the new generation P(t + 1) to replace P(t);
        t = t + 1;
    } until ( termination is met)
}
```

شکل ۴: چرخه‌ی کار الگوریتم ADM-RCGA

جزئیات عملگرهای باقیمانده در زیر آمده است.

۳-۳ بقیه‌ی عملگرهای مورد استفاده در این مقاله

در این قسمت ما انتخاب، برش و دیگر عوامل جهش را که در این مقاله مورد استفاده قرار داده‌ایم را توضیح می‌دهیم. به طور مثال SUS, BLX و MNUM, RM, PLM, NUM و PM

۳-۳-۱ نمونه برداری جهانی تصادفی:

SUS یک نام‌گذاری استاندارد برای انتخاب متغیر چرخ رولت است. به جای استفاده از اشاره‌گر در چرخ رولت، SUS از یک عدد تصادفی برای بدست آوردن نقطه‌ی شروع، استفاده می‌کند. سپس روند انتخاب تمام چرخ را با قدم‌های یکسان می‌پیماییم که اندازه‌ی هر قدم توسط تعداد موجودات که باید انتخاب شوند محاسبه می‌شود.

SUS اطمینان می‌دهد که تکرار موجودات انتخاب شده با تکرارهای مورد نظر برای دستیابی به پخش‌شدگی کمتر (حداقل گسترش) در تطابق است. که چرخ رولت اصلی این امکان را ضمانت نمی‌کند. به علاوه هر موجود می‌تواند براساس مکانش در جمعیت انتخاب شود. SUS به هیچ وجه تبعیض قائل نمی‌شود. بنابراین دلیل SUS یکی از پراستفاده‌ترین الگوریتم‌های انتخاب در ژنتیک می‌باشد[۲۴].

۳-۳-۲ برش $BLX-\alpha$ با پوشش متقاطع :

برش $BLX-\alpha$ [۲۳] یک ترکیب از انتخاب دو والد p_1 و p_2 می‌باشد.

این عامل جواب فرزندان را در رنج $(\min(p_1, p_2) - \alpha|p_1 - p_2|, \max(p_1, p_2) + \alpha|p_1 - p_2|)$ قرار می‌دهد که مقدار α باید طوری انتخاب شود تا جواب فرزندان از رنج مورد نظر خارج نشود. در این مقاله مقدار $\alpha = 0.25$ انتخاب شده است.

فرزندان c_1 و c_2 به صورت زیر تعریف می‌شوند.

$$\begin{cases} c_1 = \max(\boldsymbol{p}_1, \boldsymbol{p}_2) + \alpha|\boldsymbol{p}_1 - \boldsymbol{p}_2|.r_s \\ c_2 = \min(\boldsymbol{p}_1, \boldsymbol{p}_2) + \alpha|\boldsymbol{p}_1 - \boldsymbol{p}_2|.r_s \end{cases} \quad (12)$$

در حالت دودویی برای انتخاب اینکه کدام متغیر از دو موجود انتخاب و جفت شوند ما ابتدا پوشش متقاطع[1] را قبل از $BLX-\alpha$ اعمال می‌کنیم. در رشته‌هایی دودویی پوشش متقاطع و رشته‌ی جواب هم‌اندازه به صورت تصادفی تشکیل می‌شود. اگر پوشش متقاطع در یک بیت خاص ۱ داشته باشد. آنگاه مقدار فرزندان در آن براساس $BLX-\alpha$ از دو والد انتخاب و محاسبه خواهند شد.

۳-۳-۳ جهش تصادفی (RM):

RM نیز جهش یکنواخت نامیده می‌شود، جواب جهش‌یافته از جواب‌های اصلی با استفاده از قاعده‌ی زیر بدست می‌آید [۱۱].

$$x_k(t+1) = x_k(t) + \Delta(r-0.5) \qquad (13)$$

که r عدد تصادفی بین ۰ و ۱ است و Δ ماکسیمم مقدار انحراف تعریف شده توسط کاربر است. در این مقاله Δ به صورت زیر تعریف می‌شود:

$$\Delta = \max[2(x_k(t) - x_k^{LB}), 2(x_k^{UB} - x_k(t))] \qquad (14)$$

۳-۳-۴ جهش غیر یکنواخت (NUM):

NUM یک عامل با ظرفیت انتخاب دقیق می‌باشد. رفتار آن به تعداد زایش جمعیت بستگی دارد. از یک نقطه‌ی اصلی $x_k(t)$ ، نقطه جهش‌یافته $x_k(t+1)$ بصورت زیر بوجود می آید:

$$x_k(t+1) = \begin{cases} x_k(t) + h(t, x_k^{UB} - x_k(t)) & if\ r < 0.5 \\ x_k(t) + h(t, x_k(t) - x_k^{LB}) & if\ r \geq 0.5 \end{cases} \qquad (15)$$

تابع h داده شده در زیر مقادیر دربازه [0 , y] می‌گیرد.

$$h(t,y) = y(1 - r^{(1-(t/t_{max}))^b}) \qquad (16)$$

که t_{max} حداکثر تعداد نسل در جمعیت و b یک پارامتر سیستم است که قدرت جهش عامل را نشان می‌دهد. در نسل‌های اولیه NUM تمایل به جستجوی یکنواخت فضا دارد، در نسل‌های بعدی تمایل به جستجوی فضا به صورت محلی پیدا می‌کند[۱].

۳-۳-۵ جهش غیر یکنواخت چندگانه (MNUM):

یک عملگر جهش که تنوع ژنتیکی یک موجود نامزد را افزایش می‌دهد. در این مقاله MNUM [۲۶] به کار گرفته شده زیرا ما را قادر می‌سازد تا جستجوی یکنواخت به همراه تنظیم دقیق محل را اجرا کنیم و قابلیت بهره‌برداری از یک فضای جستجو را افزایش دهیم. عملگر به صورت زیر تعریف می‌شود.

$$x_k(t+1) = \begin{cases} x_k(t) + h(t, x_k^{UB} - x_k(t)) \cdot A(t) & if\ r < 0.5 \\ x_k(t) + h(t, x_k(t) - x_k^{LB}) \cdot A(t) & if\ r \geq 0.5 \end{cases} \qquad (17)$$

$$A(t) = \left[r_1\left(1 - \frac{t}{t_{max}}\right)\right]^b \qquad (18)$$

هر دو r و r_1 اعداد تصادفی یکنواخت بین ۰ و۱ هستند و b پارامتر شکل[2] می‌باشد. در این مقاله پارامتر b در معادلات ۱۶ و ۱۸ تنظیم می‌شود.

[1] Crossover Mask
[2] shape

۳-۳-۶ جهش چند جمله‌ای (PLM):

Deb و Goyal [۲۷] یک عملگر جهش بر پایه توزیع چندجمله‌ای ارائه می‌کند. جواب جهش یافته از جواب اصلی به صورت زیر بدست می‌آید:

$$x_k(t+1) = x_k(t) + \bar{\delta}.\delta_{max} \tag{19}$$

$$\bar{\delta} = \begin{cases} (2\mathrm{r})^{(\frac{1}{q}+1)} - 1 & if\ \ r < 0.5 \\ 1 - [2(1-\mathrm{r})]^{(\frac{1}{q}+1)} & if\ \ r \geq 0.5 \end{cases} \tag{20}$$

که q یک عدد مثبت، r یک عدد تصادفی با توزیع یکنواخت بین ۰ و۱، δ_{max} مقدار ماکسیمم اختلال تعریف شده مجاز بین جواب اصلی و جواب جهش‌یافته که توسط کاربر تعیین می‌شود در این مقاله $q = 2$ قرار داده می‌شود و δ_{max} به صورت زیر طراحی می‌شود:

$$\delta_{max} = \max[x_k(t) - x_k^{LB}, x_k^{UB} - x_k(t)] \tag{21}$$

۳-۳-۷ جهش قدرت (PM):

Deep و Thakur [۱] یک عملگر جهش بر پایه توزیع قدرت پیشنهاد می‌کنند. PM برای درست کردن جواب جدید جهش‌یافته $x_k(\mathrm{t}+1)$ در همسایگی جواب $x_k(\mathrm{t})$ به کارگرفته می‌شود به صورت زیر تعریف می‌شود.

$$x_k(t+1) = \begin{cases} x_k(t) - s\,.\,(x_k(t) - x_k^{LB}) & if\ \ u < \mathrm{r} \\ x_k(t) - s\,.\,\left(x_k^{UB} - x_k(t)\right) & if\ \ u \geq \mathrm{r} \end{cases} \tag{22}$$

که $u = (x_k(\mathrm{t}) - x_k^{LB})/(x_k^{UB} - x_k^{LB})$ و r یک عدد تصادفی با توزیع یکنواخت بین ۰ و ۱ است و عدد تصادفی s بر پایه توزیع قدرت به صورت زیر بدست می‌آید:

$$s = p\,.\,s_r^{p-1} \quad 0 \leq s_r \leq 1 \tag{23}$$

که p شاخص توزیع قدرت است و s_r یک عدد تصادفی با توزیع یکنواخت بین ۰ و ۱ است. قدرت جهش توسط شاخص p کنترل می‌شود. برای مقادیر بزرگ p تنوع بیشتری مورد انتظار است. برای مقادیر کوچک p اختلال کمتری در جواب بدست می‌آید. در این مقاله $p = 0.5$ در نظر گرفته می‌شود.

۴- آزمایش‌ها و نتایج

هدف مطالعه کنونی معرفی یک عملگر جهش جدید AMD و ارزیابی عملکرد آن در مقابل ۵ عملگر جهش متداول و ۶ الگوریتم ژنتیک متداول است. بنابراین در این کار چهار آزمایش اجرا شده است. در جدول ۴ خلاصه‌ی شرایط شبیه‌سازی به کار گرفته شده در این۴ آزمایش آمده است. تمام آزمایش‌ها بر روی یک ماشین Intel Xeon X5570 2.93 GHz با RAM 6GB تحت پلتفرم WINXP انجام می‌شود.

جدول ۴: خلاصه‌ای از شرایط شبیه‌سازی

آزمایش	نام الگوریتم ژنتیک	توابع محک	بعد	اندازه‌ی جمعیت	حداکثر تعداد نسل	تعداد اجرا
Exp.1	ADM-RCGA	توابع محک I $f_1 - f_{20}$	30	300	30.000	30
	RM-RCGA					
	PLM-RCGA					
	NUM-RCGA					
	MNUM-RCGA					
	PM-RCGA					
Exp.2	ADM-RCGA	توابع محکII $g_1 - g_9$	2,10,20,30	100	40000	100
	CMA-ES [19]					
	HYK-GA [21]					
Exp.3	ADM-RCGA	توابع محک I $f_1 - f_{20}$	30	300	5000	30
	LX-PM [1]					
Exp.4	ADM-RCGA	توابع محک III $h_1 - h_{12}$	10,100	30	12000	30
	IEA [28]					
	BOA [29]					
	OGA [30]					

۴-۱ توابع محک

به منظور بررسی عملکرد الگوریتم ADM-RCGA پیشنهادی، ۴۱ تابع محک شناخته شده و ارزش واقعی در این ۴ آزمایش با مقایسه ۵ عملگر جهش متداول و ۶ الگوریتم ژنتیک موجود به کارگرفته شده‌اند. این مسائل تست بهینه‌سازی سراسری، سطوح پیچیدگی و چندنمایی[1] مختلفی را مانند توابع پیوسته و ناپیوسته و همچنین توابع چندنمایی[2] و یک نمایی[3] را دربردارند. آنها همان‌طور که در جدول ۴ نشان داده شده است در سه گروه برای چهار آزمایش در مطالعه کنونی تقسیم می‌شوند:

گروه f_1 تا f_{20} [۱] به‌عنوان توابع محک I، g_1 تا g_9 [۱۹،۲۱] به‌عنوان توابع محک II و گروه h_1 تا h_{12} [۲۸،۲۹،۳۰] به‌عنوان توابع محک III.

تابع تست متناظر، دامنه‌ی پارامتر، و بهینه‌ی سراسری برای هر تابع در جدول ۵ ، جدول ۶ و جدول ۷ به ترتیب برای توابع محک I تا III تصحیح و فهرست شده است.

[1] multimodality
[2] multimodal functions
[3] Unimodal functions

جدول ۵: توابع محک I [۱]

	توابع محک	دامنه‌ی x_i	بهینه
f_1	$f_1 = -20\exp(-0.02\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}) - \exp(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i)) + 20 + e$	$[-30, 30]$	0 (min)
f_2	$f_2 = 0.1\sum_{i=1}^{N}\cos(5\pi x_i) - \sum_{i=1}^{N} x_i^2$	$[-1, 1]$	0.1N (max)
f_3	$f_3 = \exp(-0.5\sum_{i=1}^{N} x_i^2)$	$[-1, 1]$	1 (max)
f_4	$f_4 = 1 + \frac{1}{4000}\sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N}\cos(\frac{x_i}{\sqrt{i}})$	$[-600, 600]$	0 (min)
f_5	$f_5 = \frac{\pi}{N}\left(10sin^2(\pi y_1) + \sum_{i=1}^{N-1}(y_1 - 1)^2[1 + 10sin^2(\pi y_{i+1})] + (y_i - 1)^2\right)$ $where\ \ y_i = 1 + \frac{1}{4}(x_i + 1)$	$[-10, 10]$	0 (min)
f_6	$f_6 = 0.1\left(sin^2(3\pi x_1) + \sum_{i=1}^{N-1}(x_i - 1)^2[1 + sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + sin^2(2\pi x_n)]\right)$	$[-5, 5]$	0 (min)
f_7	$f_7 = (\prod_{i=1}^{N} x_i)^{0.2} - \sum_{i=1}^{N}[(\ln(x_i - 2))^2 + (\ln(10 - x_i))^2]$	$[2, 10]$	=997867.469 (max)
f_8	$f_8 = 10N + \sum_{i=1}^{N}[x_i^2 - 10\cos(2\pi x_i)]$	$[-5.12, 5.12]$	0 (min)
f_9	$f_9 = \sum_{i=1}^{N-1}[100(x_{i+1} - x_i^2)^2 - (x_i - 1)^2]$	$[-30, 30]$	0 (min)
f_{10}	$f_{10} = \sum_{i=1}^{N} x_i \sin(\sqrt{\lvert x_i\rvert})$	$[-500, 500]$	=12569.487 (max)
f_{11}	$f_{11} = [2.5\prod_{i=1}^{N}\sin\left(x_i - \frac{\pi}{6}\right) + \prod_{i=1}^{N}\sin(5(x_i - \frac{\pi}{6}))]$	$[0, \pi]$	3.5 (max)
f_{12}	$f_{12} = \sum_{i=1}^{N} x_i^2 + (\sum_{i=1}^{N}\frac{i}{2}x_i)^2 + (\sum_{i=1}^{N}\frac{i}{2}x_i)^4$	$[-5.12, 5.12]$	0 (min)
f_{13}	$f_{13} = \sum_{i=1}^{N} x_i^2$	$[-5.12, 5.12]$	0 (min)
f_{14}	$f_{14} = \sum_{i=1}^{N} i x_i^2$	$[-5.12, 5.12]$	0 (min)
f_{15}	$f_{15} = \sum_{i=1}^{N}\lvert x_i\rvert + \prod_{i=1}^{N}\lvert x_i\rvert$	$[-10, 10]$	0 (min)
f_{16}	$f_{16} = \max_i\{\lvert x_i\rvert, 1 \le i \le N\}$	$[-100, 100]$	0 (min)
f_{17}	$f_{17} = \sum_{i=1}^{N}(x_i^4 + rand(0,1))$	$[-10, 10]$	0 (min)
f_{18}	$f_{18} = \sum_{i=1}^{N}(x_i - i)^2$	$[-N, N]$	0 (min)
f_{19}	$f_{19} = \frac{\pi}{N}\left(10sin^2(\pi y_1) + \sum_{i=1}^{N-1}(y_1 - 1)^2[1 + 10sin^2(\pi y_{i+1})] + (y_i - 1)^2\right) + \sum_{i=1}^{N} u(x_i, 10,100,4)$ $where\ \ y_i = 1 + \frac{1}{4}(x_i + 1)$	$[-50, 50]$	0 (min)
f_{20}	$f_{20} = 0.1\left(sin^2(3\pi x_1) + \sum_{i=1}^{N-1}(x_i - 1)^2[1 + sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + sin^2(2\pi x_n)]\right)$ $+ \sum_{i=1}^{N} u(x_i, 10,100,4)$	$[-50, 50]$	0 (min)
$u(x, a, k, m) = \begin{cases} k \times pow((x - a), m) & if\ x > a \\ -k \times pow((x - a), m) & if\ x < -a \\ 0 & otherwise \end{cases}$		در توابع ۱۹ و ۲۰ مقدار از فرمول روبرو محاسبه می‌شود	

جدول ۶: توابع محک II [۱۹،۲۱]

	توابع محک	دامنه‌ی x_i	بهینه
g_1	$g_1 = \sum_{i=1}^{N} x_i^2$	$[-100, 1000]$	0 (min)
g_2	$g_2 = \sum_{i=1}^{N} \lvert x_i \rvert + \prod_{i=1}^{N} \lvert x_i \rvert$	$[-10, 10]$	0 (min)
g_3	$g_3 = \sum_{i=1}^{N-1} [100(x_{i+1} - x_i^2)^2 - (x_i - 1)^2]$	$[-29, 31]$	0 (min)
g_4	$g_4 = \sum_{i=1}^{N} i x_i^4 + random[0,1]$	$[-1.28, 1.25]$	0 (min)
g_5	$g_5 = \frac{1}{4000} \sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-100, 100]$	0 (min)
g_6	$g_6 = \sum_{i=1}^{N} [x_i^2 - 10\cos(2\pi x_i + 10)]$	$[-5.12, 5.12]$	0 (min)
g_7	$g_7 = -20\exp(-0.02\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}) - \exp(\sum_{i=1}^{N} \frac{\cos(2\pi x_i)}{N})$	$[-5.12, 5.12]$	0 (min)
g_8	$g_8 = -4x_1^2 + 2.1x_1^4 - \frac{1}{3}x_1^6 - x_1x_2 + 4x_2^2 - 4x_2^4$	$x_1 \in [-4.91017, 5.0893]$ $x_2 \in [-5.71260, 4.2874]$	1.031 (max)
g_9	$g_9 = [\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}]^{-1}$ $a_{ij} = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \dots & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \dots & 32 & 32 \end{bmatrix}$	$[-98, 34]$	0.998 (min)

جدول ۷: توابع محک III [۲۸،۲۹،۳۰]

	توابع محک	دامنه‌ی x_i	بهینه
h_1	$h_1 = \sum_{i=1}^{N} \lvert \frac{\sin(10\pi x_i)}{10\pi x_i} \rvert$	$[-0.5, 0.5]$	0 (min)
h_2	$h_2 = \sum_{i=1}^{N-1} [\sin(x_i + x_{i+1}) + \sin(\frac{2x_i x_{i+1}}{3})]$	$[3, 13]$	2N (max)
h_3	$h_3 = \sum_{i=1}^{N} (x_i - 0.5)^2$	$[-100, 100]$	0 (min)
h_4	$h_4 = \sum_{i=1}^{N} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]$	0 (min)
h_5	$h_5 = \sum_{i=1}^{N} x_i^2$	$[-5.12, 5.12]$	0 (min)
h_6	$h_6 = \sum_{i=1}^{N} [x_i \sin(10\pi x_i)]$	$[-1, 2]$	1.85N (max)
h_7	$h_7 = -\sum_{i=1}^{N} [\sin(x_i) + \sin(\frac{2x_i}{3})]$	$[3, 13]$	1.21598N(max)
h_8	$h_8 = 20\exp(-0.02\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}) - \exp(\sum_{i=1}^{N} \frac{\cos(2\pi x_i)}{N}) - 20 - e$	$[-30, 30]$	0 (max)
h_9	$h_9 = 418.9828N - \sum_{i=1}^{N} x_i \sin(\sqrt{\lvert x_i \rvert})$	$[-500, 500]$	0 (min)
h_{10}	$h_{10} = \sum_{i=1}^{N-1} [100(x_{i+1} - x_i^2)^2 - (x_i - 1)^2]$	$[-5.12, 5.12]$	0 (min)
h_{11}	$h_{11} = 6N + \sum_{i=1}^{N} \lvert x_i \rvert$	$[-5.12, 5.12]$	0 (min)
h_{12}	$h_{12} = \frac{1}{4000} \sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]$	0 (min)

۴-۲ تنظیمات شبیه‌سازی

در مطالعه کنونی، RCGA اصلی با عملگر ADM که در آن هر ژن با یک عدد اعشاری ۶۴ بیت نشان داده شده، تقویت شده است. در ضمن انتخاب و برش و جهش بر روی ژن‌ها با ارزش واقعی اعمال می‌شوند و فرزندان در نسل‌ها بوسیله انتخاب SUS و برش BLX-α با پوشش متقاطع و عملگر ADM تولید می‌شوند. سه معیار توقف مختلف وجود دارد:

۱) به تعداد حداکثر تولیدنسل رسیده باشد.

۲) ۱۰۰۰ یا ۲۰۰۰ تکرار در مقدار یکسان شایستگی باقی مانده است.

۳) خطای مطلق بین جواب بدست آمده و بهینه سراسری کمتر از حد آستانه (به طور کلی 10^{-8}) است.

هر زمانی‌که هرکدام از این سه شرط مشاهده شود فرایند تکامل در رویکرد ADM-RCGA پیشنهادی به پایان خواهد رسید. با توجه به طبیعت و منشأ تصادفی الگوریتم‌های تکاملی عملکرد تمام الگوریتم‌های مقایسه شده در هر تابع محک براساس آمار گرفته شده از ۳۰ تا ۱۰۰ مجموعه اجرا در آزمایش‌های مختلف چنان‌که در **جدول ۴** نشان داده شده است، ارزیابی می‌شود. علاوه بر این به منظور ارائه‌ی یک ارزیابی عادلانه در میان الگوریتم‌های مقایسه شده، بعد متغیرها، تعداد اندازه جمعیت و حداکثر تعداد ارزیابی‌های میزان شایستگی تمام الگوریتم‌های تست در هر آزمایش، ثابت فرض شده است. همانطور که در **جدول ۴** ارائه شده است.

۴-۳ مقایسه‌ی عملکرد ADM-RCGA

در الگوریتم ژنتیک متداول، عملکرد یک الگوریتم ژنتیک معمولا براساس سه معیار ارزیابی و اندازه گیری می‌شود: صحت و دقت پایایی و قابلیت اطمینان و کارایی یا بهره‌وری الگوریتم [۱]

صحت و دقت ارزیابی: درجه‌ای از دقت را در محل ماکسیمم بهینه‌ی سراسری بررسی می‌کند. قابلیت اطمینان: میزان دقت پراکندگی در جواب‌های بدست آمده را اندازه گیری می‌کند و کارایی یا بهره‌وری الگوریتم: میزان همگرایی الگوریتم را ارزیابی می‌کند. مطالعه کنونی از یک مقدار فاصله‌ی $|Mean - f^*|$ برای وصف و شرح صحت بین بهینه‌ی سراسری f^* و میانگین $Mean$ جواب‌های بدست آمده استفاده می‌کند. نتیجه بهتر بوسیله یک $|Mean - f^*|$ ارائه می‌شود که به صفر نزدیک‌تر است. انحراف معیار جواب‌ها بدست آمده به کار گرفته می‌شود تا پایانی و قابلیت اطمینان یک الگوریتم ژنتیک را مقایسه کند. انحراف معیارهای کوچکتر، ثبات را نشان می‌دهد و در نتیجه جواب‌های نهایی قابل اطمینان‌تر هستند. [۲۱]

میانگین تعداد ارزیابی‌های تابع و زمان اجرای کامپیوتر برای کارایی یک الگوریتم ژنتیک نشان داده می‌شود. تعداد متوسط کوچکتر از ارزیابی‌های تابع و زمان اجرای پایین‌کامپیوتر نشان از کارآمد بودن یک الگوریتم ژنتیک است.

۴-۳-۱ مقایسه‌ی با پنج عملگر جهش متداول

به منظور اثبات این‌که با عملگر جهش ADM پیشنهادی یک بهبود و پیشرفت عمومی در جهش‌های متداول موجود در ادبیات از جمله RM ، PLM ، NUM ، MNUM و PM دارد، مطالعه‌ی کنونی رفتار ADM را در طول بازه‌ی گسترده‌ای از توابع محک به خوبی سازماندهی شده چنانکه در آزمایش ۱ **جدول ۴** نشان داده شده، شاخصه‌ها و ویژگی‌های کلی مسئله که نشان‌دهنده بهینه‌سازی دنیای واقعی است، را به خلاصه ارائه و بررسی می‌کند. [۱]

مقایسه‌های متناظر عملکرد متوسط مقادیر میانگین میزان شایستگی و رتبه‌های دقت بدست آمده توسط ۶ عملگر جهش مختلف برای توابع محک I با ۳۰ متغیر در ۳۰ اجرا در **جدول ۸** فهرست شده اند. در ۱۶ مورد عملگر ADM کاملا از ۵ عملگر جهش متداول دیگر موفق‌تر است و عملگر جهش ADM بهترین عملکرد را با رتبه ۱ نهایی داراست. دقیق‌تر اینکه مقدار فاصله $|Mean - f^*|$ بین بهینه‌ی سراسری و میانگین جواب‌های بدست آمده توسط ADM برای تمام توابع به استثنای سه مورد (توابع محک f_8 , f_9 ,f_{10}) از ۰ و۱ فراتر نمی‌رود.

همچنین جدول ۸ مقایسه‌های متناظر انحراف معیارهای مقادیر میزان شایستگی و رتبه‌های قابل اطمینان بودن را در پرانتز نشان می‌دهد. در ۱۶ مورد از ۲۰ مورد ADM بهترین عملکردها را با رتبه‌های نهایی به نمایش می‌گذارد. با توجه به جدول ۸ ADM دقیق‌ترین و قابل اطمینان‌ترین الگوریتم در میان پنج عملگر جهش متداول دیگر است. بعلاوه MNUM کمی از PLM برتر است و MNUM براساس میانگین رتبه در جایگاه دوم قرار می‌گیرد. از سوی دیگر NUM و PM برترین عملکرد به ترتیب با رتبه های ۵ و ۶ دارند.

جدول ۸ : میانگین و انحراف معیار میزان شایستگی و رتبه‌ی ۶ عملگر جهش مختلف با ۳۰ متغیر در ۳۰ اجرا

توابع محک	ADM[1]			RM			PLM		
	Mean (SD)	$\lvert Mean - f^* \rvert$	Rank	Mean (SD)	$\lvert Mean - f^* \rvert$	Rank	Mean (SD)	$\lvert Mean - f^* \rvert$	Rank
f_1	4.254E−01	4.254E−01	1	1.424E−03	1.424E−03	3	3.471E−04	3.471E−04	1
	(2.454E−02)		(1)	(7.777E−04)		(2)	(1.173E−04)		(1)
f_2	4.00000	2.323E−07	2	2.99995	4.717E−05	3	2.99998	2.350E−05	2
	(4.714E−07)		(2)	(2.627E−05)		(3)	(9.229E−06)		(2)
f_3	1.2340	2.200E−07	3	0.999994	5.660E−06	4	0.999996	3.557E−06	3
	(6.000E−08)		(2)	(2.387E−06)		(4)	(1.120E−06)		(2)
f_4	4.832E−03	4.432E−03	2	1.362E−02	1.362E−02	3	5.736E−03	5.736E−03	2
	(8.456E−03)		(2)	(1.582E−02)		(3)	(7.968E−03)		(2)
f_5	0.000E+00	0.000E+00	3	2.925E−07	2.925E−07	4	8.635E−08	8.635E−08	3
	(0.000E+00)		(3)	(2.685E−07)		(4)	(8.852E−08)		(3)
f_6	0.000E+00	0.000E+00	3	1.018E−06	1.018E−06	4	3.397E−07	3.397E−07	3
	(0.000E+00)		(3)	(1.394E−06)		(4)	(6.662E−07)		(3)
f_7	100000.15	3.190E−01	3	997842.93	2.454E+01	4	997854.04	1.343E+01	3
	(1.348E−01)		(3)	(1.596E+01)		(4)	(7.871E+00)		(3)
f_8	2.360E+01	1.360E+01	1	1.296E−02	1.296E−02	2	4.310E−03	4.310E−03	1
	(3.552E+00)		(1)	(1.062E−02)		(2)	(2.389E−03)		(1)
f_9	5.434E+00	6.584E+00	3	6.981E+01	6.981E+01	5	5.126E+01	5.126E+01	3
	(4.432E+00)		(4)	(4.075E+01)		(4)	(3.432E+01)		(3)
f_{10}	3.102E+04	6.514E+02	2	1.257E+04	1.487E−01	2	1.257E+04	9.467E−02	1
	(5.996E+02)		(2)	(7.399E−02)		(2)	(3.263E−02)		(1)
f_{11}	4.0323	0.000E+00	4	3.49939	6.123E−05	4	3.499962	3.837E−05	3
	(0.000E+00)		(4)	(3.642E−05)		(4)	(2.605E−05)		(3)
f_{12}	0.000E+00	0.000E+00	4	4.052E+00	4.052E+00	4	3.076E−01	3.076E−01	3
	(0.000E+00)		(4)	(5.574E+00)		(4)	(3.246E−01)		(3)
f_{13}	0.000E+00	0.000E+00	5	3.310E−06	3.310E−06	5	5.618E−07	5.618E−07	3
	(0.000E+00)		(5)	(1.811E−06)		(5)	(3.460E−07)		(3)
f_{14}	0.000E+00	0.000E+00	5	7.572E−05	7.572E−05	5	5.361E−06	5.361E−06	3
	(0.000E+00)		(5)	(1.083E−04)		(5)	(5.511E−06)		(3)
f_{15}	0.000E+00	0.000E+00	(4)	1.771E−03	1.771E−03	4	3.358E−04	3.358E−04	3
	(0.000E+00)		3	(6.387E−03)		(4)	(7.331E−04)		(3)
f_{16}	0.000E+00	0.000E+00	(3)	2.431E−01	2.431E−01	4	4.765E−02	4.765E−02	3
	(0.000E+00)		3	(2.371E−01)		(4)	(1.578E−02)		(3)
f_{17}	6.996E+04	6.996E+04	(3)	1.062E−01	1.062E−01	4	5.003E−02	5.003E−02	3
	(2.638E−04)		3	(4.176E−02)		(4)	(2.161E−02)		(2)
f_{18}	0.000E+00	0.000E+00	(2)	3.977E−04	3.977E−04	4	6.360E−05	6.360E−05	3
	(0.000E+00)		3	(2.793E−04)		(4)	(6.003E−05)		(3)
f_{19}	0.000E+00	0.000E+00	(3)	3.337E−06	3.337E−06	5	2.112E−06	2.112E−06	3
	(0.000E+00)		3	(4.186E−06)		(4)	(3.811E−06)		(3)
f_{20}	0.000E+00	0.000E+00	(3)	5.660E−05	5.660E−05	5	3.864E−06	3.864E−06	3
	(0.000E+00)		3	(7.331E−05)		(5)	(3.075E−06)		(3)
میانگین رتبه			2.1			3.9			2.6
			(2.0)			(3.75)			(2.5)
رتبه نهایی			1			**4**			**3**
			(1)			**(4)**			**(2)**

[1] خروجی ها به رنگ قرمز امده است

توابع محک	NUM			MNUM			PM		
	Mean (SD)	$\lvert Mean - f^* \rvert$	Rank	Mean (SD)	$\lvert Mean - f^* \rvert$	Rank	Mean (SD)	$\lvert Mean - f^* \rvert$	Rank
f_1	5.908E−04	5.908E−04	2	1.816E−01	1.816E−01	4	4.459E−01	4.459E−01	6
	(8.682E−04)		(3)	(6.085E−02)		(5)	(2.104E−01)		(6)
f_2	2.99985	1.473E−04	4	2.98028	1.972E−02	6	2.99358	6.420E−03	5
	(1.355E−04)		(4)	(5.024E−02)		(6)	(5.855E−03)		(5)
f_3	0.999986	1.389E−05	5	0.999997	2.663E−06	2	0.999560	4.396E−04	6
	(9.881E−06)		(5)	(1.724E−06)		(3)	(3.657E−04)		(6)
f_4	2.447E−02	2.447E−02	4	3.655E−02	3.655E−02	5	1.021E+00	1.021E+00	6
	(2.351E−02)		(4)	(3.490E−02)		(5)	(8.691E−02)		(6)
f_5	6.430E−07	6.430E−07	5	0.000E+00	0.000E+00	1	5.924E−04	5.924E−04	6
	(9.436E−07)		(5)	(0.000E+00)		(1)	(1.555E−03)		(6)
f_6	3.772E−06	3.772E−06	5	0.000E+00	0.000E+00	1	1.425E−03	1.425E−03	6
	(1.094E−05)		(5)	(0.000E+00)		(1)	(3.002E−03)		(6)
f_7	997817.27	5.020E+01	5	997866.54	9.290E−01	2	997452.45	4.150E+02	6
	(1.962E+01)		(5)	(3.470E−01)		(2)	(1.371E+02)		(6)
f_8	4.432E−02	4.432E−02	3	1.801E+01	1.801E+01	6	8.233E+00	8.233E+00	4
	(3.624E−02)		(3)	(7.725E+00)		(6)	(6.066E+00)		(5)
f_9	6.670E+01	6.670E+01	4	4.032E+00	4.032E+00	1	4.043E+03	4.043E+03	6
	(4.945E+01)		(5)	(4.002E+00)		(1)	(1.617E+04)		(6)
f_{10}	1.257E+04	2.827E−01	3	1.200E+04	5.726E+02	5	1.255E+04	1.726E+01	4
	(3.200E−01)		(3)	(4.659E+02)		(5)	(5.940E+01)		(4)
f_{11}	3.499794	2.056E−04	5	3.499995	4.733E−06	2	3.495383	4.617E−03	6
	(1.602E−04)		(5)	(1.526E−06)		(2)	(4.471E−03)		(6)
f_{12}	4.877E+01	4.877E+01	5	0.000E+00	0.000E+00	1	7.091E+01	7.091E+01	6
	(3.077E+01)		(6)	(0.000E+00)		(1)	(3.038E+01)		(5)
f_{13}	1.128E−06	1.128E−06	4	0.000E+00	0.000E+00	1	1.195E−02	1.195E−02	6
	(1.701E−06)		(4)	(0.000E+00)		(1)	(9.260E−03)		(6)
f_{14}	2.504E−05	2.504E−05	4	0.000E+00	0.000E+00	1	4.649E−01	4.649E−01	6
	(4.956E−05)		(4)	(0.000E+00)		(1)	(3.203E−01)		(6)
f_{15}	7.410E−03	7.410E−03	5	0.000E+00	0.000E+00	1	4.600E−02	4.600E−02	6
	(3.535E−02)		(5)	(0.000E+00)		(1)	(1.101E−01)		(6)
f_{16}	9.551E−01	9.551E−01	6	5.888E−08	5.888E−08	2	8.571E−01	8.571E−01	5
	(1.646E+00)		(6)	(1.378E−07)		(2)	(1.087E+00)		(5)
f_{17}	2.609E−01	2.609E−01	5	3.240E−02	1.798E−08	2	2.756E+00	2.756E+00	6
	(1.365E−01)		(5)	(2.250E−02)		(3)	(2.349E+00)		(6)
f_{18}	7.913E−04	7.913E−04	5	1.798E−08	1.798E−08	2	6.917E−01	6.917E−01	6
	(1.287E−03)		(5)	(4.834E−08)		(2)	(1.018E+00)		(6)
f_{19}	2.565E−06	2.565E−06	4	0.000E+00	0.000E+00	1	6.102E−02	6.102E−02	6
	(4.507E−06)		(5)	(0.000E+00)		(1)	(7.037E−02)		(6)
f_{20}	3.032E−05	3.032E−05	4	0.000E+00	0.000E+00	1	5.891E−01	5.891E−01	6
	(5.318E−05)		(4)	(0.000E+00)		(1)	(3.165E−01)		(6)
میانگین رتبه			4.35			2.35			5.7
			(4.55)			(2.5)			(5.7)
رتبه نهایی			5			2			6
			(5)			(2)			(6)

برای نشان دادن اهمیت الگوریتم برنده، جدول ۹ مقدار p از یک آزمون t (برحسب ٪) و نتیجه آزمون t برای توابع محک با ۳۰ متغیر در ۳۰ اجرا که توسط ADM بدست آمده را نیز در مقابل مواردی که توسط ۵ عملگر جهش متداول دیگر بدست آمده فهرست کرده است. با استفاده از آزمون t دو طرفه با ۵۸ درجه آزادی در سطح معنی‌دار ۵٪ نتیجه آزمون t در قالب «+» «-» یا «~» ارائه می شود. علامت «~» نشان می‌دهد که هیچ تفاوت قابل توجهی در مقادیر میزان شایستگی توسط هر دو الگوریتم مقایسه شده وجود ندارد، و علامت «+» یا «-» نشان می‌دهد که میانگین مقادیر میزان شایستگی بدست آمده بوسیله ADM به طور قابل توجهی از مقادیر میزان شایستگی بدست آمده توسط دیگر الگوریتم‌ها بهتر یا بدتر است.

به طور واقع آمار نشان داده شده در جدول ۹ بیان می‌کنند که ADM با بیش از ۹۵٪ اعتماد به طور قابل توجهی از RM و PLM در ۱۷ تست از ۲۰ مورد تست بهتر عمل کرده است. ADM پیشنهادی همچنین MNUM را در ۹ مورد NUM را در ۱۶ مورد و PM را در ۱۸ مورد از ۲۰ مورد تست به ترتیب شکست داده شده است. می‌توان مشاهده نمود که مقایسه‌ی ADM پیشنهادی با ۵ عملگر جهش متداول دیگر از لحاظ آماری برای بیشتر موارد تست در توابع محک I معنی‌دار و دارای اهمیت است.

متوسط زمان اجرا، متوسط تعداد ارزیابی‌ها تابع (= متوسط تعداد تولیدها) به رتبه‌های کارایی و کارآمدی بدست آمده توسط ۶ عملگر جهش برای توابع محک I با ۳۰ متغیر در ۳۰ اجرا در جدول ۱۰ نشان داده شده‌اند. از نتایج جدول ۱۰ می‌فهمیم که ADM هم پایین‌ترین زمان اجرا و هم پایین‌ترین تعداد ارزیابی‌های تابع را در ۱۹ مورد از ۲۰ مورد در مقایسه با ۵ عملگرد جهش متداول دیگر دارد.

بعلاوه MNUM جایگاه دوم بهترین عملکرد را از لحاظ میانگین زمان اجرا و میانگین ارزیابی میزان شایستگی دارد. در مقابل RM بدترین عملکرد را با رتبه نهایی ۶ از لحاظ میانگین ارزیابی‌های تابع دارد. از جدول ۸ و جدول ۹ ، جدول ۱۰ می‌توانیم نتیجه بگیریم که ADM دارای دقت بالا، قابلیت اطمینان بیشتر و کارایی و کارآمدی بالا نسبت به پنج عملگر جهش دیگر فراهم می‌کند.

جدول ۹: نتیجه آزمون t روی ۶ عملگر جهش مختلف با ۳۰ متغیر در ۳۰ اجرا – **بدون تغییر**

توابع محک	ADM-RM		ADM-PLM		ADM-NUM		ADM-MNUM		ADM-PM	
	P (t-test) (%)	Result	P (t-test) (%)	Result	P (t-test) (%)	Result	P (t-test) (%)	Result	P (t-test) (%)	Result
f_1	0	-	0	-	0	-	0	-	0	+
f_2	0	+	0	+	0	+	0.02	+	0	+
f_3	0	+	0	+	0	+	0	+	0	+
f_4	0.67	+	52.60	~	0.01	+	0	+	0	+
f_5	0	+	0	+	0.08	+	~	~	0.02	+
f_6	0.04	+	0.93	+	6.90	~	~	~	0	+
f_7	0	+	0	+	0	+	0	+	0	+
f_8	0	-	0	-	0	-	0	+	0	-
f_9	0	+	0	+	0	+	0	-	1.42	+
f_{10}	0	-	0	-	0	-	30.02	~	0	-
f_{11}	0	+	0	+	0	+	0	+	0	+
f_{12}	0.04	+	0	+	0	+	~	~	0	+
f_{13}	0	+	0	+	0.11	+	~	~	0	+
f_{14}	0.07	+	0	+	0.97	+	~	~	0	+
f_{15}	13.97	~	1.80	+	26.03	~	~	~	0.1	+
f_{16}	0	+	0	+	0.35	+	0	+	0	+
f_{17}	0	+	0	+	0	+	0	+	0	+
f_{18}	0	+	0	+	0.21	+	0.03	+	0	+
f_{19}	0.01	+	0.05	+	0.41	+	~	~	0	+
f_{20}	0.02	+	0	+	0.40	+	~	~	0	+

جدول ۱۰: میانگین زمان اجرا، متوسط تعداد نسل‌ها و رتبه‌ی کارایی ۶ عملگر جهش مختلف با ۳۰ متغیر در ۳۰ اجرا – **بدون تغییر**

Test functions	ADM		RM		PLM		NUM		MNUM		PM	
	Time(s) (generation)	Rank	Time(s) (generation)	Rank	Time(s) (generation)	Rank	Time(s) (generation)	Rank	Time(s) (generation)	Rank	Time(s) (generation)	Rank
f_1	91.25 (2348)	1 (1)	3389.03 (29824)	6 (5)	3053.57 (30000)	5 (6)	874.63 (29590)	3 (4)	453.05 (17290)	2 (2)	933.23 (25554)	4 (3)
f_2	40.05 (1041)	1 (1)	963.21 (8186)	6 (4)	752.63 (6458)	4 (3)	301.56 (11582)	3 (5)	182.97 (4991)	2 (2)	876.98 (23647)	5 (6)
f_3	39.37 (1077)	1 (1)	902.13 (7692)	6 (4)	658.11 (5549)	4 (3)	280.31 (11162)	3 (5)	151.59 (4427)	2 (2)	675.49 (20142)	5 (6)
f_4	20.47 (519)	1 (1)	3278.67 (28743)	6 (5)	2987.69 (27288)	5 (3)	819.56 (27915))	3 (4)	478.1 (15118)	2 (2)	900.25 (28864)	4 (6)
f_5	9.98 (244)	1 (1)	2667.24 (22001)	6 (4)	2121.33 (19733)	5 (3)	813.14 (25761)	3 (5)	245.73 (7677)	2 (2)	942.46 (29677)	4 (6)
f_6	11.27 (276)	1 (1)	3113.87 (25075)	6 (4)	2136.9 (19808)	5 (3)	719.52 (26649)	3 (5)	266.36 (8988)	2 (2)	864.48 (28725)	4 (6)
f_7	48.38 (1205)	1 (1)	1766.65 (9835)	6 (6)	1250.83 (7566)	5 (4)	363.62 (9387)	4 (5)	97.58 (2489)	3 (3)	52.14 (1332)	2 (2)
f_8	57.03 (1456)	1 (1)	1638.42 (12978)	6 (4)	971.72 (10467)	5 (3)	516.03 (16410)	3 (5)	214.95 (6468)	2 (2)	709.02 (20712)	4 (6)
f_9	1025.65 (30000)	3 (3)	3198.19 (29863)	6 (2)	2355.1 (29625)	5 (1)	778.72 (30000)	1 (3)	853.46 (30000)	2 (3)	1074.35 (30000)	4 (3)
f_{10}	41.46 (1091)	1 (1)	1491.56 (8555)	6 (4)	647.87 (6837)	5 (3)	379.56 (10171)	3 (5)	173.61 (4991)	2 (2)	404.41 (11487)	4 (6)
f_{11}	45.72 (1174)	1 (1)	1086.02 (10573)	6 (4)	853.5 (8706)	5 (3)	474.1 (14122)	3 (5)	144.82 (3887)	2 (2)	748.96 (23201)	4 (6)
f_{12}	70.38 (1769)	1 (1)	1249.5 (14286)	5 (4)	1306.32 (15252)	6 (5)	239.33 (6536)	3 (3)	687.27 (27744)	4 (6)	203.16 (5454)	2 (2)
f_{13}	11.45 (285)	1 (1)	2577.08 (29500)	6 (4)	2538.16 (29678)	5 (5)	899.87 (29788)	3 (6)	336.02 (11585)	2 (2)	994.49 (28498)	4 (3)
f_{14}	12.91 (323)	1 (1)	2588.35 (29472)	6 (4)	2543.91 (29697)	5 (6)	955.39 (29575)	4 (5)	417.93 (15230)	2 (2)	953.51 (29286)	3 (3)
f_{15}	2.77 (70)	1 (1)	383.62 (3001)	5 (3)	633.16 (6001)	6 (6)	77.59 (2124)	2 (2)	172.34 (4862)	3 (5)	175.85 (4811)	4 (4)
f_{16}	347.17 (8980)	1 (1)	2438.82 (21243)	6 (2)	2325.47 (25803)	5 (5)	809.03 (23514)	4 (3)	723.57 (29944)	2 (6)	806.3 (25296)	3 (4)
f_{17}	89.96 (2305)	1 (1)	400.49 (3153)	6 (4)	318.77 (2677)	5 (2)	158.56 (4486)	3 (5)	98.91 (2922)	2 (3)	215.53 (5859)	4 (6)
f_{18}	16.57 (418)	1 (1)	3165.91 (23108)	6 (4)	2131.37 (20009)	5 (3)	833.26 (24005)	4 (5)	327.69 (12157)	2 (2)	726.3 (27452)	3 (6)
f_{19}	15.76 (386)	1 (1)	3095.71 (27337)	6 (4)	2259.14 (25014)	5 (3)	886.22 (29374)	4 (6)	269.46 (10005)	2 (2)	782.56 (28467)	3 (5)
f_{20}	15.4 (386)	1 (1)	3376.56 (28599)	6 (5)	2480.88 (28435)	5 (4)	905.79 (29014)	4 (6)	318.89 (12509)	2 (2)	753.72 (28202)	3 (3)
Averaged rank		1.1 (1.1)		5.9 (4)		5 (3.7)		3.15 (4.6)		2.2 (2.7)		3.65 (4.6)
Final rank		1 (1)		6 (4)		5 (3)		3 (5)		2 (2)		4 (5)

۲-۳-۴ مقایسه‌ی با CMA-ES و HYK-GA9

مطالعه کنونی به مقایسه شیوه ADM-RCGA پیشنهادی با برخی الگوریتم‌های تکاملی انطباقی برجسته دیگر علاوه بر عملگرهای جهش متداول علاقمند است. ما شیوه‌های جهش انطباقی مدرن CMA-ES [۱۹] و HYK-GA [۲۱] دوتا از قوی‌ترین رقبا را انتخاب می کنیم. CMA-ES از یک شیوه تعلیم هوشمند برای انطباق ماتریسم کوواریانس کامل یک توزیع جهش نرمال استفاده می‌کند.

ایده‌ی اصلی این شیوه جمع‌آوری اطلاعات در مورد گام‌های جستجوی موفق و استفاده از آن اطلاعات برای اصلاح و تعدیل قطعی ماتریس کوواریانس توزیع جهش می‌باشد. از سوی دیگر، HKY-GA یکی از آخرین و جدیدترین مدل‌های میزان جایگزینی نوکلئوتید مارکوف[1] مدل تکاملیHKY را به کار گرفته تا بر روی یک RCGA اعمال کند. هدف شیوه HKY-GA نه تنها انطباق میزان جهش است بلکه سعی دارد انواع ژن‌هایی که هدف جهش قرار می‌گیرند را نیز تعیین کند. برای فراهم‌سازی یک مقایسه عادلانه بین CMA-ES و HEY-GA و ADM-RCGA پیشنهادی شرایط شبیه سازی (همانند[۲۱]) در آزمایش ۲ **جدول ۴** داده شده‌اند.

مقایسه عملکرد مقادیر میانگین میزان شایستگی و رتبه‌های صحت و دقت به دست آمده توسط ADM-RCGA و HEY-GA و CMA-ES برای توابع محک II در ۱۰۰ اجرا در **جدول ۱۲** فهرست شده‌اند. این جدول نشان می‌دهد که ADM-RCGA پیشنهادی بهترین عملکرد را با رتبه نهایی ۱ دارد و اینکه در ۱۶ مورد از ۲۳ مورد ADM-RCGA عملکردی با رتبه ۱ داشته است. دقیق‌تر اینکه مقدار

[1] Markov models of nucleotide

حاصله $|Mean - f^*|$ بین بهینه ی سراسری و میانگین جواب‌های بدست آمده توسط ADM-RCGA برای تمام توابع به استثنای۴ مورد (g_7 وg_3) از ۱ فراتر نمی‌رود. در طرف مقابل HEY-GA و CMA-ES به ترتیب عملکرد با رتبه‌ی ۱ از ۱۲ و ۷ از ۲۳ مورد دارند. مقدار فاصله‌ی $|Mean - f^*|$ برای هر چهار مورد (توابع تست g_3 و g_7) در HEY-GA و هشت مورد (توابع تست g_2, g_3, g_5, g_6, g_9) در CMA-ES از ۱ فراتر می‌رود.

جدول ۱۲ مقایسه‌های متناظر انحراف معیارهای مقادیر میزان شایستگی و رتبه های قابل اطمینان بودن در پرانتز را نیز نشان می‌دهد. در ۱۶ مورد از ۲۳ مورد ADM-RCGA بهترین عملکرد را با رتبه‌ی نهایی۱ دارد. به عبارت دیگر به ترتیب ۱۳ مورد و ۷ مورد از ۲۳ مورد در HEM-GA و CMA-EA دارای رتبه۱ هستند.

نشان می‌دهد که AMD-RCGA دقیق‌ترین و قابل اطمینان الگوریتم و بهتر از HEY-GA و CMA-ES است.

بعلاوه HEY-GA عملکرد رقابتی نسبت به CMA-GA دارد در حالی‌که CMA-GA با رتبه نهایی ۳ بدترین عملکرد را دارد با به کارگیری آزمون t دو طرفه با ۱۹۸ درجه آزادی در سطح معنی داری ۵٪ ، آمار نشان داده شده در جدول ۱۱ بیان می‌کنند که ADM-ECGA با اعتماد بیش از ۹۵٪ به طور معناردار و محسوسی بهتر از HEY-GA در ۱۰ مورد عمل کرده است. در طرف مقابل، میانگین مقادیر میزان شایستگی ADM-RCGA به طور قابل توجهی ضعیف‌تر از HEY-GA و CMA-ES به ترتیب در ۷ و ۵ مورد هستند.

همچنین می‌توان مشاهده نمود که ADM-RCGA پیشنهادی از لحاظ آماری دارای اهمیت و کمی برتر از HEY-GA و CMA-ES می‌باشد. می‌توانیم از جدول ۱۱و جدول ۱۲ نتیجه‌گیری کنیم که ADM-RCGA صحت و دقت بیشتر، قابلیت اطمینان بیشتر و کارایی بالاتری از از HEY-GA و CMA-ES دارد.

جدول ۱۱: عملکرد مقایسه‌ای ADM ، HEY-GA و CMA-ES با آزمون t در ۱۰۰ اجرا

توابع محک	بعد	ADM–HEY-GA Result(t-test)	ADM–CMA-ES Result(t-test)
g_1	10	~	~
	20	~	~
	30	~	~
g_2	10	~	~
	20	+	+
	30	+	~
g_3	10	-	-
	20	-	-
	30	-	-
g_4	10	+	+
	20	+	+
	30	+	+
g_5	10	-	+
	20	-	-
	30	-	-
g_6	10	+	+
	20	~	+
	30	-	+
g_7	10	~	~
	20	+	~
	30	+	~
g_8	2	~	+
g_9	2	~	+

جدول ۱۲: عملکرد مقایسه‌ای ADM ، HEY-GA و CMA-ES در ۱۰۰ اجرا

توابع محک	بعد	ADM			HEY-GA [21]			CMA-ES [19]		
		Mean (SD)	$\lvert Mean - f^* \rvert$	Rank	Mean (SD)	$\lvert Mean - f^* \rvert$	Rank	Mean (SD)	$\lvert Mean - f^* \rvert$	Rank
g_1	10	0.000E+00	0.000E+00	1	0.000E+00	0.000E+00	1	0.000E+00	0.000E+00	1
		(0.000E+00)		(1)	(0.000E+00)		(1)	(0.000E+00)		(1)
	20	0.000E+00	0.000E+00	2	0.000E+00	0.000E+00	1	0.000E+00	0.000E+00	1
		(0.000E+00)			(0.000E+00)		(1)	(0.000E+00)		(1)
	30	-	-	-	0.000E+00	0.000E+00	1	0.000E+00	0.000E+00	1
		-	-	-	(0.000E+00)		(1)	(0.000E+00)		(1)
g_2	10	0.000E+00	0.000E+00	(2)	0.000E+00	0.000E+00	1	1.206E+00	1.206E+00	3
		(0.000E+00)		(2)	(0.000E+00)		(1)	(8.443E+00)		(3)
	20	2.995E−02	2.995E−02	3	2.000E−02	2.000E−02	2	7.600E−02	7.600E−02	3
		8.456E−03		(3)	(7.000E−03)		(2)	(2.350E−01)		(3)
	30	-	-	-	4.800E−02	4.800E−02	2	1.514E+00	1.514E+00	3
		-	-	-	(1.000E−02)		(2)	(8.026E+00)		(3)
g_3	10	5.993E+00	5.432E−03	2	8.800E−02	8.800E−02	1	3.990E−01	3.990E−01	2
		(1.715E+01)		(2)	(8.770E−01)		(1)	(1.196E+00)		(2)
	20	-	-	-	1.158E+00	1.158E+00	2	7.570E−01	7.570E−01	1
		-	-	-	(4.459E+00)		(2)	(1.564E+00)		(1)
	30	-	-	-	3.018E+00	3.018E+00	1	3.660E+00	3.660E+00	2
		-	-	-	(8.640E+00)		(1)	(8.745E+00)		(2)
g_4	10	0.000E+00	0.000E+00	3	9.000E−03	9.000E−03	2	2.100E−02	2.100E−02	3
		(0.000E+00)		(3)	(1.000E−02)		(2)	(1.100E−02)		(3)
	20	0.000E+00	0.000E+00	3	9.000E−03	9.000E−03	2	4.300E−02	4.300E−02	3
		(0.000E+00)		(3)	(1.300E−02)		(2)	(1.700E−02)		(3)
	30	-	-	-	2.100E−02	2.100E−02	2	6.200E−02	6.200E−02	3
		-	-	-	(3.100E−02)		(3)	(1.400E−02)		(2)
g_5	10	100000.15	3.190E−01	2	1.600E−02	1.600E−02	1	1.227E+00	1.227E+00	3
		-	-	-	(9.000E−02)		(2)	(2.998E+00)		(3)
	20	-	-	-	0.000E+00	0.000E+00	1	2.000E−03	2.000E−03	2
		-	-		(0.000E+00)		(1)	(5.000E−03)		(2)
	30	-	-	-	0.000E+00	0.000E+00	1	1.000E−03	1.000E−03	2
		-	-	-	(0.000E+00)		(1)	(4.000E−03)		(2)
g_6	10	0.000E+00	0.000E+00	1	1.760E−01	1.760E−01	2	8.931E+01	8.931E+01	3
		(0.000E+00)		(1)	(2.800E−02)		(2)	(2.282E+01)		(3)
	20	4.079E−01	4.079E−01	3	5.670E−01	5.670E−01	2	1.724E+02	1.724E+02	3
		(1.100E+00)			(3.500E−02)		(1)	(3.434E+01)		(3)
	30	-	-	-	9.930E−01	9.930E−01	1	2.536E+02	2.536E+02	3
		-	-	-	(5.100E−02)		(1)	(4.335E+01)		(3)
g_7	10	0.000E+00	0.000E+00	1	0.000E+00	0.000E+00	1	0.000E+00	0.000E+00	1
		(0.000E+00)		(1)	(0.000E+00)		(1)	(0.000E+00)		(1)
	20	0.000E+00	0.000E+00	1	1.342E+00	1.342E+00	3	0.000E+00	0.000E+00	1
		(0.000E+00)		(1)	(7.120E−01)		(3)	(0.000E+00)		(1)
	30	-	-	-	1.729E+00	1.729E+00	3	0.000E+00	0.000E+00	1
		-	-	-	(2.000E−03)		(3)	(0.000E+00)		(1)
g_8	2	-	-	-	1.031E+00	0.000E+00	1	9.390E−01	9.200E−02	3
		-	-	-	(0.000E+00)		(1)	(2.560E−01)		(3)
g_9	2	-	-	-	1.000E+00	2.000E−03	2	1.069E+01	9.688E+00	3
		-	-	-	(0.000E+00)		(1)	(6.575E+00)		(3)
میانگین رتبه							1.57			2.22
							(1.57)			(2.17)
رتبه نهایی							2			3
							(2)			(3)

۳-۳-۴ مقایسه‌ی با LX-PM

Deep وThakur [۱] یک عملگر جهش جدید به نام PM برای کد واقعی الگوریتم ژنتیک معرفی کردند و شش نسل از کد واقعی الگوریتم ژنتیک در یک مجموعه، با ۲۰ مسئله تست بهینه‌سازی سراسری معیار مقایسه کردند. نتایج آن‌ها نشان داد که RCGA با استفاده از PM در رابطه با برش لاپلاس (LX-PM) بهتر از۵ الگوریتم ژنتیک دیگر عمل می‌کند. برای فراهم آوردن یک مقایسه عادلانه میان LX-PM و ADM-RCGA پیشنهادی شرایط شبیه‌سازی (همانند[۱]) در آزمایش ۳ از جدول ۴ ارائه شده‌اند.

مقایسه‌ی متوسط عملکرد میانگین میزان شایستگی و رتبه‌های صحت و دقت بدست آمده توسط ADM-RCGA و LX-PM برای توابع

محک در ۳۰ اجرا در جدول ۱۳ فهرست شده‌اند. این جدول نشان می‌دهد که ADM-RCGA پیشنهادی با رتبه‌ی نهایی ۱ بهترین عملکرد را دارد و در ۱۳ مورد از ۲۰ مورد ADM-RCGA عملکرد با رتبه ۱ دارد. همچنین جدول ۱۳ مقایسه‌های متناظر انحراف معیارهای میزان شایستگی و رتبه‌های قابلیت اطمینان را در پرانتز نشان می‌دهد. نتایج جدول ۱۳ بیان می‌کنند که ADM-RCGA با رتبه نهایی ۱ و رتبه ی ۱ در ۱۴ مورد از ۲۰ مورد بهترین عملکرد را دارد. به علاوه تنها در ۵ مورد از ۲۰ مورد LX-PM رتبه ۱ دارد.

با استفاده از آزمون t دو طرفه با درجه‌ی آزادی ۵۸ در سطح قابل توجه ۵٪ ، آمار نشان داده شده در جدول ۱۳ نیز بیان می‌کنند که ADM-RCGA با بیش از ۹۵٪ اعتماد، به طور قابل توجهی در ۸ مورد تست عملکرد بهتری از LX-PM دارد. در طرف مقابل میانگین میزان شایستگی توسط ADM-RCGA در ۷ مورد از LX-PM ضعیف‌تر است. ADM-RCGA پیشنهادی از لحاظ آماری دارای اهمیت است و نسبتا از LX-PM برتری دارد. می‌توانیم از جدول ۱۳ نتیجه‌گیری بگیریم که ADM-RCGA دقت و صحت بیشتر، قابلیت اطمینان بیشتر و کارایی بالاتری از LX-PM دارد.

جدول ۱۳: عملکرد مقایسه‌ای ADM ، LX-PM با ۳۰ متغیر در ۳۰ اجرا

توابع محک	ADM			LX-PM [1]			ADM–LX-PM
	Mean (SD)	$\lvert Mean - f^* \rvert$	Rank	Mean (SD)	$\lvert Mean - f^* \rvert$	Rank	Result(t-test)
f_1	4.254E−01	4.254E−01	1	1.010E−10	1.010E−10	1	–
	(2.454E−02)		(1)	(1.040E−10)		(1)	
f_2	4.00000	2.323E−07	2	3.000E+00	0.000E+00	1	~
	(4.714E−07)		(2)	(0.000E+00)		(1)	
f_3	1.2340	2.200E−07	3	1.000E+00	0.000E+00	1	~
	(6.000E−08)		(2)	(1.730E−08)		(2)	
f_4	4.832E−03	4.432E−03	2	1.940E−03	1.940E−03	1	–
	(8.456E−03)		(2)	(1.570E−03)		(2)	
f_5	0.000E+00	0.000E+00	3	3.200E−19	3.200E−19	2	+
	(0.000E+00)		(3)	(2.890E−19)		(2)	
f_6	0.000E+00	0.000E+00	3	7.950E−23	7.950E−23	2	+
	(0.000E+00)		(3)	(9.920E−23)		(2)	
f_7	100000.15	3.190E−01	3	9.980E+05	1.325E+02	2	+
	(1.348E−01)		(3)	(5.590E+00)		(2)	
f_8	2.360E+01	1.360E+01	1	0.000E+00	0.000E+00	1	–
	(3.552E+00)		(1)	(0.000E+00)		(1)	
f_9	5.434E+00	6.584E+00	3	1.580E+01	1.580E+01	1	–
	(4.432E+00)		(4)	(2.150E+00)		(1)	
f_{10}	3.102E+04	6.514E+02	2	1.260E+04	3.051E+01	1	–
	(5.996E+02)		(2)	(1.850E−12)		(1)	
f_{11}	4.0323	0.000E+00	4	3.500E+00	0.000E+00	1	~
	(0.000E+00)		(4)	(2.980E−08)		(2)	
f_{12}	0.000E+00	0.000E+00	4	1.950E−20	1.950E−20	2	~
	(0.000E+00)		(4)	(6.830E−20)		(2)	
f_{13}	0.000E+00	0.000E+00	5	4.750E−11	4.750E−11	2	+
	(0.000E+00)		(5)	(3.340E−11)		(2)	
f_{14}	0.000E+00	0.000E+00	5	2.820E−12	2.820E−12	2	+
	(0.000E+00)		(5)	(2.090E−12)		(2)	
f_{15}	0.000E+00	0.000E+00	(4)	3.030E−08	3.030E−08	2	+
	(0.000E+00)		3	(1.310E−08)		(2)	
f_{16}	0.000E+00	0.000E+00	(3)	3.600E−05	3.600E−05	1	–
	(0.000E+00)		3	(1.700E−04)		(2)	
f_{17}	6.996E+04	6.996E+04	(3)	2.320E−04	2.320E−04	1	–
	(2.638E−04)		3	(1.950E−04)		(2)	
f_{18}	0.000E+00	0.000E+00	(2)	9.260E−11	9.260E−11	2	+
	(0.000E+00)		3	(1.030E−10)		(2)	
f_{19}	0.000E+00	0.000E+00	(3)	9.480E−32	9.480E−32	2	+
	(0.000E+00)		3	(1.700E−31)		(2)	
f_{20}	0.000E+00	0.000E+00	(3)	6.070E−31	6.070E−31	2	~
	(0.000E+00)		3	(2.240E−30)		(2)	
میانگین رتبه						1.5 (1.75)	
رتبه نهایی						2 (2)	

۴-۳-۴ مقایسه‌ی با IEA, BOA و OGA

Ho و همکاران [۲۸] یک الگوریتم تکاملی هوشمند (IEA) براساس یک طرح تجربی ارتوگونال[1] برای حل مسائل بهینه‌سازی با پارامترهای بزرگ پیشنهاد دادند. آنها برخی EAهای موجود را بر مبنای یک مجموعه ۱۲ مسئله‌ای محک برای پیدا کردن بهینه سراسری، مقایسه کردند. نتایج آنها نشان داد که BOA [۲۹] بهترین عملکرد را برای ۱۰ متغیر در مقایسه با شش EA دیگر دارد و IEA از هر پنج EA دیگر برای ۱۰۰ متغیر هم بهتر عمل می‌کند. به منظور فراهم‌سازی مقایسه‌ای عادلانه بین IEA ، BOA و OGA [۳۰] و ADM-RCGA پیشنهادی شرایط شبیه‌سازی (همانند[۲۸]) در آزمایش ۴ جدول ۴ ارائه شده‌اند.

مقایسه متوسط عملکرد مقادیر سازگاری میانگین و رتبه‌های صحت و دقت بدست آمده توسط ADM-RCGA ، IEA ، BOA و OGA برای توابع معیار III با ۱۰ متغیر در ۳۰ اجرا در جدول ۱۴ فهرست شده‌اند. از این جدول می‌توان دریافت که ADM-RCGA پیشنهادی بهترین عملکرد را با رتبه نهایی ۱ دارد و در ۱۱ مورد از ۱۲ مورد که در آن ADM-RCGA با رتبه‌ی ۱ عمل کرده است. در مقابل، BOA تنها در یک مورد عملکردی با رتبه ۱ دارد و هیچ موردی با رتبه ۱ برای IEA وجود ندارد.

جدول ۱۵ مقایسه متوسط عملکرد از میانگین میزان شایستگی و رتبه‌های دقت و صحت بدست آمده توسط ADM-RCGA ، IEA و OGA برای توابع محک III با ۱۰۰ متغیر در ۳۰ اجرا را نشان می‌دهد. با توجه به زمان محاسبه طولانی، BOA در۱۲ تابع تست با تنها ۱۰ متغیر آزمایش می‌شود[۲۸]. نتایج جدول ۱۵ نشان می‌دهد که ADM-RCGA با رتبه نهایی ۱ و رتبه یک در ۱۰ مورد از ۱۲ مورد بهترین عملکرد را دارد. علاوه براین تنها ۲ مورد از ۱۳ مورد وجود دارد که IEA رتبه ۱ دارد و هیچ موردی (رتبه ۱) برای OGA وجود ندارد.

از جدول ۱۴ و جدول ۱۵ می‌توانیم نتیجه‌گیری کنیم که ADM-RCGA دقت و صحت بیشتر، قابلیت اطمینان بیشتر و کارایی بالاتری از نتایج بدست آمده نسبت به IEA ، BOA و OGA دارد.

جدول ۱۴: عملکرد مقایسه‌ای ADM ، IEA، BOA و OGA با ۱۰ متغیر در ۳۰ اجرا

توابع محک	ADM		
	Mean (SD)	$\lvert Mean - f^* \rvert$	Rank
h_1	7.254E-01	7.254E-01	1
h_2	2.454E-02	3.45E-02	1
h_3	7.00000	7.00000	1
h_4	3.714E-07	3.714E-07	1
h_5	1.2340	1.2340	1
h_6	2.14E-07	5.14E-07	1
h_7	3.714E-07	3.14E-07	1
h_8	3.314E-07	3.414E-07	1
h_9	8.35E-01	94.853E-01	1
h_{10}	2.455E-01	2.455E-01	1
h_{11}	4.655E-01	4.655E-01	1
h_{12}	4.832E-01	4.832E-01	1
میانگین رتبه			1
رتبه‌ی نهایی			1

IEA [28]			BOA [29]			OGA [30]		
Mean	|Mean-f^*|	Rank	Mean	|Mean-f^*|	Rank	Mean	|Mean-f^*|	Rank
5.400E-02	5.400E-02	3	1.900E-02	1.900E-02	2	7.200E-02	7.200E-02	4
1.532E+01	4.680E+00	2	1.229E+01	7.710E+00	4	1.524E+01	4.760E+00	3
5.130E+00	5.130E+00	3	7.700E-01	7.700E-01	2	5.300E+00	5.300E+00	4
1.542E+01	1.542E+01	3	5.320E+00	5.320E+00	2	1.762E+01	1.762E+01	4
3.000E-04	3.000E-04	2	7.700E-03	7.700E-03	4	3.000E-04	3.000E-04	2
1.460E+01	3.900E+00	3	1.810E+01	4.000E-01	2	1.401E+01	4.490E+00	4
1.212E+01	4.380E-02	3	1.215E+01	8.800E-03	2	1.211E+01	5.080E-02	4
1.000E+00	1.000E+00	3	9.300E-01	9.300E-01	2	1.700E+00	1.700E+00	4
6.674E+02	6.674E+02	4	8.400E+00	8.400E+00	1	5.842E+02	5.842E+02	3
1.164E+02	1.164E+02	4	8.930E+00	8.930E+00	2	9.457E+01	9.457E+01	3
3.380E-01	3.380E-01	2	1.007E+01	1.007E+01	4	9.000E-01	9.000E-01	3
9.990E-01	9.990E-01	2	1.008E+00	1.008E+00	4	1.002E+00	1.002E+00	3
		2.83			2.58			3.42
		3			2			4

[1] orthogonal

جدول ۱۵: عملکرد مقایسه‌ای ADM ، IEA و OGA با ۱۰۰ متغیر در ۳۰ اجرا

IEA [28]			OGA [30]		
Mean	|Mean–f^*|	Rank	Mean	|Mean–f^*|	Rank
6.500E−01	6.500E−01	2	1.630E+00	1.630E+00	3
1.532E+02	4.685E+01	2	1.397E+02	6.029E+01	3
6.210E+02	6.210E+02	2	6.107E+03	6.107E+03	3
2.135E+02	2.135E+02	2	3.675E+02	3.675E+02	3
1.600E+00	1.600E+00	2	1.496E+01	1.496E+01	3
1.313E+02	5.369E+01	2	1.155E+02	6.952E+01	3
1.204E+02	1.158E+00	2	1.167E+02	4.888E+00	3
3.690E+00	3.690E+00	2	9.470E+00	9.470E+00	3
8.011E+03	8.011E+03	1	1.229E+04	1.229E+04	2
2.081E+03	2.081E+03	2	5.282E+03	5.282E+03	3
4.394E+01	4.394E+01	1	6.519E+01	6.519E+01	3
3.286E+01	3.286E+01	2	4.825E+01	4.825E+01	3
		1.83			2.92
		2			3

توابع محک	ADM		
	Mean (SD)	$\|Mean - f^*\|$	Rank
h_1			
h_2			
h_3			
h_4			
h_5			
h_6			
h_7			
h_8			
h_9			
h_{10}			
h_{11}			
h_{12}			
میانگین رتبه			
رتبه‌ی نهایی			

۵- نتیجه‌گیری

بررسی کنونی یک عملگر جهش جدید به ADM که در عین سادگی، نیرومندی و کارایی در چارچوب الگوریتم‌های ژنتیک متمرکز است، ارائه می‌کند. برای ارزیابی عملکرد الگوریتم پیشنهادی، ما یک سری آزمایش بر روی مجموعه‌ای از ۴۱ تابع محک شناخته شده با ارزش واقعی در مسائل بهینه‌سازی سراسری انجام داده‌ایم.

هنگامی ADM پیشنهادی را با پنج عملگر جهش متعارف، از جمله RM، PLM، NUM، MNUM و PM مقایسه می‌کنیم، رویکرد ADM پیشنهادی، تحت شرایط شبیه‌سازی یکسان، بهبود قابل توجهی در کیفیت جواب‌های بهینه‌ی سراسری نشان می‌دهد. بررسی کنونی نیز عملکرد ADM-RCGA پیشنهادی را با شش الگوریتم تکاملی پیشتاز، علاوه بر عملگرهای جهش معمولی، مقایسه کرده است. ADM-RCGA پیشنهادی در مقابل روش state-of-the-art adaptive evolutionary، HYK-GA و CMA-ES عملکرد بهتری دارد.

به علاوه، ADM-RCGA پیشنهادی هم بهتر از همه‌ی الگوریتم‌های ژنتیک دیگر، از جمله BOA, IEA, LX-PM و OGA است. در نهایت، می‌توان نتیجه گرفت که ADM-RCGA پیشنهادی نسبت به تمام الگوریتم‌های ژنتیک مورد بررسی شده در این مقاله از دقت بیشتر، قابلیت اطمینان بیشتر، و کارایی(بهره‌وری) بالاتری برخوردار است.

نتیجه‌ی آزمایشات برای ADM-RCGA پیشنهادی در اکثر موارد بسیار عالی است، اما هنوز هم در برخی از توابع به دلیل افزایش خطرات ناشی از تله بهینه محلی بد است. به عنوان چشم انداز آینده، ما برای بهبود بیشتر بهره‌وری تکامل به دنبال یکپارچه‌سازی رویکرد طراحی آزمایش با استفاده از الگوریتم ADM-RCGA ارائه شده، هستیم.

منابع:

[1] K. Deep, M. Thakur, A new mutation operator for real coded genetic algorithm , Applied Mathematics and Computation 193 (2007) 211–230.
[2] C. Heitzunger, S. Selberher, An extensible TCAD optimization framework combing gradient based and genetic optimizers, Microelectronics Jourual 33 (2002) 6168.
[3] T. Back, H.-P. Schwefel, An overview of evolutionary algorithms for parameter optimization, Evolutionary Computation 1 (1) (1993) 1–23.
[4] D.E. Goldberg, Genetic Algorithm in Search, Optimization and Machine Learning, Addison-Wesley, Boston, 1989.
[5] D.A. Coley, An Introduction to Genetic Algorithms for Scientists and Engineers, World Scientific, London, 1998.
[6] D.E. Goldberg, Real-coded genetic algorithms, virtual alphabets, and blocking, Complex Systems 5 (2) (1991) 139–168.
[7] S.H. Ling, F.H.F. Leung, An improved genetic algorithm with average-bound crossover and wavelet mutation operations, Soft Computing 11 (2007) 7–31.
[8] C.Z. Janikow, Z. Michalewicz, An experimental comparison of binary and floating point representation in genetic algorithms, in: Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco, 1991, pp. 31–36.
[9] P.H. Tang, M.H. Tseng, Medical data mining using BGA and RGA for weighting of features in fuzzy k-NN classification, IEEE the International Conference on Machine Learning and Cybernetics 5 (2009) 3070–3075.
[10] I. Korejo, S. Yang, C. Li, A directed mutation operator for real coded genetic algorithms, in: C. Di Chio, et al. (Eds.), EvoApplications, Part I, LNCS 6024, 2010, pp. 491–500.
[11] D.K. Pratihar, Soft Computing, Alpha Science International Ltd., Oxford, UK, 2008.
[12] D. Bhandari, N.R. Pal, S.K. Pal, Directed mutation in genetic algorithms, Information Sciences 79 (1994) 251–270.
[13] Q. Zhou, Y. Li, Directed variation in evolutionary strategies, IEEE Transactions on Evolutionary Computation 7 (4) (2003) 356–366.
[14] A. Berry, P. Vamplew, PoD can mutate: a simple dynamic directed mutation approach for genetic algorithms, in: Proceedings of International Conference on Artificial Intelligence in Science and Technology, 2004, pp. 200–205.
[15] L. Temby, P. Vamplew, A. Berry, Accelerating real valued genetic algorithms using mutation-with-momentum, in: S. Zhang, R.A. Jarvis (Eds.), AI 2005. LNCS (LNAI), 3809, 2005, pp. 1108–1111.
[16] M. Srinivas, L.M. Patnaik, Adaptive probabilities of crossover and mutation in genetic algorithms, IEEE Transaction on System, Man, and Cybernetics 24 (4) (1994) 17–26.
[17] M.S. Chen, F.H. Liao, Adaptive mutation operators and its applications, Journal of Dayeh University 7 (1) (1998) 91–101.
[18] M.H. Tseng, H.C. Liao, The genetic algorithm for breast tumor diagnosis – the case of DNA viruses, Applied Soft Computing 9 (2009) 703–710.
[19] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolutionary strategies, IEEE Transactions on Evolutionary Computation 9 (2) (2001) 159–195.
[20] F. Lobo, C. Lima, Z. Michalewicz, Parameter setting in evolutionary algorithms, Studies in Computational Intelligence 54 (2007) 47–76.
[21] F. Vafaee, P.C. Nelson, A genetic algorithm that incorporates an adaptive mutation based on an evolutionary model, in: 2009 International Conference on Machine Learning and Applications, 2009, pp. 101–107.
[22] J. Baker, Reducing bias and inefficiency in the selection algorithm, in: Proceedings of the First International Conference on Genetic Algorithms, 1985, pp. 14–21.
[23] L.J. Eshelman, J.D. Schaffer, Real-coded genetic algorithms and intervalschemata, Found Genet Algorithms 2 (1993) 187–202.
[24] A.M.S. Zalzala, P.J. Fleming, Genetic algorithms in engineering systems, in: IET, 1997.
[25] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, New York, 1992.
[26] C.R. Houck, J.A. Joines, M.G. Kay, A Genetic Algorithm for Function Optimization: A Matlab Implementation, Technical Report, North Carolina State University, Raleigh, NC, 1996.
[27] K. Deb, M. Goyal, A combined genetic adaptive search (GeneAS) for engineering design, Computer Sciences and Informatics 26 (4) (1996) 30–45.
[28] S.Y. Ho, L.S. Shu, J.-H. Chen, Intelligent evolutionary algorithms for large parameteroptimization problems, IEEE Transactions on Evolutionary Computation 8 (6) (2004) 522–541.
[29] M. Pelikan, D.E. Goldberg, E. Cantu-Paz, BOA: the Bayesian optimization algorithm, in: Procedings of the 1st Conference GECCO-99, 1999, pp. 525–532.
[30] Q. Zhang, Y.W. Leung, An orthogonal genetic algorithm for multimedia multicast routine, IEEE Transactions on Evolutionary Computation 3 (1999) 53–62.