بهبود کارایی الگوریتم‌های بهینه‌یابی در محیط‌های پویا با استفاده از الگوریتم جستجوی هارمونی خودتطبیق

طیبه طاهری [۱]، بابک نصیری[۲]، محمدرضا میبدی[۳]

[۱] دانشکده برق ، رایانه و فناوری اطلاعات، دانشگاه آزاد اسلامی، قزوین
Taheri_tayebeh2002@yahoo.com

[۲] دانشکده برق ، رایانه و فناوری اطلاعات، دانشگاه آزاد اسلامی، قزوین
nasiri_babak@yahoo.com

[۳]دانشکده مهندسی کامپیوتر، فناوری اطلاعات، دانشگاه صنعتی امیرکبیر، تهران
mmeybodi@aut.ac.ir

چکیده

الگوریتم جستجوی هارمونی یکی از الگوریتم‌های تکاملی بوده که از موسیقی الهام گرفته شده است و مبتنی بر جستجوی تصادفی است. از کاربردهای آن می‌توان به بهینه‌سازی توابع خطی و غیرخطی، دسته‌بندی داده‌ها و حل مسائل بهینه سازی اشاره نمود. الگوریتم جستجوی هارمونی تا کنون برروی مسائل ایستا اجرا شده و نتایج قابل قبولی را ارائه داده است ولی اکثر مسائل در دنیای واقعی پویا بوده و تابع هدف، متغیرهای محیطی یا محدودیت های آن در طول زمان تغییر می‌کند که در نتیجه، راه حل بهینه بطور پیوسته در حال تغییر می‌باشد. از اینرو لازم است که الگوریتم‌های بهینه سازی نیز متناسب با آن درحال تغییر باشند. در این مقاله از الگوریتم جستجوی هارمونی استاندارد برای بهینه‌یابی سراسری در کنار الگوریتم جستجوی هارمونی محدود شده به یک شعاع در بهینه‌یابی محلی بصورت همکارانه استفاده شده است. الگوریتم مذکور بر روی معیار قله‌های متحرک که از شناخته شده‌ترین معیارها در محیط پویا هستند ارزیابی شده و با نتایج حاصل از چندین الگوریتم معتبر مورد مقایسه قرار گرفته است. نتایج آزمایشات نشان دهنده صحت و کارایی روش ارائه شده می‌باشد.

کلمات کلیدی

الگوریتم جستجوی هارمونی، محیط پویا، معیار قله‌های متحرک.

۱- مقدمه

الگوریتم جستجوی هارمونی[۱](HS) یکی از الگوریتم‌های الهام گرفته شده از طبیعت است. این الگوریتم اولین بار توسط ژونگ‌هوجیم در سال ۲۰۰۱ ارائه گردید[۱]. الگوریتم جستجوی هارمونی برگرفته شده از فعالیت موسیقی موسیقی‌دانان است که به دنبال پیدا کردن یک هارمونی کامل و بی نقص هستند[۲]. از جمله مزایای آن می‌توان به غیر حساس بودن به مقادیر اولیه، جایگزین کردن جستجوی تصادفی بجای جستجوی گرادیانی و استفاده کمتر از ریاضیات اشاره کرد[۲]. این الگوریتم در کاربردهای بهینه‌سازی از قبیل: دسته‌بندی داده‌ها[۳]، بهینه‌سازی ترکیبی[۴][۵][۶] و غیره استفاده شده است.

بیشتر مسائل دنیای واقعی بطور پویا و تصادفی در طول زمان در حال تغییر هستند. این مسائل به روش‌های بهینه‌سازی قدرتمندی نیاز دارند تا هنگام برخورد با عدم قطعیت دنیای واقعی توان مقابله را داشته باشند. یکی از روشهای برخورد با عدم قطعیت استفاده از روشهای تکاملی است[۷]. مسائل غیرقطعی که تاکنون مورد بررسی قرار گرفته‌اند را می‌توان به چهار گروه تقسیم کرد: وجود نویز در تابع ارزیاب، آشفتگی در متغیرهای طراحی، تقریبی بودن تابع ارزیاب و پویا بودن راه‌حل بهینه. دراین مقاله از پویا بودن راه‌حل بهینه به عنوان عدم قطعیت استفاده شده است. حافظه غیر معتبر و از دست دادن تنوع دو مشکل عمده در روشهای پردازش تکاملی است. این دو مشکل کارایی

روشهای پردازش تکاملی در محیط پویا را کاهش می‌دهند. هنگامی که محیط تغییر می‌کند، راه‌حلهای بدست آمده موجود در حافظه دیگر معتبر نمی‌باشند، یا باید آنها را بطور کامل فراموش کرد و یا باید دوباره آنها را ارزیابی کرد. از طرف دیگر، از آنجا که بیشتر روشهای پردازش تکاملی به دلیل ماهیتشان به یک نقطه همگرا می‌شوند، لذا تنوع دسته در محیط از بین می‌رود و در صورت تغییر در محیط، همگرا شدن به یک نقطه بهینه جدید، زمان زیادی را هدر می‌دهد. روشهای زیادی برای ایجاد یا نگهداری تنوع دسته در محیط وجود دارد. در نظر گرفتن هر تغییر به عنوان ورودی یک مسئله بهینه سازی جدید، می‌تواند ساده‌ترین راه برای واکنش در برابر تغییرات محیط باشد که باید از ابتدا حل شود. درصورتی که زمان انجام کار مهم نباشد، این یک گزینه مناسب است اما در بیشتر مواقع، زمان ارزش زیادی داشته و این راه‌حل مناسب نمی‌باشد. یک تلاش طبیعی برای سرعت بخشیدن به فرآیند بهینه‌سازی بعد از یک تغییر، این است که از اطلاعات قبلی برای محیط جدید استفاده کنیم. سوال اساسی که در اینجا مطرح می‌شود این است که چه اطلاعاتی باید نگهداری شود و این اطلاعات چگونه باید در محیط جدید مورد استفاده قرار گیرد تا فرآیند جستجو به درستی انجام پذیرد. مسئله دیگر این است که در بیشتر الگوریتم‌های بهینه‌سازی، پس از اینکه الگوریتم به یک نقطه همگرا شد، تنوع خود را از دست می‌دهد که این امر باعث کاهش قابلیت انطباق و سازگاری با تغییر محیط می‌شود، بنابراین در کنار انتقال اطلاعات باید به دنبال راهکارهایی برای افزایش تنوع و سازگاری الگوریتم پس از تغییر محیط باشیم.

تاکنون برای بهینه‌سازی در محیط‌های پویا از روشهای پردازش تکاملی و هوش جمعی مختلفی از جمله الگوریتم بهینه‌سازی دسته ذرات(pso[۲])[۸][۹]، الگوریتم ماهی‌های مصنوعی(AFSA[۳])[۱۰][۱۱]، و غیره استفاده شده است. در این الگوریتم‌ها سعی شده با بهبود برخی از ویژگی‌های فوق نتایج بهتری حاصل شود. در این مقاله یک الگوریتم بهینه‌سازی مبتنی بر الگوریتم HS پیشنهاد می‌گردد که برای عمل در محیط‌های پویا طراحی شده است و تمام الزامات محیط‌های پویا را برآورده می‌کند. در الگوریتم پیشنهادی از مکانیزم HS استاندارد استفاده شده است با این تفاوت که بعد از هر بار تکرار از یک روش مبتنی بر شعاع همگرایی بهترین هارمونی استفاده شده است. الگوریتم پیشنهادی بر روی بنچمارک قله-های متحرک (MPB[۴])[۱۲] که از معروفترین بنچمارکهای محیط پویاست ارزیابی شده و کارایی آن با شش الگوریتم معتبر دیگر بنام-های،mQSO[۸]، cellular pso[۱۳]۱، fmso[۱۴]، rpso[۱۵] مقایسه شده است. معیار مقایسه خطای برون خطی[۵] می‌باشد که یکی از معیارهای اصلی مقایسه الگوریتم‌های طراحی شده برای محیط‌های پویا است. نتایج آزمایشات، صحت و کارایی الگوریتم پیشنهادی را اثبات می‌کند.

ادامه این مقاله بدین صورت سازماندهی شده است: بخش۲ الگوریتم جستجوی هارمونی را به طور مختصر شرح می‌دهد، بخش ۳توضیح مختصری در مورد کارهای انجام گرفته در زمینه محیط‌های پویا ارائه کرده، بخش۴ الگوریتم پیشنهادی را توضیح می‌دهد. در بخش ۵ نتایج آزمایشات مورد بررسی قرار می‌گیرد و بخش ۶ به بیان نتیجه‌گیری می‌پردازد.

۲- الگوریتم جستجوی هارمونی

الگوریتم جستجوی هارمونی یکی از تکنیکهای بهینه‌سازی است که از فرآیند موسیقی الهام گرفته شده است. از جمله مزایای این الگوریتم می‌توان به موارد زیر اشاره کرد: a- آسان برای درک، آسان برای پیاده-سازی و بی‌نیاز از مسائل و فرمولهای سخت ریاضی. b- از آنجا که این الگوریتم از اعداد تصادفی استفاده می‌کند نیاز به هیچ اطلاعات اضافی نداریم. c- برخلاف دیگر الگوریتم‌ها که حداکثر از دوبردار برای تولید نسل جدید استفاده می‌کنند این الگوریتم از تمام بردارها برای تولید نسل جدید استفاده می‌کند.

مراحل الگوریتم HS به ترتیب زیر می‌باشد:

۱.تعریف مسئله و پارامترها.

۲.مقدار دهی اولیه حافظه هارمونی.

۳.تولید هارمونی جدید.

۴.به روز رسانی حافظه هارمونی.

۵. بررسی شرط پایان الگوریتم.

شبه کد الگوریتم HS در شکل۱ آورده شده است.

```
begin
   Define objective function f(x), x=(x1,x2, ...,xd)^T
   Define harmony memory accepting rate (raccept)
   Define pitch adjusting rate (rpa) and other parameters
   Generate Harmony Memory with random harmonies
  while ( t<max number of iterations )
      while ( i<=number of variables)
         if (rand<r_accept),
            Choose a value from HM  for the variable i
            if (rand<r_pa),
               Adjust the value by adding certain amount
            end if
         else Choose a random value
         end if
      end while
      Accept the new harmony (solution) if better
  end while
  Find the current best solution
end
```

شکل ۱ : شبه کدی از الگوریتم جستجوی هارمونی استاندارد

۳-کارهای گذشته

از بین چالش‌های موجود در محیط‌های پویا مهمترین چالش مسئله از دست دادن تنوع می‌باشد. برای حل این مشکل راهکارهایی ارائه شده است. تعدای از این راهکارها، تصادفی سازی در تمام یا بخشی از گروه

را دربرمی‌گیرد[17]. تعدای دیگر، از راه‌حل دفع کردن برای ایجاد تنوع استفاده می‌کند[20-18]. تنظیم ساختار اشتراک اطلاعات راه دیگری است که می‌تواند با هدف کاهش موقتی انگیزه برای جابجایی به‌سوی بهترین موقعیت سراسری باشد و از این راه تنوع جمعیت را افزایش دهد[21,16]. در بسیاری دیگر برای این منظور از مدل چند گروهی استفاده شده است. مدل Mpso تنوع را در دو سطح حفظ می‌کند. در الگوریتم spso تمام ذرات بر اساس ترتیب نزولی مقدار برازندگی بهترین‌های شخصی هر ذره مرتب می‌شود.

۴- الگوریتم پیشنهادی

در این الگوریتم همانند[۸] از چند دستگی استفاده شده است. بدین صورت که کل هارمونی‌های موجود را به دسته‌های از پیش تعیین شده‌ای تقسیم کرده است. ارتباط بین دسته‌ها به دوصورت محلی براساس r_{exle} و سراسری بر اساس r_{conv} می‌باشد[۸]. در واقع هنگامی که تمام دسته‌ها همگرا شده و تعداد قله‌ها بیش از تعداد دسته‌ها باشد، دسته‌ای که کمترین مقدار شایستگی را داشته باشد دوباره مقدار دهی اولیه می‌شود. همچنین اگر دو دسته در محدوده r_{exle} باشند دسته با کمترین مقدار شایستگی برای مقدار دهی اولیه انتخاب می‌شود. این کار بدین منظور است که تمام قله‌ها تحت پوشش قرار گیرند. وقتی که تعداد قله‌ها بیش از تعداد دسته‌ها باشد بعضی از قله‌ها هیچ‌وقت شناسایی نمی‌شوند، بنابراین همانطور که در [۸] پیشنهاد شده در چنین مواقعی دسته‌ای که کمترین مقدار شایستگی را دارد، برای مقداردهی اولیه انتخاب می‌شود تا شانس پیدا کردن قله‌های دیگر که احتمالا بلندتر هستند نیز بوجود آید. در الگوریتم پیشنهادی نیز از این دو تکنیک به ترتیب به نامهای anti-convergence و exclusion استفاده شده است.

همانطور که در بالا ذکر شد الگوریتم پیشنهادی کل هارمونی‌های موجود را به دسته‌های از پیش تعیین شده‌ای تقسیم می‌کند و مقدار شایستگی هر هارمونی محاسبه می‌شود. سپس بهترین هارمونی هر دسته مشخص شده و سایر هارمونی‌های آن دسته را به اندازه شعاع همگرایی دسته طبق رابطه ۵ حرکت می‌دهد.

$$X_i(t+1) = x_i(t) + (R_{cloud} \times rand) \qquad (۱)$$

که در رابطه (۱) x_i بردار هارمونی iام و R_{cloud} شعاع همگرایی دسته، rand یک عدد تصادفی در فاصله [1,1-] است.

با این روند هارمونی‌های هر دسته در یک دایره به مرکزیت بهترین هارمونی و شعاع همگرایی آن دسته قرار می‌گیرند. این عمل باعث می‌شود تا هارمونی‌های ایجاد شده در مرحله بعد در محدوده همان شعاع قرار گیرد. در گام بعدی به ازای هر دسته طبق الگوریتم HS، هارمونی جدیدی ایجاد شده و مقدار شایستگی آن ارزیابی می‌شود. مقدار شایستگی هارمونی جدید هر دسته با بدترین هارمونی آن دسته مقایسه می‌شود و در صورتی که از آن بهتر بود جایگزین بدترین هارمونی موجود می‌شود. الگوریتم HS تضمین می‌کند که در هر تکرار هارمونی بهتری در صورت وجود جایگزین شود. این روند تا جایی ادامه پیدا می‌کند که دسته مورد نظر به یک قله همگرا شود. زمانی که دسته همگرا شد از یک تکنیک به نام خواباندن دسته استفاده می‌کنیم. بدین معنی که آن دسته را از چرخه جستجو خارج کرده تا تعداد مقایسه‌ها کاهش یابد و عمل جستجو روی سایر دسته‌ها متمرکز شود. در الگوریتم پیشنهادی دسته دیگری به عنوان دسته ناظر ایجاد کرده تا کل محیط را جستجو کند که اگر قله‌ای وجود داشت که توسط هیچ دسته‌ای شناسایی نشده بود دسته‌ای را که هیچ قله‌ای را پیدا نکرده به آن معرفی کند. و خود دوباره به محیط باز گشته و به جستجوی قله‌های ناشناس می‌پردازد. دسته ناظر نیز از الگوریتم HS معرفی شده در قسمت جستجوی محلی برای بهبود هارمونی‌های خود استفاده می‌کند با این تفاوت که به هیچ شعاعی محدود نشده تا بتواند کل محیط را بررسی کند.

همانطور که از رابطه ۱ مشاهده می‌شود مقدار شعاع R_{cloud} در تعیین موقعیت سایر هارمونی‌ها نقش اساسی دارد. در واقع کارایی HS محلی تا حد زیادی وابسته به اندازه این شعاع است. در صورتی که مقدار R_{cloud} کوچک باشد توانایی جستجوی محلی الگوریتم بسیار بالا می‌رود، زیرا هارمونی‌های جدید در محدوده کوچکی از فضا ایجاد خواهد شد و شباهت زیادی به سایر هارمونی‌ها خواهد داشت اما باعث کاهش طول گام حرکت در هر تکرار و در نتیجه کاهش سرعت همگرایی الگوریتم می‌شودو از توانایی تنوع دسته نیز کاسته می‌شود. در صورتی که مقدار این شعاع بزرگ در نظر گرفته شود، احتمال ایجاد هارمونی‌های بهتر کاهش می‌یابد و توانایی جستجوی محلی نیز کم می‌شود. در واقع مقدار این پارامتر باید به گونه‌ای تعیین شود که بتواند مشکل کاهش تنوع را در دسته پس از تغییر محیط به سرعت از بین ببرد و پس از آن کوچکتر شود تا دسته بتواند با انجام یک جستجوی محلی قوی‌تر به نتایج بهتری دست یابد. در الگوریتم پیشنهادی برای دستیابی به این هدف مقدار R_{cloud} را دوبرابر طول گام حرکت قله‌ها در نظر گرفته سپس با استفاده از یک رهیافت خود تطبیقی مقدار این پارامتر کاهش می‌یابد. شبه کد الگوریتم پیشنهادی در شکل ۲ نشان داده شده است.

```
// initialize swarms
for each swarm i
    for each ahrmony j
        initialize x_{i,j}
        pbest _{i,j} = x_{i,j}
    end
    Gbest= arg min f(pbest_{i,j})
end
 // initialize swarm finder
for each harmony finder
    initialize x_{i,j}
end
 repeat:
for each swarm i
    for each harmony j
        update x_{i,j} based on equation 1
    end
       call HS Algorithm
end
call HS Algorithm for finder
if finder convergence to peak
    replace worst swarm with finder
end
Execute exclusion and anti-convergence
for each swarm i
    if swarm i convergence to peak
        swarm i is sleep
    end
end
until stopping criterion is met
```

شکل ۲: شبه کد الگوریتم پیشنهادی

۵- نتایج آزمایشات

اکثر شبیه‌سازی تغییرات محیط پویا در زمان، روی MPB که یکی از شناخته شده‌ترین معیارها برای ارزیابی است، صورت می‌گیرد. محیط آن شامل تعداد مشخصی قله با محل، ارتفاع و عرضهای مختلف می‌باشد. برازندگی محیط برابر ماکزیمم مقدار در میان توابع پیک تعریف می‌شود.

برای بررسی کارایی الگوریتم پیشنهادی، این الگوریتم را با چهار الگوریتم شناخته شده دیگر به نامهای mQSO[۸] ، cellular pso۱[۱۳]، fmso[۱۴]، rpso[۱۵] مقایسه شده است. نتایج با توجه به پارامترهای جدول ۱ که با نام سناریوی ۲ در مسئله MPB معروف است، بدست آمده‌اند. تنها تفاوت آن با سناریوی ۲، در تعداد قله‌هاست که بین ۱ قله تا ۲۰۰ قله در فرکانس‌های تغییر ، ۵۰۰، ۱۰۰۰، ۵۰۰۰ درنظر گرفته شده است.

جدول۱: پارامترهای MPB

پارامتر	مقدار
M تعداد قله‌ها	بین ۱ تا ۲۰۰
فرکانس تغییر	۵۰۰۰و۱۰۰۰و۵۰۰
میزان تغییر ارتفاع	۷
میزان تغییر عرض	۱
شکل قله	cone
تابع اولیه	ندارد
Sطول جابجایی	۱
Nتعداد ابعاد	۵
محدوده مکانی قله‌ها	[۰و۱۰۰و۰]
محدوده پارامتر ارتفاع	[۳۰و۷۰]
محدوده پارامتر عرض	[۱و۱۲]
مقدار ارتفاع اولیه قله‌ها	۵۰

نتایج بدست آمده از خطای برون خطی و خطای استاندارد بر روی قله‌های مختلف در جدول (۲) نشان داده شده است. خطای استاندارد درون پرانتز نوشته شده است. در این جدول p نشان دهنده تعداد قله‌هاست. در جدول (۲)، بهترین نتیجه هر سطر و الگوریتم پیشنهادی بصورت پررنگ مشخص شده است. همانطور که مشاهده می‌شود، الگوریتم پیشنهادی نتایج قابل قبولی را ارائه کرده است. علی‌الخصوص زمانی که تعداد قله‌ها با تعداد دسته‌ها برابر است. زمانی که تعداد قله‌ها کمتر از تعداد دسته‌هاست چون ممکن است چند دسته برروی یک قله باهم رقابت کنند، در نتیجه دسته‌ای که شایستگی بدتری دارد بازنده شده و مجبور است دوباره مقدار دهی شود و این باعث می‌شود نتایج بدست آمده با حالتی که تعداد قله‌ها برابرند اختلاف چشمگیری داشته باشد. ولی با این وجود الگوریتم پیشنهادی در این حالت نیز بهتر از سایر الگوریتم‌ها عمل کرده چون حتی بعد از مقدار دهی نیز این دسته باید هارمونی‌های قابل توجه‌ای نسبت به دسته برنده داشته باشد تا بتواند آن قله را در اختیار بگیرد. اما زمانی که تعداد قله‌ها خیلی بیشتر از تعداد دسته‌ها می‌شود، تعدادی از قله‌ها بدون پوشش باقی مانده و در نتیجه الگوریتم نتایج بهتری را نمی‌تواند ارائه کند. هرچند مکانیزم اجرا شده تا حدودی این مشکل را برطرف نموده ولی بازهم نتیجه مورد انتظار را نمی‌دهد.

جدول(۲): خطای برون خطی و خطای استاندارد الگوریتم‌ها در فرکانس‌های تغییرات ۵۰۰۰و۱۰۰۰و۵۰۰ با تعداد قله‌های مختلف

frequency	P	Cellular PSO	rPSO	mQSO	FMSO	الگوریتم پیشنهادی
۵۰۰۰	۱	۲,۵۵(۰,۱۲)	**۰,۵۶(۰,۰۴)**	۵,۰۷(۰,۱۷)	۳,۴۴(۰,۱۱)	**۰,۹۵(۰,۰۷)**
	۵	**۱,۶۸(۰,۱۱)**	۱۲,۲۲(۰,۷۶)	۱,۸۱(۰,۰۷)	۲,۹۴(۰,۰۷)	**۱,۸۶(۰,۰۸)**
	۱۰	۱,۷۸(۰,۰۵)	۱۲,۹۸(۰,۴۸)	**۱,۷۵(۰,۰۶)**	۳,۱۱(۰,۰۶)	**۱,۷۲(۰,۰۸)**
	۲۰	**۲,۶۰(۰,۰۷)**	۱۲,۷۹(۰,۵۴)	۲,۷۴(۰,۰۷)	۳,۳۶(۰,۰۶)	**۲,۰۷(۰,۱۲)**
	۳۰	**۲,۹۳(۰,۰۸)**	۱۲,۳۵(۰,۶۲)	۳,۲۷(۰,۱۱)	۳,۲۸(۰,۰۵)	**۲,۶۹(۰,۱۰)**
	۴۰	**۳,۱۴(۰,۰۸)**	۱۱,۳۷(۰,۴۱)	۳,۶۰(۰,۰۸)	۳,۲۶(۰,۰۴)	**۲,۸۷(۰,۱۰)**
	۵۰	۳,۲۶(۰,۰۸)	۱۱,۳۴(۰,۲۹)	۳,۶۵(۰,۱۱)	**۳,۲۲(۰,۰۵)**	**۲,۹۰(۰,۱۰)**
	۱۰۰	۳,۴۱(۰,۰۷)	۹,۷۳(۰,۲۸)	۳,۹۳(۰,۰۸)	**۳,۰۶(۰,۰۴)**	**۳,۱۴(۰,۱۱)**
	۲۰۰	۳,۴۰(۰,۰۶)	۸,۹۰(۰,۱۹)	۳,۸۶(۰,۰۷)	**۲,۸۴(۰,۰۳)**	**۳,۹۳(۰,۱۲)**
۱۰۰۰	۱	۶,۷۷(۰,۱۲)	**۱,۹۴(۰,۰۸)**	۱۸,۶(۱,۶)	۱۴,۴۲(۰,۴۸)	**۱۰,۳۲(۰,۰۸)**
	۵	**۵,۳۰(۰,۳۲)**	۱۳,۷۷(۰,۴۷)	۶,۵۶(۰,۳۸)	۱۰,۵۹(۰,۲۴)	**۷,۹۴(۰,۰۸)**
	۱۰	**۵,۱۵(۰,۱۳)**	۱۵,۵۵(۰,۵۱)	۵,۷۱(۰,۲۲)	۱۰,۴۰(۰,۱۷)	**۴,۶۰(۰,۱۹)**
	۲۰	**۵,۳۲(۰,۱۸)**	۱۵,۵۴(۰,۴۵)	۵,۹۰(۰,۱۷)	۱۰,۳۳(۰,۱۳)	**۴,۸۹(۰,۰۹)**
	۳۰	**۵,۹۳(۰,۲۰)**	۱۴,۳۸(۰,۳۸)	۶,۰۲(۰,۱۴)	۱۰,۰۶(۰,۱۴)	**۵,۱۰(۰,۰۹)**
	۴۰	**۵,۹۴(۰,۱۸)**	۱۴,۱۱(۰,۳۳)	۶,۲۲(۰,۱۶)	۹,۸۵(۰,۱۱)	**۵,۱۱(۰,۱۰)**
	۵۰	**۵,۵۵(۰,۱۴)**	۱۳,۷۵(۰,۲۹)	۵,۹۰(۰,۱۷)	۹,۵۴(۰,۱۱)	**۶,۲۱(۰,۰۹)**
	۱۰۰	**۶,۲۷(۰,۱۴)**	۱۲,۲۷(۰,۲۳)	۶,۴۸(۰,۱۵)	۸,۷۷(۰,۰۹)	**۶,۷۷(۰,۱۱)**
	۲۰۰	**۶,۰۱(۰,۱۱)**	۱۱,۳۲(۰,۱۶)	۶,۱۸(۰,۱۳)	۸,۰۶(۰,۰۷)	**۶,۸۰(۰,۱۰)**
۵۰۰	۱	۱۳,۴(۰,۷۴)	**۴,۲۷(۰,۱۷)**	۳۳,۶۷(۳,۴)	۲۷,۵۸(۰,۹۴)	**۱۸,۶۹(۱,۰۴)**
	۵	**۹,۶۳(۰,۴۹)**	۱۶,۱۹(۰,۴۲)	۱۱,۹۱(۰,۷۶)	۱۹,۴۵(۰,۴۵)	**۹,۸۳(۰,۶۲)**
	۱۰	**۹,۴۲(۰,۲۱)**	۱۷,۳۴(۰,۲۷)	۹,۶۲(۰,۳۴)	۱۸,۲۶(۰,۳۲)	**۶,۹۰(۰,۱۷)**
	۲۰	۸,۸۴(۰,۲۸)	۱۷,۰۶(۰,۳۷)	**۸,۷۹(۰,۲۳)**	۱۷,۳۴(۰,۳۰)	**۷,۲۱(۰,۲۱)**
	۳۰	**۷,۸۸(۰,۲۳)**	۱۶,۹۸(۰,۲۷)	۸,۸۰(۰,۲۱)	۱۶,۳۹(۰,۴۸)	**۷,۹۰(۰,۲۱)**
	۴۰	**۷,۸۳(۰,۲۱)**	۱۶,۴۶(۰,۲۸)	۸,۵۵(۰,۲۱)	۱۵,۳۴(۰,۴۵)	**۸,۲۳(۰,۶۵)**
	۵۰	۸,۶۲(۰,۲۳)	۱۵,۷۷(۰,۱۸)	**۸,۵۶(۰,۲)**	۱۵,۵۴(۰,۲۶)	**۸,۵۵(۰,۲۴)**
	۱۰۰	۱۱,۳۸ (۰,۲۳)	۱۴,۵۵(۰,۲۲)	**۸,۵۴(۰,۱۶)**	۱۲,۸۷(۰,۶۰)	**۱۰,۰۲(۰,۳۲)**
	۲۰۰	۱۱,۳۴(۰,۲۷)	۱۳,۴۰(۰,۱۳)	**۸,۱۹(۰,۱۷)**	۱۱,۵۲(۰,۱۶)	**۱۲,۸۷(۰,۳۸)**

۶- نتیجه‌گیری

در این مقاله از الگوریتم جستجوی هارمونی و شعاع همگرایی و همچنین مکانیزم خواباندن دسته استفاده شده و نتایج آن بر روی تابع بنچمارک قله‌های متحرک با چندین روش شناخته شده دیگر مقایسه گردید. نتایج آزمایشات نشان داد که روش پیشنهادی کارایی قابل قبولی مخصوصا زمانی که تعداد قله‌ها کمتر از تعداد دسته‌ها باشد و بهترین نتیجه مربوط به سطری است که تعداد قله‌ها و دسته‌ها برابر است. اما هرچه تعداد قله‌ها افزایش پیدا می‌کند نیجه بدتر می‌شود. بنابراین اگر بتوان از ابتدا تعداد دسته ها را نیز مطابق تعداد قله‌ها انتخاب کنیم نتایج فوق‌العاده خواهد شد. از طرف دیگر اگر از ابتدا دسته جستجوگر شروع به جستجو کند و زمانی که به قله‌ای همگرا شد یکی از دسته‌ها جایگزین آن شود تابع ارزیابی کمتری استفاده شده و نتیجه بهبود قابل توجهی خواهد داشت.

مراجع

[1] Geem, Z.W., Kim, J.H., Loganathan, G.V, "A new heuristic optimization algorithm: Harmony search". Simulation 76, 60–68 (2001).

[2] Xin-She Yang, "Harmony Search as a Metaheuristic Algorithm", Springer-Verlag Berlin Heidelberg, vol. 191, 1-14 (2009).

[3] M. Mahdavi, M. Haghir Chehreghani, H. Abolhassani, R. Forsati, "Novel meta-heuristic algorithms for clustering web documents", Applied Mathematics and Computation 201, 441–451 (2008).

[4] M. Fesanghary, M. Mahdavi, M. Minary-Jolandan, Y. Alizadeh, ,"Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems", Comput. Methods Appl. Mech. Engrg. 197, 3080–3091 (2008).

[5] Osama Moh'd Alia, Rajeswari Mandava , Mohd Ezane Aziz, "A hybrid harmony search algorithm for MRI brain segmentation", Springer, Evol. Intel 4, 31–49 (2011).

[6] A. Kaveh, S. Talatahari," Particle swarm optimizer, ant colony strategy and harmony search scheme

hybridized for optimization of truss structures", Computers and Structures 87, 267-283, (2009).

[7] Y. Jin and J. Branke, "Evolutionary Optimization in uncertain environments –A Survey", in IEEE Transaction on Evolutionary Computation, vol. 9 , No. 3 , pp. 303- 317(2005).

[8] T. Blackwell and J. Branke, "Multiswarm, Exclusion, and Anti-Convergence in Dynamic Environment", in IEEE Transaction on Evolutionary Computation, Vol. 10, No. 4, pp. 459-472, (2006).

[9] T. Blackwell and J. Branke, "Particle Swarms for Dynamic Optimization Problems", in Swarm Intelligence, pp.193–217, (2008).

[10] D. Yazdani, S. Golyari and M. R. Meybodi, "A New Hybrid Algorithm for Optimization Based on Artificial Fish Swarm Algorithm and Cellular Learning Automata", in 5th nternational Conference on Telecommunication (IST2010), Tehran, Iran, (2010).

[11] D. Yazdani and M. R. Meybodi, "AFSA-LA: A New Model for Optimization", in 15th Conference of Computer Society of Iran (CSICC2010), Tehran, Iran, (2010).

[12] http://www.aifb.unikarlsruhe.de/~jbr/MovPeaks/

[13] B. Hashemi and M. R. Meybodi, "Cellular PSO: A PSO for Dynamic Environments", in Advances in Computation and ntelligence, Lecture Notes in Computer Science, vol. 5821, pp. 422-433, (2009).

[14] H. Richter, "Memory Design for Constrained Dynamic Optimization Problems," Applications of Evolutionary Computation, pp. 552–561,(2010).

[15] D. Parrott and X. Li, "Locating and Tracking Multiple Dynamic Optima by A Particle Swarm Model Using peciation," in IEEE Transaction on Evolutionary Computation, vol. 10, No. 4, pp. 440–458, (2006).

[16] S. Janson and M. Middendorf. "A hierachical particle swarm optimizer for dynamic optimization problems," In: G. R. Raidl, editor, APPlications of Evolutionary Computing, volume 3005 of Lecture Notes in Computer Science. Springer, Berlin, Germany, pp.513–524, (2004).

[17] X. Hu and R.C. "Eberhart. Adaptive particle swarm optimisation: detection and response to dynamic system," In: Proc. of the 2002 Congress on Evolutionary Computation, IEEE Press, pp. 1666–1670,(2002).

[18] J. Vesterstrom T. Krink and J. Riget. "Particle swarm optimisation with spatial particle extension," In: Proc. of the 2002 Congress on Evolutionary Computation, IEEE Press, pp. 1474–1479,(2002).

[19] T. M. Blackwell and P. J. Bentley. "Dynamic search with charged swarms," In: W. B. Langdon et al., editor, Proc. of the 2002 Genetic and Evolutionary Computation Conference, Morgan Kaufmann, pp.19–26, (2002).

[20] T. M. Blackwell and P. Bentley. " on' push me! Collision avoiding swarms," In: Proc. of the 2002 Congress on Evolutionary Computation, IEEE Press pp. 1691–1696, (2002).

[21] X. Li and K. H. Dam. "Comparing particle swarms for tracking extrema in dynamic environments," In: Proc. of the 2003 Congress on Evolutionary Computation, IEEE Press, pp.1772–1779,(2003).

[1] Harmony Search Algorithm
[2] Particle Swarm Optimization
[3] Artificial Fish Swarm Algorithm
[4] Moving Peak Benchmark
[5] Offline Error